NIST Special Publication 800-57

RECOMMENDATION FOR KEY MANAGEMENT
Part 3: Application-Specific Key Management Guidance

Elaine Barker
William Burr
Alicia Jones
Timothy Polk
Scott Rose
Miles Smid

NIST Special Publication 800-57

RECOMMENDATION FOR KEY MANAGEMENT
Part 3: Application-Specific Key Management Guidance

Elaine Barker
William Burr
Alicia Jones
Timothy Polk
Scott Rose
National Institute of Standards and Technology

Miles Smid
Orion Security Solutions

DRAFT – August 2008

U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

National Institute of Standards and Technology
Patrick Gallagher, Acting Director

Abstract

Special Publication 800-57 provides cryptographic key management guidance. It consists of three parts. Part 1 provides general guidance and best practices for the management of cryptographic keying material. Part 2 provides guidance on policy and security planning requirements for U.S. government agencies. Finally, Part 3 provides guidance when using the cryptographic features of current systems.

KEY WORDS: accreditation; assurances; authentication; authorization; availability; backup; certification; compromise; confidentiality; cryptanalysis; cryptographic key; cryptographic module; digital signature; key management; key management policy; key recovery; private key; public key; public key infrastructure; security plan; trust anchor; validation.

Acknowledgements

The National Institute of Standards and Technology (NIST) gratefully acknowledges and appreciates contributions by Quynh Dang and Sheila Frankel of NIST and by Chris Bean of the National Security Agency. NIST also thanks the many contributions by the public and private sectors whose thoughtful and constructive comments improved the quality and usefulness of this publication.

Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This guideline has been prepared for use by Federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright. (Attribution would be appreciated by NIST.)

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official.

Conformance testing for implementations of key management as specified in this Recommendation will be conducted within the framework of the Cryptographic Module Validation Program (CMVP), a joint effort of NIST and the Communications Security Establishment of the Government of Canada. Cryptographic implementations must adhere to the requirements in this Recommendation in order to be validated under the CMVP. The requirements of this Recommendation are indicated by the word “shall.”

Overview

"Application Specific Key Management Guidance", Part 3 of the *Recommendation for Key Management* is intended primarily to help system administrators and system installers adequately secure applications based on product availability and organizational needs and to support organizational decisions about future procurements. The guide also provides information for end users regarding application options left under their control in normal use of the application. Recommendations are given for a select set of applications, namely:

Section 2 - Public Key Infrastructures (PKI)
Section 3 - Internet Protocol Security (IPsec)
Section 4 – Transport Layer Security (TLS)
Section 5 - Secure/Multipurpose Internet Mail Extensions (S/MIME)
Section 6 – Kerberos
Section 7 - Over-the-Air Rekeying of Digital Radios (OTAR)
Section 8 - Domain Name System Security Extensions (DNSEC)
Section 9 – Encrypted File Systems (EFS)
Section 10 – Secure Shell (SSH)
Section 11 – IEEE 802.1x Port Based Network Access Control

The following is provided for each topic:

- A brief description of the system under discussion that is intended to provide context for the security guidance,
- Recommended algorithm suites and key sizes and associated security and compliance issues,
- Recommendations concerning the use of the mechanism in its current form for the protection of Federal government information,
- Security considerations that may affect the security effectiveness of key management processes,
- General recommendations for purchase decision makers, system installers, system administrators and end users.

Although this document will be updated as mechanisms and techniques evolve, it may not always reflect a comprehensive view of current products and technical specifications.

RECOMMENDATION FOR KEY MANAGEMENT

Part 3: Application-Specific Key Management Guidance

Table of Contents

RECOMMENDATION FOR KEY MANAGEMENT
Part 3: Application-Specific Key Management Guidance

1 Introduction

1.1 Purpose

Part 3 of the *Recommendation for Key Management*, *Application-Specific Key Management Guidance*, is intended to address the key management issues associated with currently available cryptographic mechanisms. *General Guidance*, Part 1 of the *Recommendation for Key Management*, contains basic key management guidance for users, developers and system managers regarding the "best practices" associated with the generation and use of the various classes of cryptographic keying material. *General Organization and Management Requirements,* Part 2 of the Recommendation, provides a framework and general guidance to support establishing cryptographic key management within an organization, and a basis for satisfying the key management aspects of statutory and policy-based security planning requirements for Federal government organizations.

This document, Part 3 of the Recommendation, is designed for system installers, system administrators and end users of existing key management infrastructures, protocols, and other applications, as well as the people making purchasing decisions[1] for new systems using currently available technology. Note that end users who act as their own system installers, administrators and purchasing agents may find the guidance intended for administrators, installers and purchasers to be beneficial. In centrally managed organizations, the organization's management must establish a security policy that acts as a foundation for all end user guidance.

Recommendations are made for mechanisms designed to protect stored data and data in transit. This document will not provide a complete restatement of existing standards or implementation directives. Standards and guidelines with this level of detail are referenced where appropriate.

For each of the key management infrastructures, protocols, and applications addressed in Part 3, the following is provided:

- A brief description of the system under discussion that is intended to provide context for the security guidance,
- Recommended algorithm suites and key sizes and associated security and compliance issues,
- Recommendations concerning the use of the mechanism in its current form for the protection of Federal government information,
- Security considerations that may affect the security effectiveness of key management processes,

[1] This is not necessarily a procurement officer but likely a person making the decision on the IT product to be used.

- General recommendations for purchase decision makers, system installers, system administrators and end users.

The logistics of how one should obtain, store or transfer keys or key pairs within a given application or system are application and implementation-specific and beyond the scope of this document. In large Federal systems, these functions are frequently handled by system administrators or completed with direct guidance from system administrators. For end users faced with these tasks on their own, an informative appendix has been included with general information intended to point the end user in the right direction.

Since some of the infrastructures, protocols and applications addressed in this Recommendation will be refined or replaced over the next few years, the guidance provided herein will become obsolete. Similarly, it is anticipated that new infrastructures, protocols, and applications will be developed. Although this document will be updated as mechanisms and techniques evolve, it may not always reflect a comprehensive view of current products and technical specifications. Hence, references to version numbers or other implementation status information are provided to enable an evaluation of the applicability of particular elements of guidance to the specific version of an infrastructure, protocol, or application into which a mechanism is integrated.

Note that many of the applications described in Part 3 are currently in use by U.S. government agencies. Some of these applications were developed and implemented prior to the release of Part 1 of this Recommendation, and therefore, may not follow all of the principles identified in Part 1. The use of current implementations of these applications may be necessary until more carefully designed applications are available. It is very important that each implementation that does not comply with NIST standards and guidelines be evaluated for associated risks and that steps be taken to mitigate those risks.

1.2 Requirement Terms

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

1. **shall**: This term is used to indicate a requirement of a Federal Information Processing Standard (FIPS) or a requirement that must be fulfilled to claim conformance to this Recommendation. Note that **shall** may be coupled with **not** to become **shall not**.
2. **should**: This term is used to indicate an important recommendation. Ignoring the recommendation could result in undesirable results. Ignoring recommendations to accommodate the acceptance of messages protected with commonly used, unapproved ciphers may create interoperability issues. Ignoring recommendations to select new products with approved, seldom used cryptographic mechanisms may leave an organization ill-prepared to migrate away from mechanisms that will soon be inappropriate for the protection of Federal systems. Note that **should** may be coupled with **not** to become **should not**.

1.3 General Protocol Considerations

There are a number of general issues associated with the protocols discussed in Part 3. Four of these issues are briefly discussed in order to familiarize the reader with concepts that will be repeated throughout the document and to help frame the upcoming discussions:

- Mandatory-to-implement vs. optional-to-implement,
- Cryptographic negotiation,
- Single or multi-use keys, and
- Algorithm and key size transitions.

1.3.1 Mandatory-to-Implement versus Optional-to-Implement

Many of the cryptographic security services described in this document are based on public standards (e.g., IETF RFCs, American National Standards, etc.). In these standards, algorithms are frequently described as mandatory-to-implement or optional-to-implement. Neither of these terms provides information about the security of the algorithm.

Mandatory-to-implement algorithms will be in any product that meets the public standards, allowing interoperability between products.

Optional-to-implement algorithms tend to be next-generation algorithms that may become mandatory-to-implement algorithms in a future version of the standard. There could be considerable delay in the widespread use of these new algorithms for a variety of reasons, ranging from a need for supporting hardware or software upgrades, to issues of interoperability. For example, an algorithm that is optional-to-implement within an S/MIME protocol may not currently be supported by the system's cryptographic module. However, these algorithms often offer improved security that could significantly increase the longevity of the system. Therefore, one may want to consider buying products that support the optional-to-implement algorithms, even if those algorithms will not be available to all end users immediately.

As previously defined, the terms **shall** and **should** are used to provide information about whether algorithms have adequate security for use on Federal computer networks. As such, there may be mandatory-to-implement algorithms (e.g., DES or RC2) that do not provide adequate security, and this document will say they **shall not** be used. Similarly, there may be optional-to-implement algorithms that have greater security (e.g., AES) and which this document may say **should** or **shall** be used in a given situation.

The distinction between mandatory-to-implement and optional-to-implement is important when two users on different systems desire to communicate or when different levels of security may be required for different applications running on the same system. This is further discussed in the next section on cryptographic negotiation.

1.3.2 Cryptographic Negotiation

Parts 1 and 2 of this Recommendation establish a sound basis for selecting appropriate cryptographic algorithms and managing the corresponding cryptographic keys. However, enforcing these guidelines can be problematic for a number of reasons, including the unavailability of certain algorithms or key sizes, the preferences of the parties in communication or other system limitations. When servers dictate the algorithms used, the server may select the algorithms that optimize overall system performance rather than the algorithms that provide the highest level of security.

In some multi-party protocols where multiple algorithms are supported for the same purpose, a client can enforce the rules in Parts 1 and 2 through negotiation within the protocol. Some protocols (e.g., S/MIME) allow the initiating client to select the cryptographic algorithms without negotiating with the receiving client. In this case, as in the case where applications do not permit negotiation, a receiving client may be presented with information that has been inadequately protected. For example, a receiving client may receive a signed and encrypted S/MIME email message that was encrypted using DES and signed with a 512-bit RSA key[2]. Rejecting such messages does not necessarily enhance security (in this case, the message has already been sent over the Internet), but the receiving user should be aware that the security services purportedly provided by the digital signature and content encryption are suspect and cannot be depended upon. It may be appropriate to reject the message or terminate the protocol. A risk assessment and subsequent organizational policy may be required to determine the appropriate course of action.

In other protocols (e.g., TLS), the client proposes a set of options, and the server chooses from the proposed list during a negotiation phase of the protocol. Where negotiation is supported, protocols may be designed to negotiate cipher suites or to negotiate each algorithm independently. In either case, a client or server may be faced with a situation where the preferred algorithms of the client or server and the proposed algorithms of the other party are not of the same security strength, or where approved algorithms are not available.

Another issue may arise when a protocol is designed to negotiate algorithms, but not key sizes. In such a case, the clients may find themselves communicating with approved algorithms, but inadequate key sizes. For example, after negotiating for RSA signatures, the client might get a message signed with a 512-bit RSA key[3].

Enforcing the recommendations from Parts1 and 2 may also be complicated by system or application design decisions. Systems may have application-specific controls for cryptographic algorithms, or they may have system-wide controls. For example, a user may wish to restrict one application to using AES, and another to using TDEA, while the

[2] The DES algorithm and the 512-bit RSA key size do not provide adequate security (see Part 1).

[3] A 512-bit RSA key does not provide an acceptable security strength (see Part 1).

system design may only allow the use of TDEA. Often the only limitation on public key sizes is an indirect limitation through the choice of root CA keys. (See Section 2.1).

When there are a variety of algorithms or key sizes available for a given communication protocol, the following questions need to be addressed:

- Is negotiation mandatory, optional or unsupported?
- When negotiation is supported, who proposes the cryptographic mechanisms to be used, who selects the mechanisms, and what are the selection criteria?
- Is negotiation based on predefined cipher suites, or is each algorithm proposed independently?
- What is the granularity for the negotiation: just algorithms, both algorithms and key sizes, combinations of algorithms and/or key sizes, or protocol versions?
- What cannot be specified?

A good start at ensuring communication security in a multi-algorithm setting would be to:

- Limit the list of algorithms available to the application to those best suited for users of the system and those needed for interoperability,
- Adopt a policy that disallows sending messages using an inadequate level of protection,
- Adopt a policy explaining how to respond to messages received without adequate protection, and
- Adopt a policy explaining what to do when faced with a need for secure communications with a party using un-approved algorithms or inadequate key sizes.

1.3.3 Single or Multi-Use Keys

A major thrust from Part 1 of this Recommendation is that, in general, keys **should not** be used for multiple cryptographic purposes. For example, in most cases, the same keys **should not** be used to generate a digital signature and to establish other keying material (See Part 1, Section 8.1.5.1.1.2 for the rare exceptions to this guidance). It is less clear as to whether a digital signature key, for example, can be used only for a specific application (e.g., signing e-mail) or for multiple applications (e.g., for both signing e-mail and signing documents). In some cases, it may be acceptable for an application to share keys with other applications. In other cases, sharing keys may not be desirable. For example, best practices indicate that a server's TLS keys **should not** be used to support other applications. Even where keys are used to perform the same cryptographic operation (e.g., digital signatures), sharing keys may be inappropriate because one application could be providing one service (e.g., authentication), while a second application could be providing a different service (e.g., non-repudiation). It is important to remember that it may be a bad idea to use keys for multiple applications.

An agency **should** perform a risk assessment when considering the use of the same key for multiple applications.

1.3.4 Algorithm and Key Size Transition

Part 1 of this Recommendation provides timeframes for transitioning from algorithms and key sizes currently in use by many applications and protocols in order to increase the strength of the security mechanisms in the future. In many cases, the algorithms and key sizes required to provide adequate security are not available within the current implementations or are unavailable uniformly across the community of users that need to interoperate. Transitions to new algorithms or key sizes will not necessarily occur instantaneously, but will require gradual upgrades across a system. For example, a system owner may have the need to upgrade his system's email package before upgrading the cryptographic module. Hence, for a period of time, the system may be running with an email package capable of TDEA and AES 128 encryption, but a cryptographic module that can only handle TDEA. There will be a need to upgrade components of a system with new capabilities, while continuing to support the old capabilities, until all components have been upgraded.

During this transition period, interaction between components can proceed in one of the following ways:

1. Some means is provided to determine when the new security mechanism is available to all parties in a given transaction so that it can be used instead of the old security mechanism (e.g., using a protocol that negotiates the security mechanisms to be used). When the new security mechanism is not available to all parties involved in a transaction, the old security mechanism can be used. This approach has the advantage that when a set of parties have the newer mechanism, their transactions are protected at a higher security level. The disadvantage is that those transactions using the old security mechanism are not as well protected; this also raises the possibility that the same information could be sent in different transactions between two or more sets of parties using security mechanisms of different strengths – in effect, nullifying the higher security strength provided by the new security mechanism[4].
2. All components use the old security mechanism until all components have been updated; at that time, the system immediately transitions to the new capability. This approach has the potential problem that all components would not be updated by the deadline, thus providing inadequate protections for all information during the period following the deadline until such time as all components have been upgraded. However, this approach has the advantage that the same data will not be sent at two different security levels.[5]

Most of the applications and protocols discussed in Part 3 require an upgrade of the available security mechanisms to be compliant with Part 1. The following sections provide guidance on how the existing mechanisms may best be used until appropriate

[4] This becomes an issue when higher security level users are unaware that others may be using a lower security level mechanism to protect the same information.

[5] Assuming that data sent before the transition is not also sent after the transition.

upgrades can be made. Organizations and system administrators must determine the approach for transitioning to stronger security mechanisms within a system.

2 Public Key Infrastructure

2.1 Description

A Public Key Infrastructure (PKI) is the most common key management approach for the distribution of public keys. As described in SP 800-57, Part 1, public keys are used to establish security services after obtaining a variety of assurances: assurance of integrity, assurance of domain parameter validity, assurance of public key validity, and assurance of private key possession. In most cases, applications must also establish the identity of the user associated with this key pair. In a PKI, the infrastructure establishes the user's identity and the required assurances to provide a strong foundation for security services in PKI-enabled applications and protocols, including IPsec (Section 3), Transport Layer Security (Section 4), S/MIME secure mail (Section 5) and some versions of Kerberos (Section 6). This section presents basic guidance for PKI-based key management. For broader and more detailed information on PKI, see [SP 800-32].

Public key certificates bind two names to a public key, the user's name and the issuer's name, using a digital signature generated by the issuer. The user is the party authorized to use the private key associated with the public key in the certificate. The issuer is a trusted third party that generates and signs the certificate after verifying: the identity of the user; the validity of the public key, associated algorithms and any relevant parameters; and the user's possession of the corresponding private key. The issuer is known as a Certificate Authority (CA). In many cases, the CA will delegate responsibility for the verification of the subject's identity to a Registration Authority (RA). The certificate is used to distribute the user's public key to other interested parties, known as relying parties, since they rely on the assurances provided by the PKI and the certificate creation process.

CAs generally issue a self-signed certificate called a *root certificate* (sometimes also called a *trust anchor*); this is used by applications and protocols to validate the certificates issued by a CA. They play a key role in many protocols and applications, and are generally kept in what is often called a *root certificate store*. Much of the business of properly configuring applications and protocols consists of ensuring that only appropriate root certificates are loaded into the root certificate store. In Microsoft Windows operating systems, there are root certificate stores that are maintained by the operating system for various purposes that are shared by various Microsoft protocols and applications, and by other applications that may choose to use them. There is a similar "Keychain" facility in the Apple operating systems. Some applications, intended to be portable between operating systems, maintain their own root certificate stores.[6]

Certificates are generally issued in accordance with a *certificate policy*. Generally that policy can be found on the issuing CA's website. If an organization's policy, for example, is to accept only certificates that use at least 2048-bit RSA, 2048-bit DSA or

[6] The various Mozilla browsers and E-mail clients, and the Apache web servers are examples. Microsoft Internet Explorer, Outlook and Internet Information Server all use the Windows root certificate store; Apple Safari and Mail use the Keychain; and Mozilla Firefox, Thunderbird and SeaMonkey all use root certificate stores associated specifically with the application.

224-bit elliptic curve cryptography, and either SHA-1, SHA-224 or SHA-256, then the only practical way to ensure that public key sizes meet the requirements is usually to ensure that the root certificate store contains only root certificates with a certificate policy that requires these algorithms and key sizes in its subordinate certificates. Current applications that use PKI will check to ensure that a certificate has been issued under the root certificate in the application's root store, and that it has not been subsequently revoked, but will not otherwise check the suitability of the public key or hash algorithms used in the certificate – the application will simply use the specified keys to compute the mathematically correct results. So, correctly configuring root certificate stores is a critical step in key management.

The specifics of where the root certificate store is located and how it is managed for each application and protocol are beyond the scope of this Recommendation. Typically, however, there are menus for viewing and managing certificate stores in the browser applications, but this is subject to change with each product update. There may also be utilities and features in the operating system or application for centralized management by the system administrator. When a browser or other application encounters an unrecognized CA certificate, end users may be prompted to add that certificate to their permanent trusted certificate store, temporarily trust the certificate, or reject the certificate and close the application.

The most common certificate format is the X.509 version 3 (X.509v3) certificate [RFC 5280]. In addition to the user and issuer names and the public key, all X.509 certificates also include a digital signature, an issuance and expiration date, and identifiers that specify the cryptographic algorithm(s) to be used with the public key and signature. X.509v3 certificates include an extensibility feature; CAs usually include standard extensions in their certificates to indicate which cryptographic operations the public key was intended to support, the policy that governed certificate issuance, and where to find out if the certificate has been revoked (i.e., an authoritative source for certificate status information). CAs may also include "private" extensions in their certificates that contain information particular to an application or domain of users.

A relying party is an individual or organization that relies on the certificate and the CA that issued the certificate to provide valid information (see Appendix A). Before a relying party uses the public key in a certificate, he must determine whether the key used by the issuer to sign this certificate can be trusted. In the simplest case, the relying party knows about the issuer, and has already decided to trust certificates issued by that CA. CAs that a relying party trusts directly are called *trust anchors*. When multiple trust anchors are recognized, the set of trust anchors is referred to as the trust list.

In most cases, a relying party will wish to process user certificates that were signed by issuers other than a CA in its trust list. To support this goal, CAs issue cross certificates that bind another issuer's name to that issuer's public key. Cross certificates are an assertion that a public key may be used to verify signatures on other certificates. A relying party may be able to develop a certification path – a sequence of certificates – demonstrating that a user's public key certificate can be trusted, even though it was issued by a CA that is not in the relying party's trust list. All certification paths begin with a trust anchor, include zero or more intermediate certificates, and end with the certificate that contains the user's public key. This can be an iterative process, and finding the appropriate intermediate certificates (a.k.a., path discovery) is one of PKI's challenges.

The entire path must be examined to ensure that the certificates have not been revoked, were issued under appropriate policies, and that each public key is suitable for the use to which it has been put. This process is known as path validation.

As noted above, the certificate itself will usually include a pointer to an authoritative source for certificate status information. Certificate status information may be provided using one of two standard mechanisms:

- The most common source is a certificate revocation list, or CRL. An X.509 CRL contains a list of certificates issued by that CA that have been revoked, indicates when they were revoked, and may include the reason for revocation (see [RFC 5280]). If the serial number of an unexpired certificate does not appear on the CRL, then it is still valid. CRLs are digitally signed, like a certificate, so they can be distributed through untrusted systems. Most commonly, CRLs are distributed via LDAP[7] directories or web servers.
- An alternative source for this information is an Online Certificate Status Protocol (OCSP) responder (see [RFC 2560]). An OCSP responder is a trusted system, and provides signed status information, on a per certificate basis, in response to a request from a relying party. Relying parties can authenticate the response by verifying the OCSP responder's digital signature. As the OCSP responder is providing authoritative status information, there is generally a formal (e.g., contractual) relationship between the CA and OCSP responder.

In many cases, PKIs will also provide key recovery services (using Recovery Servers) to support business continuity. Key recovery services store private keys that support key establishment to ensure that the plaintext of encrypted data may be recovered in the future. These services can provide the private key to the user in the event of loss or failure of their cryptographic module, or to the user's management when policy or legal requirements exist. When supported, this service removes a key management burden from PKI-enabled applications.

This section provides guidance for general purpose PKIs when users from different organizations need to support a variety of applications. For large, general purpose PKIs, interoperability is an important consideration. Less commonly, PKIs may be deployed to support a small, closed community of users or for a single application, where wider interoperability is less important. The requirements within this section are focused on large, general purpose PKIs, such as the Federal PKI. For PKIs requiring less

[7] **Lightweight Directory Access Protocol** (LDAP) is a software protocol for enabling anyone to locate organizations, individuals, and other resources, such as files and devices in a network, whether on the Internet or on a corporate intranet. See RFC 4511 Lightweight Directory Access Protocol (LDAP): The Protocol and RFC 4512 Lightweight Directory Access Protocol (LDAP): Directory Information Models.

interoperability, these requirements **should** be evaluated for appropriateness within their systems. In general, cryptographic algorithm and key size standards **should** be met by all PKIs.

2.2 Security and Compliance Issues

2.2.1 Recommended Key Sizes and Algorithms

The following table summarizes the recommended key sizes for key pairs used by PKI users and infrastructure components. Some of the dates in this table are slightly different than those that appear in Part 1 of this document, but are consistent with those specified in [SP 800-78-1]. [8] At the writing of Part 1, the Federal PKI was a 2 certificate PKI, with one certificate being used for authentication and digital signatures and the other for key transport or key agreement. Part 1 states that a 1024 bit RSA key used for a digital signature and for authentication is approved for use through 2010. This was a pragmatic, product-based compromise aimed at supporting FIPS 201 compliance. Since authentication has no forward security implications, 1024 bit authentication keys are approved for use through 2013. On the contrary, keys used for digital signatures *do* have forward security implications and given anticipated advances in factoring techniques must be phased out by 2008.

[8] 1024 bit RSA is permitted to leverage current products and promote efficient adoption of FIPS 201, but must be phased out by 12/31/2013 for authentication keys and 12/31/2008 for digital signature and key management keys. Two key Triple-DES (2TDEA) authentication keys must be phased out by 12/31/2010. These requirements anticipate that digital signature and key management keys will be used to protect data for longer periods of time, while data enciphered solely for authentication is usually a random challenge (rather than sensitive information) and is generally not retained.

Table 2-1 Recommended Key Sizes

Key Type	Time Period for Use	Algorithms and Key Sizes
Authentication keys (for Users or Devices)	Through 12/31/2013	RSA (1024 or 2048 bits) ECDSA (Curve P-256)
	After 12/31/2013	RSA (2048 bits) ECDSA (Curve P-256)
Digital Signature and Key Establishment keys (for Users or Devices)	Through 12/31/2008	RSA signature or key transport (1024 or 2048 bits) Diffie-Hellman (1024 or 2048 bits) ECDSA, ECDH (Curves P-256 or P-384)
	After 12/31/2008	RSA signature or key transport (2048 bits) Diffie-Hellman (2048 bits) ECDSA, ECDH (Curves P-256 or P-384)
CA and OCSP Responder Signing Keys	N/A	RSA: 2048, 3072, or 4096 bits ECDSA: Curves P-256 or P-384

Note that some approved algorithms and key sizes are omitted to enhance interoperability. For example, only two of the elliptic curves are recommended for use in PKIs [FIPS 186-3]. Similarly, Elliptic Curve Diffie-Hellman is recommended to support key establishment, rather than Elliptic Curve MQV. As stated in the introduction to this section, other approved algorithms may be used where PKIs are deployed to support a small, closed community of users or a single application.

The following table summarizes the recommended algorithms, key sizes, and padding schemes for signing certificates, CRLs, and OCSP status messages.

Table 2-2 Recommended Signature Algorithms

Signature Generation Date	Public Key Algorithms and Key Sizes	Hash Algorithms	Padding Scheme
Through 12/31/2009	RSA (2048, 3072, or 4096 bits)	SHA-1	PKCS #1 v1.5
	RSA (2048, 3072, or 4096 bits)	SHA-256	PKCS #1 v1.5
	ECDSA (Curve P-256)	SHA-256	N/A
	ECDSA (Curve P-384)	SHA-384	N/A
1/1/2010 through 12/31/2010	RSA (2048, 3072, or 4096 bits)	SHA-1	PKCS #1 v1.5
		SHA-256	PKCS #1 v1.5, PSS
	ECDSA (Curve P-256)	SHA-256	N/A
	ECDSA (Curve P-384)	SHA-384	N/A
After 12/31/2010	RSA (2048, 3072, or 4096 bits)	SHA-256	PKCS #1 v1.5, PSS
	ECDSA (Curve P-256)	SHA-256	N/A
	ECDSA (Curve P-384)	SHA-384	N/A

User certificates containing RSA or Diffie-Hellman public keys **should** be signed using the RSA signature algorithm. User certificates containing elliptic curve public keys **should** be signed using ECDSA.

Not all combinations of algorithms and key sizes are appropriate for the protection of Federal government information. To enhance interoperability, users **should** obtain authentication, signature, and key establishment certificates with complementary algorithms for all public keys.[9] For most users, signature and key establishment keys **should** provide consistent cryptographic strength. The following table shows preferred combinations for user keys; periods of use for these combinations **should** be calculated using the most conservative dates specified in Table 2-1.

While symmetric key cryptography is not strictly required, block ciphers are used in practically all PKI implementations and PKI-enabled applications. All components using block ciphers **shall** support the AES-128 algorithm. To support legacy implementations, components that process RSA keys **should** support three key Triple-DEA [SP800-67].

[9] In general, protocols and applications are designed to use cryptographic algorithms from one mathematical family. For example, applications that encounter ECDSA digital signatures would expect to use elliptic curve Diffie-Hellman or MQV for key establishment services. Users that obtain an ECDSA certificate and an RSA key establishment certificate, for example, may find they cannot use the keys together in a single application. Other combinations, such as DSA and Diffie-Hellman certificates, are commonly used in combination. Users should obtain authentication, signature, and key establishment certificates that are complementary to ensure that the keys can be used together in applications and protocols.

Components that support P-384 elliptic curve keys and the SHA-384 algorithm **shall** support AES-256.

Table 2-3 Recommended Combinations of Algorithms and Key Sizes

Authentication Key Type	Signature Key	Key Establishment Key
RSA 1024	RSA 2048	RSA 2048
RSA 1024	RSA 2048	Diffie-Hellman 2048
RSA 2048	RSA 2048	RSA 2048
RSA 2048	RSA 2048	Diffie-Hellman 2048
ECDSA P-256	ECDSA P-256	ECDH P-256
ECDSA P-256	ECDSA P-384	ECDH P-384
ECDSA P-384	ECDSA P-384	ECDH P-384

2.3 Procurement Guidance

The following provides guidance for those responsible for making decisions on which products to purchase in support of a PKI.

2.3.1 CA/RA Software and Hardware:

1. CA and RA software **shall** support the Certificate Management Protocol (CMP) [RFC 4210] or Certificate Management Using CMS (CMC) [RFC 5272]. CA software **should** also generate and process certificate requests using PKCS #7, [PKCS-7] and PKCS # 10 [PKCS-10].

2. All CAs **shall** support the generation of certificates and CRLs that conform to [RFC 5280]. (Specific requirements with respect to certificate and CRL extensions are detailed below.)

3. CAs **shall** be capable of issuing multiple certificates to users, and for all said certificates, asserting the key usage extension, including the Extended Key Usage extension and the "anyExtendedKeyUsage" value.

4. CAs **shall** be capable of including the CRL distribution points extension. At a minimum, CAs **shall** support the inclusion of LDAP and HTTP URLs to specify the location of CRLs. CAs **shall** be capable of specifying an authoritative OCSP responder in the Authority Information Access extension.

5. Each PKI has its own certificate profile, identifying certificate extensions that appear in the certificates and CRLs it issues.[10] CAs **shall** be able to generate all mandatory

[10] This profile is often documented explicitly, but may be implicitly specified through the certificate policy.

extensions in the appropriate profiles. For CAs owned or operated on behalf of Federal agencies, the following specific guidance applies:

a) CAs that implement Federal agency-specific policies **shall** be able to generate certificates and CRLs that meet the agency profile and the Federal PKI Certificate Profile [FPKI PROF].
b) CAs that implement the Common Policy Framework [COMMON] **shall** be able to generate certificates and CRLs meeting the Shared Services Certificate and CRL Profile [COMMON PROF].

6. CAs **should** support the inclusion of "private" extensions in certificates and CRLs.[11]

7. CAs **shall** support at least one of the following algorithms for digitally signing certificates and CRLs: RSA with PKCS#1 v1.5 padding; RSA with PSS Padding [RFC 3447], or ECDSA. To maximize flexibility, CAs **should** support all of the above algorithms.[12]

8. CAs **shall** include backup and archive capabilities to support reconstitution of the CA in the event of a disaster (e.g., fire, earthquake).[13] CAs **should** include backup and archive capabilities in order to establish when certificates were issued and revoked, and under whose authority.

9. CA/RA components **shall** be shipped or delivered via controlled methods that provide a continuous chain of accountability, from the purchase location to the CA's or RA's physical location.

2.3.2 OCSP Responders:

1. OCSP responders **shall** conform to [RFC 2560], Online Certificate Status Protocol.

2. OCSP responders **shall** be capable of processing both signed and unsigned requests and **shall** be capable of processing requests that either include or omit the name of the relying party making the request. However, OCSP responders may ignore signatures and requester names, if present.

3. OCSP responders **shall** be capable of processing certificate status requests and generating responses for non-error conditions as specified in [RFC 5019].

[11] Private extensions are defined by an organization to meet their own unique requirements. Note that noncritical private extensions do not impact the interoperability of certificates or CRLs.

[12] The algorithm used to sign certificates and CRLs in an operational CA is dependent upon both the cryptographic module in use and the CA's software. The selected algorithm must appear in both sets of supported algorithms.

[13] In cases where the root key has been compromised, destroyed or lost, it is necessary to rebuild the CA using a backup root key, rather than simply recover the lost state of the CA.

4. Where supported, the OCSP responder **should** sign the OCSP response with the algorithm used to sign the certificate. OCSP responders **shall** support at least one of the following algorithms for digitally signing response messages: RSA with PKCS#1 v1.5 padding; RSA with PSS Padding, or ECDSA. The supported algorithms **should** include the algorithm(s) used by the corresponding CA when signing the certificate whose status is in question. To support future algorithm transitions by the CA, OCSP responders **should** support all of the aforementioned algorithms.[14]

2.3.3 Cryptographic Modules

1. Cryptographic modules for CAs, Key Recovery Servers, and OCSP responders **shall** be hardware modules validated as meeting FIPS 140-2 Level 3 or higher.

2. Cryptographic modules for RAs **should** be hardware cryptographic modules validated as meeting FIPS 140-2 Level 2 or higher.

3. Relying party and user cryptographic modules **shall** be validated as meeting FIPS 140-2 Level 1 or higher.

2.3.4 Key Recovery Servers

1. If the PKI supports key establishment (i.e., certificates will include key transport or key agreement keys), the PKI **should** include a key recovery mechanism.

2. Implementations **should** support automated, user-initiated key recovery; key recovery by the organization **should** also be supported.[15]

2.3.5 Relying Party Software

1. Relying party path validation
   a) Relying party implementations **shall** implement RFC 5280-conformant path validation [RFC 5280].
   b) Where interoperability outside a single organization (e.g., a single Federal agency) is required, path validation modules **should** conform to requirements for an Enterprise PVM, as specified in NIST Recommendation for X.509 Path Validation.[16]

[14] As with CAs, the algorithm used to sign responses in an operational OCSP responder is dependent upon both the cryptographic module in use and the OCSP responder's software. The selected algorithm must appear in both sets of supported algorithms.

[15] Organizational key recovery should emphasize security and privacy, rather than performance. Dual control for recovery of a user's keys by the organization is strongly recommended.

[16] This document is available in draft form and can be found at http://csrc.nist.gov/groups/ST/crypto_apps_infra/documents/NIST_Recommendation_for_X509_PVMs.pdf.

c) Where interoperability across organizations is required, path validation modules **should** conform to requirements for a Bridge-Enabled PVM, as specified in NIST Recommendation for X.509 path validation.
d) Relying party implementations **shall** support CRLs for certificate status, and **should** support OCSP for certificate status.

2. Building certificate paths
   a) Relying party implementations **shall** be able to build certification paths.
   b) At a minimum, relying party implementations **should** be able to obtain CA certificates and CRLs using LDAP from an organizationally-designated local directory, as well as locations specified within a user certificate.
   c) Implementations **should** support http-based certificate retrieval.

3. Relying parties that work within a single organizational PKI (e.g., a PKI that supports a company or agency) **should** be able to discover paths for user certificates issued by CAs that are hierarchically subordinate to the trust anchor CA.

4. Relying parties that accept certificates from other organizations **should** be able to discover paths in non-hierarchical PKIs.

2.3.6 Client Software

1. Client implementations **shall** support multiple private keys and certificates per end user in order to support different cryptographic services. For example, the client implementation **should** support and differentiate between private keys associated with public keys in certificates supporting digital signatures, and private keys associated with public keys in certificates supporting key establishment.

2. Client cryptographic modules **shall** be validated at FIPS 140-2 Level 1 or higher.

3. Client implementations **should** support the certificate management protocol supported by the organization's CA.[17]

2.4 Recommendations for System Installers/Administrators

The system installer and administrator is the person (or people) who are responsible for establishing the PKI and who are responsible for the tasks associated with its day-to-day operation**.** The system administrator **shall** ensure that end users are trained and that the organization's security policy is enforced.

[17] Where keys and certificates are stored on smart cards, and all updates are performed at the RA, user implementations need not support the certificate management protocol.

2.4.1 Certificate Issuance

1. CAs **shall** be configured to ensure that certificates specify public keys with approved key sizes, valid domain parameters, and approved algorithms.

2. For maximum interoperability, CAs and users **should** use RSA key pairs for digital signatures and key transport.

3. For maximum security and performance, CAs and users **should** use elliptic curve key pairs for digital signatures and key agreement.

4. When signing certificates or CRLs, CAs **shall** generate digital signatures as specified in Table 2-2. When signing certificates or CRLs with RSA, digital signatures **should** be generated using the SHA-1 hash algorithm until 12/31/2010 to maximize interoperability.

5. CAs **shall** sign signature certificates with a key whose security strength is equal to or greater than the security strength of the key in the certificate. For key establishment keys, the signature may be generated using a weaker key in the certificate than will be used for key establishment.

6. Generating key pairs:
   a) Users **should** generate their own digital signature key pairs.
   b) The user or the PKI may generate key pairs for key establishment on the user's behalf; where required, the PKI may retain copies of the key establishment private key to permit key recovery.
   c) CAs **should** perform proof of possession on all key pairs before issuing certificates.

7. CAs **shall obtain** assurance of public key validity before issuing certificates.

8. Key usage extension.
   a) All certificates issued **shall** include the key usage extension.
   b) The key usage extension **should** restrict acceptance of the private key to a single cryptographic function: digital signatures, user authentication, or key establishment.
   c) Dual-use certificates (where a single key is used for both digital signatures and key establishment) **should not** be issued to users.
   d) Dual-use certificates may be issued to devices where required to support legacy applications.

9. Extended Key Usage extension.
   a) Certificates may include the extended key usage extension to support specific applications (e.g., smart card logon).
   b) If a certificate is intended for general use, in addition to some list of specific applications, the extended key usage extension **shall** also specify "Any Key Usage".

10. All certificates **shall** include the CRL distribution points extension to support the retrieval of status information.

11. If an OCSP responder is supported, a certificate **shall** include an appropriate URL in the Authority Information Access extension.

2.4.2 Certificate Revocation Requests

1. CAs **should** be configured to automate revocation processing where practical:
   a) CAs **should** be configured to authenticate and process revocation requests electronically.
   b) Where the CA can authenticate a digitally signed request submitted by the user of the associated key pair or an RA, the request **should** be handled without manual intervention.

2. RAs **should** be configured to submit digitally signed revocation requests on behalf of users or the organization.

2.4.3 Certificate Revocation List Generation

1. To maximize interoperability, all CAs **should** be configured to generate full CRLs. A full CRL is a single CRL that lists all revoked and unexpired certificates issued by this CA.

2. CAs that serve a large community **should** generate CRL distribution points in addition to full CRLs. Each CRL distribution point lists a subset of the revoked certificates for a given CA. The number of certificates covered by a CRL distribution point **should** be limited to a maximum of 250,000 to ensure that the distribution point CRLs do not grow to an unmanageable size.

2.4.4 PKI Repositories for the Distribution of Certificates and CRLs

1. PKIs **should** be configured to provide certificates and CRLs to requesters without authentication.

2. PKI repositories **shall** be configured to require authenticated access to modify the set of certificates and CRLs distributed by the repository.

3. At a minimum, repositories **shall** support either the HTTP version 1.1 or LDAP version 3 interface.

4. For maximum interoperability, both HTTP and LDAP **should** be supported.

5. Replication of repositories (e.g., through directory shadowing or web server replication) to maximize availability **should** be considered.

6. PKI repositories **should** contain all CA certificates issued by or to the corresponding PKI.

7. PKI repositories **shall** contain all current CRLs.

2.4.5 OCSP Responders

For Federal agencies, detailed configuration guidance for OCSP responders is specified in *Draft Guidance for OCSP Responders in the U.S. Federal PKI.*[18]

1. If maximum interoperability is required then:
   a) OCSP responders **shall not** require that requests be signed and **shall not** limit the set of relying parties to which certificate status information is provided.
   b) The responders **shall** generate OCSP basic responses, and the responses **shall not** include critical extensions.

2. Where interoperability requirements are limited to a closed community:
   a) OCSP responders may require signed requests, and may reject requests from entities outside that community.
   b) OCSP response messages may include private extensions known within the target community.

2.4.6 Backup and Archive

1. To maintain the availability of status information, CAs **shall** ensure that sufficient information is stored in a secure location to reconstitute the CA after a disaster.

2. CAs **should** archive sufficient information to establish when certificates were issued, and under whose authority.

3. As a general rule, all audit logs **should** be maintained, along with any certificates and CRLs issued by the CA.

4. User public signature verification keys **should** be archived, along with their corresponding certificates, for at least 7 ½ years after certificate expiration.

2.4.7 Relying Party Integration and Configuration

1. Path discovery components **shall** be configured to enable path discovery, and require the retrieval of status information.

2. Status information **should** be accepted in both CRL and OCSP formats.

[18] Draft guidance is available at http://cio.nist.gov/esd/emaildir/lists/pkits/doc00000.doc.

3. Relying party implementations **shall** be configured to recognize the smallest set of acceptable trust anchors possible.

4. For business-to-government and government-to-government applications, Federal agencies **should** use either the Common Policy Root CA or an agency CA that is cross certified with the Federal Bridge as the trust anchor.

5. For citizen-to-government applications with limited security requirements (e.g. Level 2 e-Authentication requirements as specified in [OMB 04-04]) and high interoperability requirements, agency applications may use the pre-installed trust anchors provided in COTS products.

6. Path validation modules:
   a) For end user applications and applications with minimal security requirements, path validation modules **should** be configured to accept any valid path.
   b) For systems with more significant security requirements (e.g., systems using PKI to satisfy Level 3 or Level 4 e-Authentication), path validation modules **should** be configured to only accept paths that are valid under appropriate policies.

2.5 User Guidance (Subscribers)

In a PKI, the subject is the identity of the user associated with a public key. The subject may be a person or a device. For the purposes of this section, the term user is either the person associated with a public key, or the administrator of a device associated with a public key.

1. Users **should** generate key pairs for digital signature and authentication.

2. Users may generate their own key pairs for key establishment, or the key establishment key pairs may be imported from a trusted source.

3. Users **shall** protect the authenticators (e.g., the PIN or password) that control access to their private keys.

4. Users **shall** request the revocation of their certificates if they believe the authenticator or cryptographic module has been stolen, copied or compromised.

5. Users **shall** control the disposition of "old" key pairs after certificates expire unless otherwise controlled in accordance with Federal agency policy and procedures
   a) Private signature keys **should** be destroyed after the corresponding certificate(s) expire.

b) Private key establishment keys need not be destroyed after the corresponding certificate(s) expire. The user **should not** destroy the private key establishment key until all symmetric keys established using this key have been recovered or otherwise protected (e.g., by encrypting under a different key). Premature destruction of private key establishment keys may prevent recovery of the subscriber's plaintext data.

3 IPsec

3.1 Description

IPsec is a suite of protocols for securing Internet communications at the network layer and operates within the Internet Protocol (IP). It is frequently used to establish Virtual Private Networks (VPNs), requiring both parties to share keying material, and enabling telecommuters or travelers to gain secure access to their business networks. IPsec provides the cryptographic security functions for both versions 4 and 6 of the Internet Protocol.

IPsec operates by inserting one of two special IPsec headers after the IP header in each message. The Authentication Header (AH) provides integrity protection. The Encapsulating Security Protocol (ESP) Header provides confidentiality and/or integrity protection. Both ESP and AH provide data origin authentication, and optionally provide replay protection. AH protects the IP header and the data following the IP header. ESP, when applied directly to a packet (i.e., in transport mode), protects the data, but not the IP header. However, ESP in tunnel mode (with a new IP header inserted) does protect the original IP header. Furthermore, using ESP with automated keying protects the source and destination addresses in the IP header, in either transport or tunnel mode. Since AH processing introduces unnecessary complexity, and since ESP can provide equivalent functionality, the use of AH is not recommended.

There have been three versions of IPsec.[19] All new systems **should** implement IPsec-v3, as it has many enhancements not found in the previous versions.[20] However, IPsec-v2 is still implemented in numerous current systems, despite the fact that it is obsolete.[21]

Two classes of key management methods are specified for IPsec: manual keying and automated keying. Manual keying involves an agreement (in an unspecified manner) by the parties in a communication on the IPsec protections to be applied and the symmetric keys to be used. This has a major downside in that it severely limits the scalability of the security solution and requires re-keying to be done in an unspecified manner. A Security Association (SA, i.e., a relationship between two or more entities that describes how each entity will use the security services to communicate securely) and its secret keys cannot be easily renewed in the cases where the SA expires, has been used for the maximum allowable volume of traffic, or if its keys are compromised.

To use automated keying, a negotiation between peers prior to exchanging IPsec-protected traffic determines the IPsec protections to be applied and the symmetric keys to be used. The same method can be used to maintain, delete, or renegotiate the SA (e.g., to rekey). This approach permits a decoupling of the key management mechanism from the

[19] There are no generally accepted names for *IPsec-v3* and *IPsec-v2*; these terms are used in this document to make the requirements more understandable

[20] IPsec-v3 is specified in RFC 4301, RFC 4302, and RFC 4303.

[21] IPsec-v2 is specified in RFC 2401, RFC 2402 and RFC 2406.

other security mechanisms, thus facilitating the use of alternative key management methods without having to modify other security mechanisms.

The preferred automated keying method is IKE, the Internet Key Exchange protocol that was designed specifically for use with IPsec. IKE generates the necessary keying material for IPsec via an authenticated secure channel between the two IKE peers. There are two versions of IKE in use: IKEv1 [RFC 2407, RFC 2408, and RFC 2409]] and IKEv2 [RFC 4306 and RFC 4718]; both versions perform mutual authentication, and establish and maintain security associations. SAs will be valid for a specified period of time or volume of traffic. These two versions of IKE are not interoperable.

Table 3-1 provides the IETF reference materials for versions 2 and 3 of IPsec.

Table 3-1. Summary of References for IPsec

Version	Security Architecture	Privacy	Authentication	Automated Key Management
IPsec-v2	RFC 2401	RFC 2406	RFC 2402, RFC2406	RFC 2407, RFC 2408, RFC 2409
IPsec-v3	RFC 4301	RFC 4303	RFC 4302, RFC 4303	RFC 4306, RFC 4718

The IPsec security mechanisms are not tied to any specific cryptographic algorithms; in fact, many algorithms and modes have IETF Requests For Comment (RFCs) describing their use with IPsec. This, however, can result in a situation where there are so many choices for typical system administrators to make that it is difficult to achieve interoperability. To improve interoperability in IPsec-v3, a small number of named cipher suites that cover typical security policies, referred to as user interface (UI) suites, were specified [RFC 4308]. The use of these suites is optional. Implementers may allow the individual selection of security algorithms (i.e., rather than selecting a pre-specified suite of algorithms), but users must be aware that picking non-standard groupings of algorithms may result in limited interoperability. However, when IPsec is used in the context of a VPN, security policy can be centrally managed, thus ensuring interoperability without the use of pre-defined cipher suites. Complete IETF algorithm guidance is provided in [RFC 4835] (for AH and ESP), [RFC 4307] (for IKEv2) and [RFC 4109] (for IKEv1).

3.2 Security and Compliance Issues

3.2.1 Cryptographic Algorithms

The following table gives cryptographic algorithm recommendations for use within IPsec. The algorithms that are specified for IKE are used to protect IKE's own traffic. The ESP and AH algorithms are used to provide IPsec protection to data traffic; for these algorithms to be used within ESP or AH, IKE must be capable of negotiating their use.

Table 3-2. Cryptographic Algorithm Recommendations

Protocol	Cryptographic Service	Algorithm/Mode	IETF Requirement for Implementation	Approved for Federal Use
ESP	Encryption	TDEA in CBC mode [RFCs 2451, 4835]	Mandatory	Yes, with three distinct keys
ESP	Encryption	NULL [RFCs 2410, 4835]	Mandatory	Yes, for cases where integrity protection is required but not encryption
ESP	Encryption	AES with 128-bit keys in CBC mode [RFCs 3602, 4835]	Mandatory	Yes
ESP	Encryption	AES-128 in Counter mode [RFCs 3686, 4835]	Optional[22]	Yes but must be used with non-NULL integrity protection [RFC 3686]
ESP or AH	Integrity Protection	HMAC SHA1-96 [RFCs 2404, 4835]	Mandatory	Yes
ESP or AH	Integrity Protection	HMAC SHA-256-128 [RFC 4868]]	Optional	Yes

[22] The RFCs actually have 3 levels of requirements: Recommended (SHOULD), and Optional (MAY). Mandatory (MUST), This is actually IETF Recommended. In this table, IETF Recommended and IETF Optional are both listed as Optional in keeping with the use of the words Recommended and Optional throughout the rest of this document.

Protocol	Cryptographic Service	Algorithm/Mode	IETF Requirement for Implementation	Approved for Federal Use
ESP	Integrity Protection	NULL [RFC 4835]	Optional	Yes, but only when combined-mode algorithms are used to provide integrity-protection.
ESP	Encryption and Integrity Protection	AES-128 in Galois/Counter Mode [RFC 4106, RFC 4835]	Optional	Yes
ESP	Encryption and Integrity Protection	AES-128 in Counter mode with CBC-MAC [RFCs 4309, 4835]	Optional	Yes
ESP or AH	Integrity Protection	AES-128 in GMAC Mode [RFC 4543]	Optional	Yes
IKEv1 or IKEv2	Encryption	TDEA in CBC mode [RFCs 4109, 4307]	Mandatory	Yes, with three distinct keys
IKEv1 or IKEv2	Encryption	AES-128 in CBC mode [RFCs 4109, 4307]	Optional[22]	Yes
IKEv1 or IKEv2	Pseudo-random function	HMAC-SHA1 [RFCs 4109, 4307]	Mandatory	Yes
IKEv1 or IKEv2	Pseudo-random Function	HMAC-SHA-256 [RFC 4868]	Optional	Yes
IKEv1	Diffie-Hellman Group	1024-bit MODP [RFCs 4109, 4307]	Mandatory	Yes
IKEv1 or IKEv2	Diffie-Hellman Group	2048-bit MODP [RFCs 4109, 4307]	Optional[22]	Yes
IKEv2	Integrity	HMAC-SHA1-96 [RFC 4307]	Mandatory	Yes

Protocol	Cryptographic Service	Algorithm/Mode	IETF Requirement for Implementation	Approved for Federal Use
IKEv2	Integrity	HMAC-SHA256-128 [RFC 4868]	Optional	Yes
IKEv1	Peer Authentication	RSA[22] or DSA with SHA1 [RFCs 2409, 4109, 4306]	Optional	Yes
IKEv2	Peer Authentication	RSA with SHA1 [RFCs 2409, 4109, 4306]	Mandatory	Yes
IKEv2	Peer Authentication	DSA with SHA1 [RFCs 2409, 4109, 4306]	Optional	Yes
IKEv1 or IKEv2	Peer Authentication	RSA or DSA with SHA256 [RFCs 2409, 4109, 4306]	Optional	Yes

NULL integrity protection or NULL encryption means that no integrity protection is applied or no encryption is used, respectively. NULL integrity protection (often referred to as NULL authentication) is used for situations where confidentiality is required without the need for integrity protection. NULL integrity protection **shall not**, and in fact cannot, be used with NULL encryption. ESP, for example, could send unencrypted packets, (encryption set to NULL), but would be required to integrity protect them; for example by using HMAC-SHA1. On the other hand, ESP could send packets encrypted with AES-128 in CBC mode but omit the integrity check, (integrity protection set to NULL). ESP could not send packets both unencrypted and with no integrity protection! It is recommended that IPsec-protected traffic **should** always be integrity-protected, whether or not it is encrypted.

When IPsec-protected traffic is integrity-protected, an Integrity Check Value (ICV) is stored in the Integrity Check Value field of the ESP payload, [RFC 4303], or in the Integrity Check Value field of the Authentication Header (AH) [RFC 4302]. (This field is referred to as the “Authentication Data” field in IPsec V2 [RFCs 2406 and 2402]).

In the case of HMAC for integrity protection, the length of the ICV value is at most the size of the output value of the hash function. For IPsec applications, the ICV value **shall** be truncated to 96 and 128 bits for HMAC-SHA1 and HMAC-SHA256, respectively, [RFCs 4307, 4868].

Although the IETF is transitioning to AES-XCBC-MAC, [RFC3566 and 4434], and also allows the use of HMAC-MD5 (not shown in the table), neither is approved for use by the Federal government. As such, AES-XCBC-MAC and HMAC-MD5 **shall not** be used for integrity protection.

A new class of algorithms, called combined-mode algorithms, appears in the above table. They can be negotiated by IKE and used within IPsec-v3 cipher suites. These algorithms provide both encryption and integrity protection. Two combined-mode algorithms have been approved for Federal government use: AES in Galois/Counter Mode (GCM) [RFC 4106] and AES in counter mode with CBC-MAC (AES-CCM) [RFC 4309].

There is also a variant of AES-GCM, referred to as AES-GMAC [RFC 4543] , that provides integrity protection but does not provide encryption. This mode can be used within either the ESP or AH header.

The maximum size of the ICV for AES-CCM, AES-GCM and AES-GMAC is 16 bytes. Implementations **shall** support a size of 16 bytes for these three algorithms [RFCs 4309, 4106, 4543].

AES-GCM, AES-CCM, AES-CTR, and AES-GMAC **shall not** be used with manually distributed keys. If the counter value, in AES-CTR or AES-CCM, or the IV value, in AES-GCM or AES-GMAC, is used for more than one packet with the same key, the security of the algorithm's confidentiality mechanism is compromised. Since manual keying presents a major challenge to this limit, it **shall not** be used with these algorithms. Automated keying using IKE establishes secret keys for the two peers within each Security Association, with an extremely small probability of duplicate keys.

In previous IETF guidance, DES using the CBC mode [RFC 2405] was mandatory-to-implement; however, this algorithm **shall not** be used to protect information.

IPsec allows the individual selection of security algorithms. As an example, an implementer using Table 3-2 and following the guidance of Part 1 of SP 800-57, could select the following algorithms to form an IPsec suite with an overall security strength of 80 bits:

- ESP Encryption: AES in CBC mode
- ESP Integrity Protection: HMAC-SHA1
- IKEv2 Encryption: AES in CBC mode
- IKEv2 Pseudo-random function: HMAC-SHA1
- IKEv2 Diffie-Hellman group: 1024-bit MODP
- IKEv2 Integrity: HMAC-SHA1
- IKEv2 Peer Authentication: 1024-bit RSA with SHA-1

Note that if the implementer wanted to ensure a security strength of 112 bits, he would have to make the following changes to the suite above:

- IKEv2 Diffie-Hellman group: 2048-bit MODP
- IKEv2 Peer Authentication: 2048-bit RSA with SHA-256

The following two cipher suites, described in Tables 3-3 and 3-4, incorporate the algorithms from NSA’s Suite B. These algorithms are all approved for Federal government use. The Suite-B-GCM-128 and Suite-B-GCM-256 suites are both defined in [RFC 4869]. At present, these cipher suites are not widely available or deployed. However, wherever practical, implementations **should** be procured that support these cipher suites, and they **should** be selected for use wherever very high performance and security strength are required. Recall that AES-GCM is a combined-mode algorithm that provides both encryption and integrity protection, so these suites provide integrity, despite the fact the integrity mechanism is listed in [RFC 4869] as NULL for both suites.

Table 3-3: Suite-B-GCM-128 Cipher Suite

Suite-B-GCM-128	Mechanism	Guidance
ESP	Encryption and integrity protection	AES with 128-bit keys and 16-octet Integrity Check Value (ICV) in GCM mode [RFC 4106]
IKEv1 or IKEv2	Encryption	AES with 128-bit keys in CBC mode [RFC 3602]
IKEv1 or IKEv2	Pseudo-random function	HMAC-SHA-256 [RFC 4868]
IKEv1 or IKEv2	Diffie-Hellman Group	256-bit random ECP group [RFC 4753]
IKEv1 or IKEv2	Integrity	HMAC-SHA-256-128 [RFC 4868]
IKEv1 or IKEv2	Peer Authentication	ECDSA-256 [RFC 4754]

Table 3-4: Suite-B-GCM-256 Cipher Suite

Suite-B-GCM-256	Mechanism	Guidance
ESP	Encryption and integrity protection	AES with 256-bit keys and 16-octet Integrity Check Value (ICV) in GCM mode [RFC 4106]
IKEv1 or IKEv2	Encryption	AES with 256-bit keys in CBC mode [RFC 3602]
IKEv1 or IKEv2	Pseudo-random function	HMAC-SHA-384 [RFC 4868]
IKEv1 or IKEv2	Diffie-Hellman Group	384-bit random ECP group [RFC 4753]
IKEv1 or IKEv2	Integrity	HMAC-SHA-384-192 [RFC 4868]
IKEv1 or IKEv2	Peer Authentication	ECDSA-384 [RFC 4754]

IKEv1 implementations **shall** support the pre-shared key format for peer authentication for interoperability, as referenced in [RFC 2409]. The authentication method used with IKEv1 may use either a pre-shared key or ECDSA [RFC4869].

A pre-shared key is a (symmetric) secret key that is established before an IPsec protocol is executed. The pre-shared key **shall** provide a security strength that meets or exceeds the target security strength for the data to be protected. For example, if 112 bits of security is required, HMAC SHA1 is used for integrity (authentication) protection and AES-128 is used for encryption (confidentiality protection), the pre-shared key **shall** have at least 112 bits of security strength. In addition, when encryption is used, the pre-shared key **should** have a security strength that is equal to the length of the encryption key. In this example, the pre-shared key **should** have 128 bits of security.

A pre-shared key **shall** be generated by one of the approved random bit generators as specified in [SP 800-90] that meets or exceeds the required security strength, and is then securely distributed (out-of-band) to the other communicating party in an IPsec connection.

[RFC 4869] also defines two additional suites: Suite-B-GMAC-128 and Suite-B-GMAC-256. These suites substitute the GMAC mode for GCM; thus, they provide ESP integrity protection without ESP encryption. These two suites are also approved for Federal use.

3.2.2 Additional Recommendations

1. NULL encryption and NULL integrity protection **shall not** both be used at the same time within ESP. NULL integrity protection in ESP **shall not** be used with AES-128 in Counter Mode.

2. The Authentication Header (AH) **should not** be used in IPsec version 3.

3. IKE **should** be used for automated key management to ensure a re-keying capability and scalability.

4. Once an ESP Security Association has expired or is no longer in use, its ESP encryption keys **shall** continue to be protected by the system and kept secret as long as the data they were used to protect needs to be kept secret.

3.3 Procurement Guidance

These recommendations are written to assist individuals responsible for selecting security products that include IPsec for the security of the IP layer.

1. Any IPsec system for use within the Federal government **should** include an IKE implementation for automated key management.

2. IPsec implementations **shall** include approved algorithms for each IPsec security component. In addition, the implementation **should** be capable of using one or more of the approved cipher suites specified in Section 3.2. Being capable of using one or more of these cipher suites will improve interoperability.

3. IPsec implementations **should** include the algorithms used in the Suite B cipher suites.

3.4 Recommendations for System Installers

Systems installers are those individuals that install products that include IPsec for security.

1. IKE **should** be used for automated key management within any IPsec system.

2. NULL encryption **shall** only be employed when integrity protection is required, but confidentiality is not needed.

3. Installers **should** select approved algorithms for each security component, as specified in Section 3.2.

3.5 Recommendations for System Administrators

System administrators are those individuals responsible for the day-to-day functioning of the security product containing IPsec. System administrators **shall**:

1. Ensure that end users are properly trained and that the organization's security policy is enforced.

2. Ensure that a key used by the product is protected throughout its lifespan.

3.6 Recommendations for End Users

An end user is the individual using a product that relies on IPsec for security. End users **shall**:

1. Be aware of and trained to follow the organization's security policy for using the product.

2. Operate their system as instructed by their organization and system administrator.

4 Transport Layer Security (TLS) and Secure Socket Layer (SSL)

4.1 Description

The Transport Layer Security (TLS) and Secure Socket Layer (SSL) protocols are the primary end-to-end security protocols used to protect information on the Internet. TLS is an enhanced version of SSL; the protocols are similar, but not identical. The Internet Engineering Task Force (IETF) governs the TLS protocol [RFC 5246]. 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 a robust protocol that is used to protect various links (e.g. authentication server to a wireless access point or the e-mail link between client and server). Specific guidance could vary greatly, based on the intended application. The current discussion has, therefore, primarily been limited to the classic scenario where a browser is acting as a client for a human user, and is interfacing with a web site. Some of the other applications of TLS (for example, dedicated network infrastructure applications primarily involving machines with no human user) may not require all of the features of general purpose web browsers and servers. Moreover, some cipher suites, as indicated below, are primarily intended for these infrastructure or more restricted applications.

A TLS session occurs between a client (typically an end user's web browser) and a server (typically a web server belonging to an entity with which the end user wants to conduct business or needs to exchange information). Symmetric keys for a session are determined during a protocol involving either Diffie-Hellman (DH) key agreement or RSA key transport. The latter method implicitly authenticates the server to the client. In a DH key agreement, the server authenticates itself by signing an ephemeral DH key and sending a certificate with its public signing key. Thus, the server will always send a certificate, with either a signing key or a key-agreement key. The server may request a certificate from the client. Except for non-Suite B static DH key-agreement, client certificates will always contain a signing key. An "anonymous" exchange, i.e., without server authentication, **shall not** be used by implementations conforming to this Recommendation. Static-static DH key establishment **should not** be used by implementations conforming to this Recommendation.

A TLS session begins with a cryptographic negotiation between the client and the server to select a suite of algorithms to be used in the session (often called a cipher suite) and to establish a set of cryptographic keys to be used for a variety of functions throughout the session. This negotiation takes the form of a series of "handshake" messages between the client and server. The client states the "cipher suites" that it can handle in the order that it prefers them and provides a non-secret random nonce. A cipher suite bundles the choice of key-establishment method (e.g., RSA key transport or DH key agreement method), symmetric key algorithm and data-integrity hash function to be used into a single value. The server then selects the "cipher suite" to be used from the list provided by the client, and provides its own random nonce and a certificate containing a signing or key-establishment key. The server may also request certificate-based client authentication during the handshake.[23] Session keys are then established in accordance with the selected cipher suite, with the random numbers providing assurance of liveliness and freshness to the derived keys.

When only the server supplies a certificate, the TLS protocol provides for integrity and authentication services to a client whose identity has not necessarily been verified. This is sufficient under many circumstances, such as when TLS is used to protect passwords or credit card numbers during transactions over the Internet. For example, if an end user is trying to purchase something from Acme Flowers online, he wants to be certain that he is doing business with Acme Flowers (and not Acme Flours, the credit card thief), but the Acme Flowers server may only be concerned that the credit card number supplied is valid. In this case authentication of the end user's identity (and his authority to use the credit card supplied) could be handled outside of the TLS protocol (e.g., by contacting the credit card company to verify the validity of the credit card number).

After completion of the handshake sequence, TLS provides a secure communications channel between the server and client for the duration of a communications session. All cipher suites provide authentication and integrity protection for transferred data, and most TLS cipher suites also provide encryption. All TLS cipher suites recommended in this document provide authentication and integrity. Most also provide encryption. If encryption is provided, data is encrypted when sent, and decrypted when received. TLS does not, however, provide a cryptographic non-repudiation service to allow a validation of the session data or authentication after the communications session has ended, e.g., by a third party.

4.2 Security and Compliance Issues

4.2.1 General

1. Servers make the selection of the cipher suite to be used, based on the choices offered by the client during the handshake protocol. Therefore, if a client offers any cipher suite not listed in the tables below, the server might select it, even when the client indicates (by the order of its list of suites) that its preference is for another cipher suite.

2. Symmetric key sizes **shall** be compliant with guidance in Part 1, and public key sizes **shall** be consistent with guidance in the PKI section above.[24] Clients **shall not** accept any certificate signed using a key smaller than the approved key size

[23] In the initial Client Hello Message the client proposes a list of all the TLS versions and cipher suites that it can use, and, in the Server Hello reply, the server selects a version and cipher suite from the choices offered by the client. TLS accommodates several versions (1.0, 1.1 and 1.2) as well as a bewildering number of cipher suites. While the ordering of the cipher suites in the Client Hello message is supposed to indicate the order of the client's preference, experimentation reveals that few servers honor this; most simply follow their own preference, if it is anywhere on the client's list.

[24] See Part 1 Sections 5.6.1 and 5.6.2.

for the date of certificate verification by the client. This is generally accomplished through control of the root certificate key-store.[25]

3. Hash functions used during the signing of TLS certificates **shall** be consistent with the public key sizes, as given in Section 5.6.2 of Part 1.

4. In choosing algorithm suites, Federal agencies need to consider the value or sensitivity of the information being protected and how long it must be protected (see Part 1, Section 6). Public keys contained in certificates (i.e., subject public keys) that are used only for authentication need only be large enough that they are secure at the time they are used in a TLS session. However, while TLS does not create encrypted files that are saved, an eavesdropper could record a TLS session and attack it at a later time when more powerful technology is available. This **should** be taken into consideration when determining the security strength needed for the session.

TLS cipher suites have the form:

TLS_key_establishment_alg_WITH_encryption alg_message authentication_alg.[26]

For example,

TLS_RSA_WITH_AES_128_CBC_SHA.

The "TLS" distinguishes these cipher suites from SSL protocol suites. In the example, the key establishment method is the RSA public key algorithm. For this example, the server provides a public key certificate containing an RSA modulus and encryption exponent and the AES block cipher algorithm is used with 128-bit keys in cipher block chaining (CBC) mode to encrypt data and HMAC_SHA1 to provide integrity protection. Acceptable combinations of algorithms are referenced in the handshake messages with a two-byte indicator assigned by the Internet Assigned Numbers Authority (IANA).

Not all cryptographic algorithms available for use are appropriate for the protection of Federal government information. Federal servers and clients **shall** use approved algorithms for transmitting information. Federal clients **should** support the use of un-approved algorithms in receiving information in accordance with organizational policy, in order to support interoperability. Not all combinations of algorithms are appropriate for the protection of Federal government information; most options that use AES or 3-key TDEA (also called 3DES in TLS cipher suites) are acceptable. The IETF identifies one TLS cipher suite as mandatory-to-implement:

TLS_RSA_WITH_AES_128_CBC_SHA

[25] See Section 2 above

[26] Beginning with TLS 1.2, the "message_authentication_alg" is also used in the PRF (pseudo-random function) for deriving keying material.

As described in [RFC4346], all implementations are required to include this particular cipher suite. All TLS cipher suites begin with “TLS” (this distinguishes them from the original SSL protocol that was the ancestor of TLS). “TLS” is followed by a key establishment method; in the cipher suite example given above, this is the RSA public key algorithm. For this method, the server is required to have a server public key key establishment certificate using the RSA algorithm. “WITH_AES_128_CBC_SHA” specifies the cipher mechanism actually used to transfer “payload data.” In this example, the AES block cipher is used with 128-bit keys in cipher block chaining mode, and HMAC-SHA1 provides integrity protection.

4.2.2 Recommended Cipher Suites for Federal Government Use

A large number of cipher suites have been defined for TLS. Many are not suitable for Federal government use. The cipher suites listed in the following tables may be used to protect Federal government information. Anonymous cipher suites that do not authenticate the server, or that use HMAC-MD5 or unapproved encryption algorithms **shall not** be used to protect Federal government information and are not included in the tables below. Four tables are provided: Conventional Public Key cipher suites, Elliptic Curve cipher suites, Pre-shared Key cipher suites and Elliptic Curve-AES suites that are specific to TLS version 1.2.

Conventional public key cipher suites with HMAC-SHA1 are listed in Table 4-1. All suites listed in the table are allowed for use in protecting Federal information. The cipher suites without the [256] suffix all use HMAC-SHA1 to authenticate and provide integrity protection for payload data, and use the RSA, Diffie-Hellman and DSA public key algorithms for key establishment and server authentication. These cipher suites may be used with TLS Versions 1.0, 1.1 or 1.2. The [256] suffix means that similar cipher suites have also been defined, as of TLS 1.2, which use HMAC-SHA256 for integrity protection and for key derivation, replacing SSL’s original MD5/SHA1 pseudorandom function (PRF).

Table 4-1: Conventional Public Key Cipher Suites Using HMAC-SHA1 and HMAC-SHA256

SUITE	Key Exchange	Encryption
TLS_RSA_WITH_NULL_SHA	RSA	NULL
TLS_RSA_WITH_3DES_EDE_CBC_SHA *	RSA	3DES_EDE_CBC
TLS_RSA_WITH_AES_128_CBC_SHA [256] **	RSA	AES_128_CBC
TLS_RSA_WITH_AES_256_CBC_SHA [256]	RSA	AES_256_CBC
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH_DSS	3DES_EDE_CBC
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH_RSA	3DES_EDE_CBC
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA ***	DHE_DSS	3DES_EDE_CBC
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE_RSA	3DES_EDE_CBC
TLS_DH_DSS_WITH_AES_128_CBC_SHA [256]	DH_DSS	AES_128_CBC
TLS_DH_RSA_WITH_AES_128_CBC_SHA [256]	DH_RSA	AES_128_CBC
TLS_DHE_DSS_WITH_AES_128_CBC_SHA [256]	DHE_DSS	AES_128_CBC
TLS_DHE_RSA_WITH_AES_128_CBC_SHA [256]	DHE_RSA	AES_128_CBC
TLS_DH_DSS_WITH_AES_256_CBC_SHA [256]	DH_DSS	AES_256_CBC
TLS_DH_RSA_WITH_AES_256_CBC_SHA [256]	DH_RSA	AES_256_CBC
TLS_DHE_DSS_WITH_AES_256_CBC_SHA [256]	DHE_DSS	AES_256_CBC
TLS_DHE_RSA_WITH_AES_256_CBC_SHA [256]	DHE_RSA	AES_256_CBC

* This cipher suite is named by IETF as mandatory-to-implement for TLS version 1.1 [RFC 5246].
** This cipher suite is named by IETF as mandatory-to-implement for TLS version 1.2
*** This cipher suite is named by IETF as mandatory-to-implement for TLS version 1.0 [RFC 2246].

Elliptic curve cipher suites using HMAC-SHA1 are listed in Table 4-2. The cipher suites without [256] or [384] suffixes all use HMAC-SHA1 for payload data integrity and use Elliptic Curve algorithms for key agreement. They may use either the Elliptic Curve Digital Signature Algorithm (ECDSA) or RSA for server authentication. These cipher suites may be used with TLS Versions 1.0, 1.1 or 1.2. The [256]/[384] suffix means that similar cipher suites have also been defined, as of TLS 1.2, which use HMAC-SHA256 or HMAC-SHA384 for integrity protection and for key derivation, replacing SSL’s original MD5/SHA1 PRF. ECDHE denotes the use of a key exchange algorithm in which each side (client and server) each generate an elliptic curve Diffie-Hellman ephemeral key pair for use in that particular TLS session.

Table 4-2: Elliptic Curve Cipher Suites Using HMAC-SHA1/HMAC-SHA256/HMAC-SHA384

SUITE	Key Exchange	Encryption
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	ECDH_ECDSA	3DES_EDE_CBC
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA[256]	ECDH_ECDSA	AES_128_CBC
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA[384]	ECDH_ECDSA	AES_256_CBC
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	ECDHE_ECDSA	3DES_EDE_CBC
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA[256]	ECDHE_ECDSA	AES_128_CBC
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA[384]	ECDHE_ECDSA	AES_256_CBC
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA	ECDH_RSA	3DES_EDE_CBC
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA[256]	ECDH_RSA	AES_128_CBC
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA[384]	ECDH_RSA	AES_256_CBC
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	ECDHE_RSA	3DES_EDE_CBC
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA[256]	ECDHE_RSA	AES_128_CBC
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA[384]	ECDHE_RSA	AES_256_CBC

Pre-shared key (PSK) cipher suites using HMAC-SHA1 are listed in Table 4-3; pre-shared keys **shall** be distributed in a secure manner, such as secure manual distribution or using a digital signature or key establishment certificate. These cipher suites all use HMAC-SHA1 to authenticate and provide integrity protection for payload data. They employ a pre-shared key for entity authentication (for both the server and client) and may also use RSA or ephemeral Diffie-Hellman (DHE) algorithms for key establishment. For example, when DHE is used, the result of the Diffie Hellman computation is combined with the pre-shared key and other input to determine the pre-master secret.

The pre-shared key **shall** have a minimum security strength of 112-bits. Because these cipher suites require pre-shared keys, these suites are not generally applicable to classic secure web site applications and are not expected to be widely supported in web browser clients or general web servers. NIST suggests that these suites be considered in particular for infrastructure applications, particularly if frequent authentication of the network entities is required. These cipher suites may be used with TLS Versions 1.0, 1.1 or 1.2.

Table 4-3: Pre-shared Key Cipher Suites Using HMAC-SHA1

SUITE	Key Exchange	Encryption
TLS_PSK_WITH_3DES_EDE_CBC_SHA	PSK	3DES_EDE_CBC
TLS_PSK_WITH_AES_128_CBC_SHA	PSK	AES_128_CBC
TLS_PSK_WITH_AES_256_CBC_SHA	PSK	AES_256_CBC
TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA	DHE_PSK	3DES_EDE_CBC
TLS_DHE_PSK_WITH_AES_128_CBC_SHA	DHE_PSK	AES_128_CBC
TLS_DHE_PSK_WITH_AES_256_CBC_SHA	DHE_PSK	AES_256_CBC
TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA	RSA_PSK	3DES_EDE_CBC
TLS_RSA_PSK_WITH_AES_128_CBC_SHA	RSA_PSK	AES_128_CBC
TLS_RSA_PSK_WITH_AES_256_CBC_SHA	RSA_PSK	AES_256_CBC

Four new cipher suites are introduced with TLS 1.2 using only Suite B algorithms. These are listed in Table 4-4. These cipher suites are a departure from earlier TLS cipher suites in several respects, and only work with TLS version 1.2, which allows the negotiation of the Pseudo-Random Function (PRF) used in the process of deriving the symmetric keys (used to protect payload data) from the "pre-master secret" arrived at by the key agreement operation. Earlier TLS versions (1.1 and 1.0) use the original SSL PRF, a combination of MD5 and SHA-1 to derive keying material. (The hash function used for data integrity was separate and negotiated as part of the cipher suite). For these cipher suites, ECDHE is used for key establishment. Moreover, TLS 1.2 adds authenticated encryption as a symmetric cipher type. The AES GCM cipher suites offer the potential for higher performance than the AES CBC suites. The CBC modes offer compatibility with those implementations still using AES-CBC. Elliptic curve key establishment offers performance advantages over the RSA alternatives and at higher security strengths. These cipher suites are too new to yet be widely available or deployed, but NIST suggests that, wherever practical, servers and clients be procured that support these cipher suites, and that they be selected for use wherever very high performance and security strength are required.

Table 4-4: TLS 1.2-Suite B Cipher Suites

Suite	MAC	PRF
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	Galois Ctr.	P_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	Galois Ctr.	P_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	HMAC-SHA256	P_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	HMAC-SHA384	P_SHA384

4.3 Procurement Guidance

The following recommendations are for any individual that makes a purchasing decision for acquiring a TLS/SSL component. Recommendations for purchasing are:

General requirements:

1. New procurements of clients and servers **shall** support TLS (SSL 3.0 or higher).
2. Implementations **shall** support approved algorithms (see Section 4.2 above).
3. Implementations **shall** support client authentication using public key certificates.
4. Implementations **shall** use an approved random bit generator [SP 800-90].

Server implementations:

1. Web server implementations **should** support the following cipher suites:
   TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
   TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
   TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
   TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
2. Implementations **shall** protect the server's private key from unauthorized disclosure.
3. Implementations **should** select cipher suites in the order of preference submitted by the client.

Client implementations:

1. Federal client implementations **should** support at least one of the following four cipher suites:
   TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
   TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
   TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
   TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
2. Client implementations **should** allow system administrators to provide the list of cipher suites to be provided to the server in order of preference.
3. Client implementations **shall** support the recognition and verification of the certificate path for the server's certificate before trusting the server's certificate (see Section 2 above).
4. Client implementations **shall** be capable of verifying that the identifier presented in the server's certificate matches the identifier displayed by the browser.
5. Client implementations **should** give the end user an opportunity to verify that the site named in the certificate and displayed by the browser is consistent with the site that the end user was attempting to reach.
6. Client implementations for human users **shall** provide an indication to the user (e.g, by generating a message) if the site named in the certificate does not match the URL[27] that the end user requested.

[27] Uniform Resource Locator; the unique address for a file (e.g., a web page) that is accessible on the Internet.

7. Client implementations **shall** be capable of supporting a well-managed root certificate key-store (see Section 2 above.)

4.4 Recommendations for System Installers

The system installer is the individual(s) that installs the TLS/SSL application and performs the initial configuration of the system.

System installers **should**:

1. Configure the root certificate store appropriately (See Section 2)[28].
2. Disable unapproved cipher suites, wherever possible.
3. Disable SSL 2.0 (most clients allow this).
4. Properly configure and install client certificates, in accordance with applicable policy.
5. Configure client implementations to provide a list of cipher suites to the server in order of preference if allowed by the application.
6. Configure server implementations to select the client's most preferred cipher suite from the cipher suite lists presented by the clients in their ClientHello messages.
7. Ensure that server certificates are not expired and that they chain to a CA root certificate that is normally in the root store of the intended user population.
8. The public keys in certificates **should not** be used for multiple purposes (e.g., digital signatures and key establishment). However, TLS applications use the RSA server keys for both key agreement and server authentication, and this is acceptable because it is a carefully analyzed key agreement scheme.
9. Systems **shall** be configured to use approved algorithms and key sizes.
10. Federal government servers **should** be configured to include the following four cipher suites:
    TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,
    TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
    TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA, and
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA.
11. Federal government clients **should** be configured to include at least one of the following four cipher suites:
    TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,
    TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
    TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA, and
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA.

[28] Note that some common desktop operating systems permit centralized management of root stores; however, many systems and browsers are shipped pre-configured with hundreds of root certificates pre-installed.

4.5 Recommendations for System Administrators

The System Administrator is the individual who runs the TLS/SSL application on a day-to-day basis and, on client implementations, interacts with the end user. This includes acquiring certificates for the server and assisting the user in acquiring certificates. The system administrator **shall** ensure that end users are properly trained and that the organization's security policy for using the product is enforced.

System administrators **should**:

1. Obtain and install an appropriate server certificate with an appropriate strength public key that can be verified using the SHA-1 or SHA-256 hash algorithm
2. Maintain the server such that it only accepts approved algorithms and cipher suites that use 3TDES or AES for encryption.
3. Configure the client system to provide the list of cipher suites to the server in the order of their preference with the most preferred cipher suite presented first and the least preferred cipher suite presented last.

If client authentication is to be used, system administrators **should:**

1. Maintain the root and intermediate certificate store for the intended client certificates,
2. Support policy processing and path discovery according to organizational policy if they are used, and
3. Support CRL or OCSP responder processing.

4.6 Recommendations for End Users

An end user is the individual using a client to access the system. End users **shall**:

1. Operate their client systems as instructed by their organization and system administrator.
2. Protect the private key associated with the public key contained in the user's certificate if it is contained outside the client (e.g., on a smart card), as advised by the system administrator.
3. Look for browser indicators that a secure session is in progress (e.g., Internet Explorer's closed padlock)
4. Verify that the site named in the certificate and displayed by the browser is consistent with the site that the end user is attempting to reach.

5 Secure/Multipart Internet Mail Extensions (S/MIME)

5.1 Description

Secure/Multipurpose Internet Mail Extensions (S/MIME) provides a consistent way to send and receive secure Internet mail. S/MIME is a set of specifications that are defined by a series of IETF RFCs, namely [RFC 3851], [RFC 3852], [RFC 2045], [RFC 2046], [RFC 2047], [RFC 2048], and [RFC 2049]. S/MIME provides the following cryptographic security services for electronic messaging applications:

- Authentication of a sending party using digital signatures,
- Message integrity and non-repudiation of origin using digital signatures, and
- Confidentiality using encryption.

S/MIME, therefore, requires a suite of algorithms for creating digital signatures, generating hash values, establishing keys and encrypting the content of the email, as well as some means of establishing and sharing digital identities. Federal implementations rely on a public key infrastructure, specifically X.509 PKI, to establish S/MIME user identities, to bind those identities to the user's public key through public key certificates, to provide digital signatures and to provide keys to be used for content encryption or to establish symmetric keys for use on a per message basis. For detailed information on PKIs, see [SP 800-32].

Stored electronic mail encompasses key management issues associated with encrypted file integrity and with transmission over a network. It is therefore necessary 1) to establish pair-wise and/or multicast (sent to more than one recipient) key management relationships between the sender and receiver(s) and 2) to securely store the key(s) associated with encrypted email until it is no longer necessary for a recipient to be able to decrypt or verify the integrity of the email.

S/MIME is not restricted to email; it can be used with any transport mechanism that employs MIME protocols, such as the Hypertext Transfer Protocol (HTTP).

5.2 Security and Compliance Issues

S/MIME products can be implemented with different combinations of security features and a variety of cryptographic algorithms. Senders and receivers may have different capabilities and may be sending messages protected with algorithms of different strengths. This can lead to numerous interoperability issues. Federal clients using secure email **shall** be able to perform the following:

- Send and receive signed messages,
- Send and receive encrypted messages,
- Send and receive signed and encrypted messages,
- Request, send and process signed receipts, and

- Process messages from secure email list clients (includes suppressing receipts, as required, and nondisclosure of list recipients, as required).

Furthermore, Federal systems **shall**:

- Utilize cryptographic modules that are FIPS 140-1 or FIPS 140-2 certified,
- Support cryptographic suites ALS 1 and ALS 2 (see Tables 5-1 and 5-2 below), and
- Require X.509 certificates that conform to Federal PKI X.509 Certificates and the CRL Extensions Profile.

Federal clients **should** be capable of sending and processing email with security labels and securely binding senders' certificates to their signatures through the signing certificate attribute as described in [RFC 5035].[29]

The most widely accepted, standard S/MIME profile is [RFC 3851][30]. Not all cryptographic algorithms available for use in support of these features are appropriate for the protection of Federal government information. The S/MIME specifications allow the selection of individual algorithms. However, a number of cipher suites have been specified to define a specific combination of algorithms. Federal organizations **shall** use approved algorithms within S/MIME implementations for key establishment and transmitting messages. The following tables specify a variety of cipher suites that may be used to protect Federal information and information systems. Any of the algorithms listed in the following tables may be used, in accordance with security strength time frame restrictions given in Part 1, to protect Federal information in combinations other than those displayed.

Table 5-1: Cipher Suite 1 (ALS1)

Mechanism	Guidance
Digital Signatures	RSA with; key sizes ≥ 1024 bits [RFC 3447]
Hash	SHA-1 [FIPS 180-3]
Key Transport	RSA with key sizes ≥ 1024 bits [RFC 3447] and [SP 800-56B][31] (SP 800-56B under development)
Encryption	TDEA in CBC mode [SP 800-67]

[29] Both of these services are defined in S/MIME V3 standards [RFC 2634].

[30] RFC3851 is currently under revision. See http://www.rfc-editor.org/queue.html for additional information on draft 3851bis, draft-ietf-smime-3851bis-01.txt

[31] SP 800-56B is under development. A draft should be ready for public comment by the summer of 2008

Table 5-2: Cipher Suite 2 (ALS2)

Mechanism	Guidance
Digital Signatures	DSA with; key sizes ≥ 1024 bits [FIPS186-3]
Hash	SHA-1 [FIPS 180-2][32]
Key Transport	RSA with key sizes ≥ 1024 bits [RFC 3447] and [SP 800-56B][31]
Encryption	TDEA in CBC mode [SP 800-67, RFC 3851]

Table 5-3: Cipher Suite 3 (ALS3)

Mechanism	Guidance
Digital Signatures	RSA with keys ≥ 1024 bits [RFC 3447] and [FIPS 186-3]
Hash	SHA-1 [FIPS 180-3]
Key Transport	RSA with key sizes ≥ 1024 bits [SP 800-56B][311]
Encryption	AES-128 in CBC mode [FIPS 197] and [SP 800-38A]

Table 5-4: Cipher Suite 4 (ALS4)

Mechanism	Guidance
Digital Signatures	DSA with key sizes ≥ 1024bits [FIPS186-3]
Hash	SHA-1 [FIPS 180-3]
Key Agreement	Diffie-Hellman [RFC 2631] and [SP 800-56A]
Encryption	TDEA in CBC mode [SP 800-67], [SP 800-38A]

[32] FIPS 180-2 is under revision. See http://csrc.nist.gov/publications/PubsFIPS.html for additional information on FIPS 180-3.

Table 5-5: Cipher Suite 5 (ALS5)

Mechanism	Guidance
Digital Signatures	DSA with key sizes ≥ 1024 bits [FIPS 186-3]
Hash	SHA-256 [FIPS 180-3]
Key Agreement	Diffie-Hellman [RFC 2631] and [SP 800-56A]
Encryption	AES in CBC mode [FIPS 197] and [SP 800-38A]

Table 5-6: Cipher Suite B, Level 1*

Mechanism	Guidance
Digital Signatures	ECDSA with P-256 [X9.62]
Hash	SHA-256 [FIPS 180-3]
Key Agreement	ECDH with P-256 [SEC1]
Key Derivation	Based on SHA-256 [SEC1]
Key Wrap	AES-128 [RFC 3394]
Encryption	AES-128 in CBC mode [FIPS 197]

* RFC 5008

Table 5-7: Cipher Suite B, Level 2*

Mechanism	Guidance
Digital Signatures	ECDSA with P-384 [X9.62]
Hash	SHA-384 [FIPS 180-3]
Key Agreement	ECDH with P-384 [SEC1]
Key Derivation	Based on SHA-384 [SEC1]
Key Wrap	AES-256 [RFC 3394]
Encryption	AES-256 in CBC mode [FIPS 197]

* see [RFC 5008]

Note that after 2010, Federal systems **shall not** use SHA-1 as a hash algorithm for digital signatures. Also, at that time, algorithm and key size requirements may also change. (See SP 800-57 Part 1.)

Federal clients **shall** be supported by a Public Key Infrastructure with valid Federal PKI X.509 certificates for senders and receivers.

Cryptographic modules used in Federal systems **shall** comply with FIPS 140-2.

Federal S/MIME implementations may, in accordance with organizational policies, be capable of receiving messages protected with algorithm suites that are not approved for Federal use in sending protected messages. In those instances, users **should** be presented with a warning banner explaining that the cryptographic mechanisms used are weak and, therefore, integrity and authentication cannot be assured.

5.3 Procurement Guidance

The following recommendations are for any individual that makes a purchasing decision for acquiring an S/MIME-enabled component.

1. In support of security and compatibility across the Federal government, all Federal information systems **shall** support ALS 1 and ALS 2.

2. Procurement officials **should** buy products that support hash algorithms that provide more protection for digital signatures than SHA-1, such as ALS 5, Cipher Suite B level 1 and Cipher Suite B level 2 (see [FIPS 180-3]).

3. Federal clients **should** support MD5 and RC2 in the event that users receive correspondence signed or encrypted with these weaker, unapproved algorithms.

4. Federal agencies **shall not** use SHA-1 for digital signatures after 2010. Therefore, when selecting a product with S/MIME functionality, procurement officials **should** consider organizational needs for cryptography beyond 2010 and *purchase for the future*. Cryptographic algorithm implementations **should** be modular so as to allow for new algorithms.

5. If an S/MIME client needs to generate a key pair, then the S/MIME client or some related administrative utility or function **shall** be capable of generating public private key pairs on behalf of the user.

5.4 Recommendations for System Installers

The system installer is the individual that installs the S/MIME application and performs the initial configuration of the system.

1. Federal clients **shall** be configured to support cipher suites ALS1 and ALS2 for interoperability as described above in Section 5.2.

2. In accordance with security requirements beyond 2010, Federal clients **shall** be configured to support cipher suites with algorithms and key size requirements as described in SP 800-57, Part 1.

3. Systems **shall** be configured so that they only permit the use of approved cryptographic algorithms and approved key sizes to encrypt or sign new messages.

4. Installers **should** install and configure S/MIME clients such that they default to the use of an approved cipher algorithm suite. Furthermore, installers **should** configure clients such that there is a straightforward means for end users to change default settings and select algorithms as needed for interoperability and in accordance with organizational needs and policies.

5. System installers **should** configure clients such that end users can use unique certificates for each security function (e.g., encryption, digital signatures) at their disposal.

5.5 Recommendations for System Administrators

The System Administrator is the individual who runs the S/MIME application on a day-to-day basis and, on client implementations, interacts with the end user.

1. The system administrator **shall** ensure that end users are properly trained and that the organization's security policy is enforced.

2. In accordance with security requirements beyond 2010, Federal clients **shall** be maintained to support cipher suites with algorithms and key size requirements as described in SP 800-57 Part 1.

3. Systems **shall** be maintained so that they only permit the use of approved cryptographic algorithms and approved key sizes to encrypt or sign new messages.

4. Administrators **should** maintain S/MIME clients such that they default to the use of an approved cipher algorithm suite. Furthermore, administrators **should** maintain a straightforward means for end users to change default settings and select algorithms as needed for interoperability and in accordance with organizational needs and policies.

5. System administrators **should** provide training for users on the relative security provided by various cryptographic algorithms and on organizational policies for their use.

6. System administrators **should** provide end users with guidance on how certificates and keys are stored and managed, and identify the end user's related responsibilities.

5.6 Recommendations for End Users

An end user is the individual using a client to access the system. Even within a centrally managed environment, end users may find that they have a significant amount of control over some of the security features within an SMIME implementation.

1. Users **shall** operate their system as instructed by their organization and system administrators.

2. Users **should** use unique certificates for each security function at their disposal.[33]

3. Users **shall** protect their private keys from unauthorized disclosure.

4. Users **should not** send the same message both encrypted and in plain text.

[33] If they have not been supplied with certificates by their home organizations, users can obtain certificates from a number of organizations via the web.

6 Kerberos

6.1 Description

The Kerberos authentication mechanism was developed at MIT to enable the secure authentication of users to Target Servers (TSs) over an unprotected network, where client software acts on behalf of a user[34]. The original design and implementation of Kerberos and its first three revisions (i.e., versions 1 through 4) were primarily the work of Steve Miller, Clifford Neuman, Jerome Saltzer and Jeffrey Schiller[35]. Kerberos is used for local logins, remote (over the network) authentication, and for client-to-TS requests. It can also be extended to provide for the establishment of cryptographic keys between a client and a TS. Kerberos has been designed so that a user and a TS rely on a trusted third party to provide assurance of each party's identity. This assurance is granted by means of tickets and authentication information, each encrypted with symmetric keys.

The trusted third party is a Key Distribution Center (KDC), which consists of an Authentication Server (AS) and a Ticket Granting Service (TGS). The AS and TGS may or may not reside on the same machine. The KDC has a database of user, TS, and TGS symmetric keys. All KDC symmetric keys are accessible by the TGS. The user's key is normally created by hashing a user's password with other information.

An overview of the Kerberos version 5 protocol is shown in Figure 6-1. The following is a simplification of the process (e.g., the generation of most keys and the use of most cryptographic operations are not specified). For example, tickets and authentication information are protected with checksums and encryption when transmitted.

1. A user logs onto a client by entering a password, from which a user symmetric key is generated.
2. The client, acting on behalf of the user, requests a Ticket Granting Ticket from the AS.
3. The AS generates a Ticket Granting Ticket, for a specified validity period, and sends it to the client.
4. The client provides the Ticket Granting Ticket to the TGS, along with the client's own authentication information, which includes the client identifier and a time stamp.
5. The TGS checks the authentication information and the validity period of the Ticket Granting Ticket. The TGS then generates a Target Server Ticket and sends it to the client.

[34] Note that a single client implementation may be used by multiple users, and a single user may use multiple client implementations (e.g., a user could access different workstations, each with its own client implementation).

[35] The design was based in part upon a protocol proposed in by Needham and Schroeder [NEED] with modifications provided by Denning and Sacco [DENN]. For more detail on the goals, motivations, and rationale of Kerberos see [NEUM].

6. The client sends authentication information and the Target Server Ticket to the TS.
7. The TS checks the authentication information and the validity period of the Target Server Ticket; if the information is reasonable, the user is authenticated to the TS.

The protocol may be extended to authenticate the TS to the user, and a ticket may be re-used within its validity period.

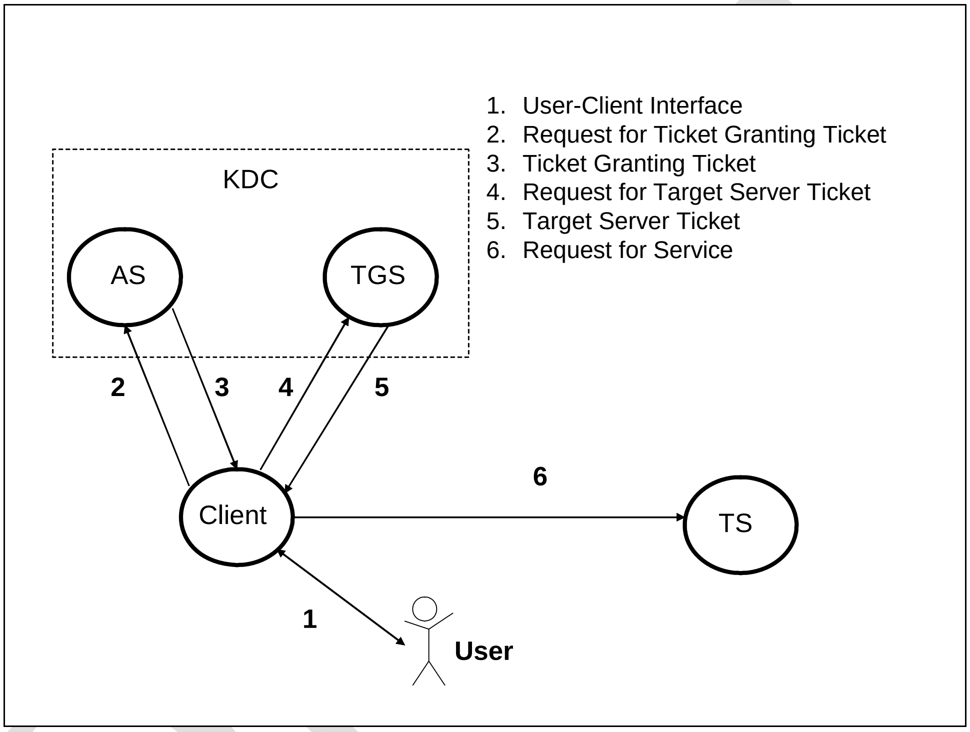
1. User-Client Interface
2. Request for Ticket Granting Ticket
3. Ticket Granting Ticket
4. Request for Target Server Ticket
5. Target Server Ticket
6. Request for Service
KDC
AS
TGS
2
3
4
5
6
Client
TS
1
User

Figure 6-1: The Kerberos Protocol

Each TGS has its own “realm” of clients and TSs. However, different realms may be linked by the sharing of inter-realm keys between TGS's (see Figure 6-2). A client in Realm 1 wishing a service on a TS in Realm 2 may obtain a ticket from TGS1 that introduces the client to TGS2. This ticket is encrypted with the inter-realm key shared between TGS1 and TGS2. The client can then request a ticket from TGS2 for the desired service on the TS in Realm 2. Thus, realms may be networked to provide clients with inter-realm services.

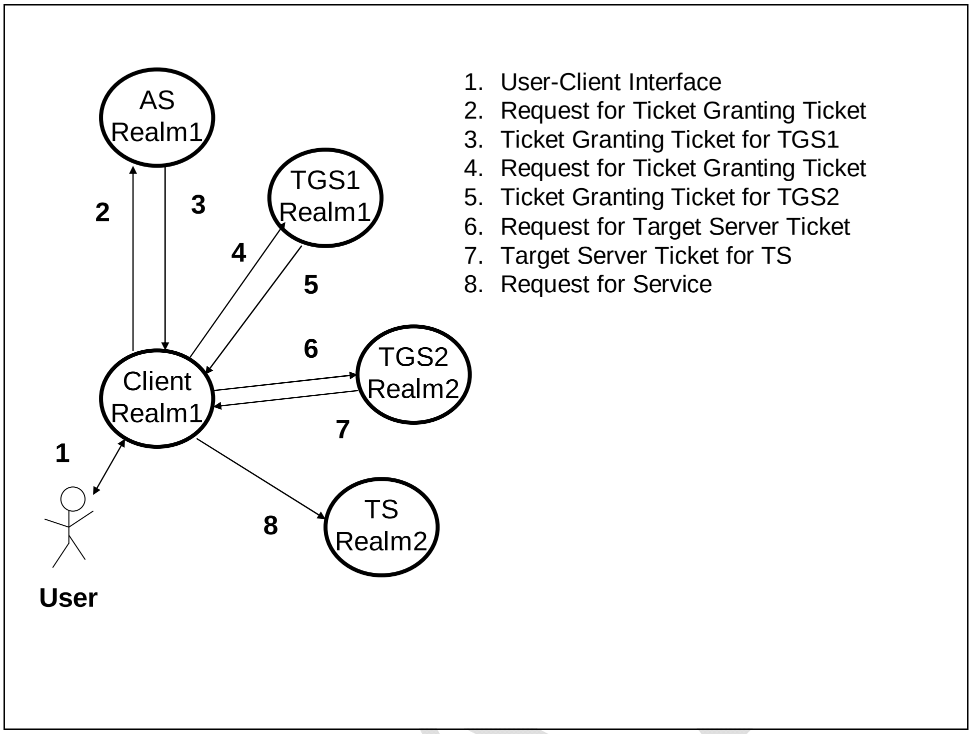
AS
Realm1
TGS1
Realm1
TGS2
Realm2
Client
Realm1
TS
Realm2
User
1
2
3
4
5
6
7
8
1. User-Client Interface
2. Request for Ticket Granting Ticket
3. Ticket Granting Ticket for TGS1
4. Request for Ticket Granting Ticket
5. Ticket Granting Ticket for TGS2
6. Request for Target Server Ticket
7. Target Server Ticket for TS
8. Request for Service

Figure 6-2: Cross-Realm Authentication

In an alternative Kerberos protocol between the client and the AS (as specified in [RFC 4556]), either both the user and the AS have key establishment public key pairs with corresponding certificates, or the user has a key establishment key pair and associated certificate, and the AS has a digital signature key pair and digital signature certificate. The user symmetric key can then be established between the client and the KDC in one of two ways:

1. using key agreement (e.g., Diffie-Hellman) between the AS and the client[36], or
2. using key transport (e.g., RSA), where the AS generates the user symmetric key, and sends the key to the client[37].

Once the user symmetric key is established, the remainder of the protocol proceeds as previously described. The need for user symmetric keys generated from passwords can thus be avoided.

[36] In this case, both the user and the AS have key establishment key pairs.

[37] In this case, the user has a key establishment key pair, and the AS has a digital signature key pair.

6.2 Security and Compliance Issues

1. Kerberos version 5 was initially specified in [RFC 1510]. More recently, the security was updated in version 5 [RFC 4120]; however, many existing implementations still correspond to the initial RFC.
2. Many current Kerberos implementations based on [RFC 1510] rely on DES for symmetric encryption functions. DES is no longer approved for use in protecting Federal government information.
3. If a keyless checksum computation is used for the data integrity of Kerberos messages, the integrity of the message may be inadequate.
4. Some Kerberos implementations rely solely on the entropy (i.e., randomness) provided by the user password to generate the symmetric client key (the client key is a hash of the user's password). Passwords, in general, do not provide enough randomness for generating a key. In such cases, a dictionary attack[38] is feasible. If passwords are used to generate cryptographic keys, they **should** be selected to maximize the difficulty of a password guessing attack, thus increasing the difficulty of an off-line dictionary attack [SP 800-118].[39]
5. Compromising the client, KDC, or TS could compromise the symmetric keys that they contain and thereby compromise the system. In particular, the KDC stores keying information for all the KDC users, the TGS, and any TS that communicates directly with the KDC TGS. These symmetric keys require protection that is commensurate with the protection required for the data that they protect (e.g., tickets, other keys, authentication information, and shared data).
6. Tickets are kept in the memory of both a client and a TS. The protection of these tickets is dependent on the strength of the access control protection afforded by the client and TS systems.
7. The TGS has read-only access to the KDC database. If the TGS and database do not reside on the same machine, a secure channel is required for the TGS to obtain the required TS keys.
8. A failure of the AS or the TGS would prevent all users from obtaining services from the TS.
9. Clocks must be synchronized in order to accurately assess the validity of clock authentication information and tickets. If the TS is running late, then previous authentication information and tickets could be played back to the TS after they have expired.

[38] A **dictionary attack** is a technique for guessing a password by selecting candidate passwords from a list of words commonly found in a dictionary, or derived from words commonly found in a dictionary. Each selected candidate is tested as though it were the actual password until the result of the test indicates that the correct password has been selected.

[39] SP 800-118 Guide to Enterprise Password Management is currently under development. A draft will be available viewing and comment by November 2008.

6.3 Procurement Guidance

The following recommendations are for any individual that makes a purchasing decision for acquiring a Kerberos capability.

1. New procurements **shall** conform to version 5 [RFC 4120].
2. Government procurements **shall** specify the inclusion of approved symmetric key encryption algorithms (e.g., the Advanced Encryption Standard (AES)) [RFC 3962].
3. An approved MAC computation (e.g., HMAC-SHA1) **shall** be available for data integrity with encryption (See [RFC 3962]).
4. Kerberos version 5 permits the use of smart cards or tokens (e.g., FIPS 201 Personal Identity Verification cards, [FIPS 201]) to store a user's password. Passwords stored on tokens **shall** be randomly generated; therefore, when tokens are used, a means of generating random passwords and securely writing them on the token **shall** be available. When tokens are used, manual entry of passwords **shall not** be permitted except to authenticate the user to the token.
5. If passwords are used to form user symmetric keys, then the password mechanism **shall** support the use of strong passwords, [SP 800-118],[39] and an approved hash algorithm (e.g., SHA-1 or stronger) **shall** be used as the hash algorithm.
6. If passwords are generated by users, then the system software **shall** enforce a strong password policy in accordance with [SP 800-118].[39]
7. Kerberos with public key authentication and subsequent key establishment can provide stronger security than the use of password-based keys and **should** be available where PKI mechanisms are available. See [RFC 4556] for further information.
8. Procurement officials **should** consider whether inter-realm networking is necessary and include the capability in the software if it's needed.
9. Cryptographic modules used by CAs, TSs and clients **shall** be validated at FIPS 140-2 Level 1 or higher.

6.4 Recommendations for System Installers

The system installer is any individual(s) that installs a Kerberos capability and performs the initial configuration of the system.

1. New systems **shall** conform to version 5 [RFC 4120].
2. Government systems **shall** be configured so that approved algorithms (e.g., AES) **shall** be used (see [RFC 3962]), and DES **shall not** be used.
3. An approved MAC checksum (e.g., HMAC-SHA1) **shall** be installed and used in all implementations for data integrity purposes [RFC 3962].

4. The AS, TGS, TSs, and clients **shall** use strong access control mechanisms[40] (physical and logical) for protecting and updating keys.
5. Kerberos version 5 permits the use of smart cards or tokens (e.g., FIPS 201 Personal Identity Verification cards) to store a user's password. Passwords stored on tokens **shall** be randomly generated; therefore, when tokens are used, a means of generating random passwords and securely writing them on the token **shall** be used. When tokens are used, manual entry of passwords **shall not** be permitted except to authenticate the user to the token.
6. Kerberos with public key-based user authentication and key establishment can provide stronger security than password-based keys and **should** be installed where PKI mechanisms are available and the software has the capability. See [RFC 4556] for further information.
7. If user passwords are generated by the system, the system **shall** generate strong passwords [SP 800-118].[39]
8. If passwords are used to form user symmetric keys, then an approved hash algorithm (e.g., SHA-1 or stronger) **shall** be used as the password hashing algorithm.
9. If user passwords are used to generate cryptographic keys, the password mechanism **shall** be configured to use and require strong passwords [SP 800-118].[39]
10. A backup AS and TGS **should** be provided so as to minimize the impact in case of operational failure or denial-of-service attacks.
11. Clocks **should** be synchronized periodically and whenever a new system is brought on-line[41]

6.5 Recommendations for System Administrators

The System Administrator is any individual(s) who runs a system with a Kerberos capability on a day-to-day basis and interacts with the end user.

1. System administrators **shall** ensure that users are properly trained and that the organization’s security policy is enforced.
2. The AS, TGS, TS, and client **shall** be physically secured.
3. Tickets **shall** be encrypted or physically protected at the client, TS, and TGS sites.

[40] Strong access control mechanisms either prevent or detect unauthorized attempts to access or replace sensitive data. These controls may be physical (e.g., locks, guards, or alarms) or logical (e.g., encryption, data integrity, or entity authentication).

[41] (see http://www.nist.time.gov).

4. If the passwords are generated by the user, then the system administrator **shall** develop a policy for selecting strong passwords that is enforced by the software [SP 800-118].[39]
5. System clocks **shall** be periodically verified to ensure synchronization.

6.6 Recommendations for End Users

An end user is the individual using the Kerberos capability.

1. If user-selected passwords are allowed, they **shall** be generated in accordance with the organization's password policy**.**
2. Users **shall** protect their password from unauthorized disclosure. If a token containing a password or key is provided, users **shall** protect the token from unauthorized use. Users **shall** report the loss of physical tokens or the compromise of passwords.

7 Over-The-Air Rekeying (OTAR) Key Management Messages (KMMs)

7.1 Description

A key management protocol has been specified for over-the-air rekeying of digital radios [OTAR]. This protocol has been designed to handle several types of cryptographic security, one of which, Type 3, has been designed for unclassified, sensitive communications and is discussed herein. The only differences between the security types are the cryptographic algorithms used and the security requirements. The Type 3 algorithms and security requirements are addressed in both [OTAR] and [OTAR1][42].

For key management, a secure mobile system consists of Key Management Facilities (KMFs) and mobile radios that are subordinate to each KMF. Key Management Messages (KMMs) are exchanged between each KMF and its subordinate mobile radios (see Figure 7-1). Cryptographic keys are transferred from a KMF to a mobile radio, protected using a key-wrapping algorithm and key wrapping key; many of the KMMs are protected by encrypting the data in the messages; the integrity of the messages is protected using a Message Authentication Code (MAC).

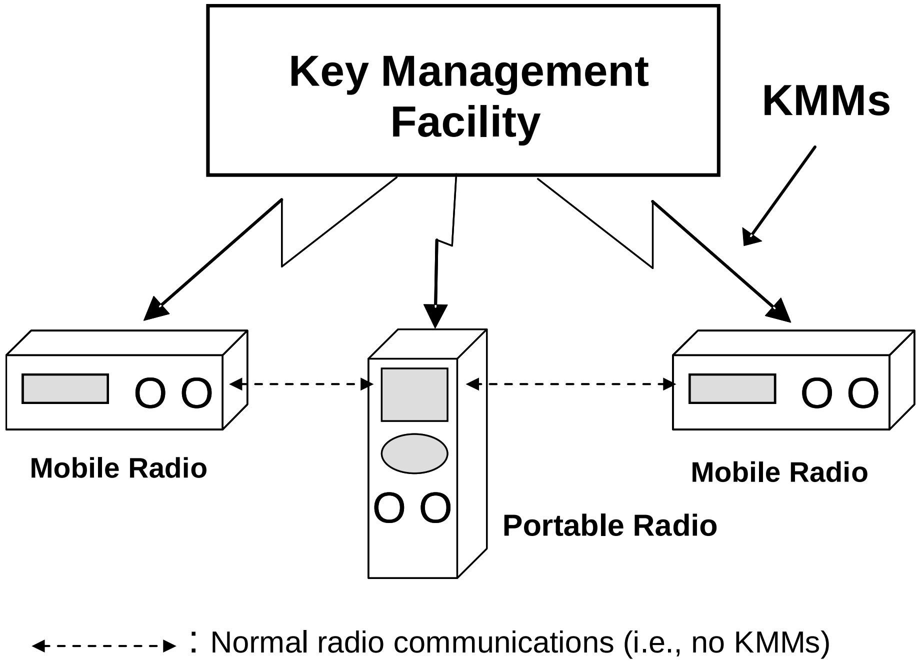

Figure 7-1: Radio Communications with OTAR

[42] [OTAR] provides an overview of the key management techniques and the protocol. [OTAR1] specifies the general security requirements for transmitting Type 3 key management messages (KMMs), the requirements for wrapping the keys, the techniques used for KMM integrity and the mechanism used to protect against replay of the KMMs.

Three general types of keys are used in OTAR: a Key-Wrapping Key (KWK), a Traffic-Encryption Key (TEK) and a key to be used for the computation of a Message Authentication Code (MAC).

7.2 Security and Compliance Issues

7.2.1 Cryptographic Algorithms

Although the protocol has been designed to allow the use of any block cipher algorithm to apply the cryptographic protection, only three block cipher algorithms have been included in the specification: DES, TDEA and AES.

Approval for DES has been withdrawn because DES no longer provides the security that is needed to protect Federal government information.

TDEA [SP800-67] uses three DES encryption/decryption operations with a “key bundle” consisting of three separate DES keys. Two versions of TDEA have been included in the OTAR specification: a one-key version, whereby all three keys are the same for compatibility with DES, and a three-key version (3-TDEA), whereby the three keys are different. Since DES is no longer considered secure, the one-key version of the TDEA is also no longer considered secure.

7.2.2 Message Authentication and Cryptoperiods

A Message Authentication Code (MAC) is used to authenticate and protect the integrity of many of the KMMs as specified in [OTAR1], using the CBC-MAC mode of operation. The security of a MAC depends, in part, on the block size of the MAC algorithm. AES has a larger block size than 3-TDEA, and so the security of AES CBC-MAC is better than 3-TDEA CBC-MAC. The OTAR documentation provides no guidance on the length of cryptoperiods (i.e., the number of messages or the length of time that a key may be used before it must be changed).

For AES, the number of messages that can be authenticated using a given key is, in practice, not an issue. However, AES keys **shall** be periodically updated because of other threats to the system, e.g., lost radios or an undetected compromise of a key.

When using 3-TDEA, no more than 1,000,000 messages **shall** be sent using a given key because of threats to the security of the algorithm. However, like AES, it may be prudent to update the 3-TDEA keys more frequently because of other threats to the system.

7.2.3 Key Usage

Part 1 of this Recommendation states that keys **should** be used for only one purpose.[43] However, [OTAR1] states that the key used to generate a MAC must be either a key reserved for authentication and integrity protection purposes, or a key derived from a Traffic-Encryption Key (TEK) using a key wrapping algorithm. In this latter case, note that the TEK might be used for both encryption and for key derivation. In order to comply with the recommendation to use a key for only one purpose, the MAC key **should** be a key reserved for a single purpose.

7.2.4 Backup

The KMF **should** backup all keying material shared with and among the mobile radios so that it can be recovered if necessary. When a key is no longer required, it **should** be deleted from both normal operational storage and backup storage.

7.2.5 Rekeying

Procedures **shall** be in place to rekey all radios in the network in the event of a key compromise. If a radio is lost, procedures **shall** enable rekeying other radios in the network so that the lost radio no longer has the capability of communicating securely with other radios in the network.

7.2.6 Random bit generators

Keys **shall** be generated at the KMF using an approved random bit generator that provides sufficient randomness for the desired security strength of the cryptographic processes [SP 800-90].

7.3 Procurement Guidance

The following recommendations are for any individual(s) that makes a purchasing decision for acquiring OTAR equipment.

1. Procurements **shall** include the AES or TDEA algorithm.

2. If TDEA is provided in an implementation, the three-key version **shall** be included.

3. Procurements that include TDEA **shall** be capable of limiting the number of uses of a single TDEA key bundle to 1,000,000.[44]

4. KMFs and radios **shall** conform to [OTAR] and [OTAR1].

[43] There is an allowed exception to this rule, but it does not apply to OTAR (see Section 5.2 in Part 1).

[44] See Section 7.2.2.

5. When keys are generated within KMFs**,** they **shall** be generated using approved random bit generators.

6. Cryptographic modules used by the KMF and the mobile radios **shall** be validated at FIPS 140-2 Level 1 or higher.

7. KMFs **shall** include backup and archive capabilities to support reconstitution of the KMF in the event of a disaster (e.g., fire, earthquake).

7.4 Recommendations for System Installers

The system installer is the individual(s) that installs an OTAR capability and performs the initial configuration of the system components.

1. The KMF **shall** use and have strong physical and logical access control mechanisms to protect the cryptographic keys (e.g., physical locks, alarms or password token).

2. Backup KMFs **should** be provided, along with a strategy and procedures to transition from a primary KMF to a backup KMF, and from a backup KMF to the primary KMF.

3. A TEK **should not** be used for multiple purposes. Reserved MAC keys **should** be used for message authentication and integrity protection.

4. Maximum cryptoperiods for each key type **shall** be determined at the KMF in accordance with the organization's security policy and Section 7.2.2.

5. Radios **shall** be accounted for; in the case of a lost or stolen radio, an assessment of the effect of a loss of the keys contained in that radio **shall** be made. The use of any key contained in that radio **shall** be discontinued. Procedures **shall** be in place for replacing these keys if used by the KMF or by other radios.

6. Implementations **shall** be configured to use the AES or TDEA algorithms, and to disallow the use of DES.

7. If TDEA is provided and is to be used, the three-key version **shall** be used, and the one-key version **shall not** be used.

8. For implementations using TDEA in which the cryptoperiod of a key bundle is configurable, the cryptoperiod **shall** be set to a value less than 1,000,000 messages.

7.5 Recommendations for System Administrators

The System Administrator is the individual who manages the OTAR system or its components on a day-to-day basis and interacts with the end users.

1. System administrators **shall** ensure that the organization's security policy is enforced.

2. System administrators **shall** protect the keying material from disclosure and modification.

3. Procedures **shall** be in place for replacing keys at the end of their cryptoperiod.

4. To maintain the availability of the KMF, system administrators **shall** ensure that sufficient information is stored in a secure location to reconstitute the KMF after a disaster.

5. System administrators **shall** train end users in the use of their radios and the procedures to be followed in the case of lost radios or suspected key compromises.

6. Audit logs **should** be maintained at the KMF with sufficient information to indicate which keys are shared by which radios.

7.6 Recommendations for End Users

An end user is the individual using a radio that has an OTAR capability.

1. End Users **shall** operate radios as instructed by their organization and system administrators.

2. End users **shall** protect their radios from loss and unauthorized access.

3. In the event that a radio is lost or a key is suspected of being compromised, end users **shall** immediately notify the system administrator in accordance with the organization's security policy.

8 Domain Name System Security Extensions (DNSSEC)

8.1 Description

Domain Name System (DNS), as defined in [RFC 1034] and [RFC 1035], is the global hierarchical distributed database system for mapping Internet addresses, SMTP servers, and other information to a human readable name. Its main purpose is handling mappings between host domain names and Internet addresses, but it can handle other forms of data as well, such as host system information, geographic-location of servers, even encoded digital certificates. DNS data is stored as individual Resource Records (RR) that associates a piece of data (e.g., IP address, mail server name) with a domain name and an identifying Resource Record type code (RR type). All the RRs for a particular organization are stored in an administrative unit called a zone. Multiple zones form a domain. A domain is hierarchical, in that one zone may act as a delegating parent to one or more child delegated zones. For example, most Federal agencies are child delegations under the ".gov" parent zone.

Zone information is maintained on *authoritative* servers, which are distributed all over the Internet to answer queries according to the DNS network protocols. The DNS infrastructure is comprised of a small group (or single server) known as a primary master authoritative server that has a local zone database, and multiple secondary servers that obtain their copies of the zone database from the primary authoritative master server. Another set of components are *caching recursive* servers[45], which query the authoritative servers and cache any replies. On the end user's client system, software components known as resolvers make DNS queries to recursive caches and/or authoritative servers. Figure 8-1 depicts the relationship between the DNS components.

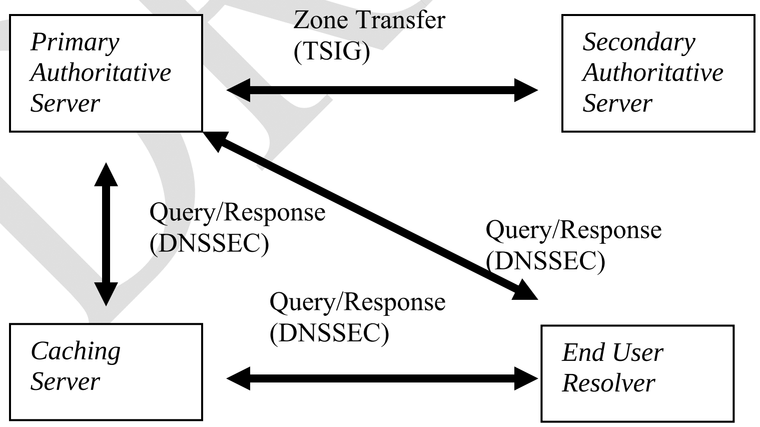

Figure 8-1: DNS Components

[45] Caching Recursive Server is sometimes shortened to "caching server" or "recursive server". However, the role remains the same.

The basic DNS does not have many security features [SP 800-81]. A suite of RFC's has been developed to provide security enhancements contained in three IETF documents, collectively called the DNS security extensions (DNSSEC) [RFC 4033, RFC 4034, and RFC4035]. DNSSEC provides a layer of authentication and integrity protection for any kind of data stored in the DNS, including data used by other protocols. For example, there are RR types allocated for storing Secure Shell (SSH) keys in the DNS, which then rely on DNSSEC to protect the integrity of that information.

8.1.1 DNS Data Authentication

Cryptographically generated public key-based digital signatures provide authentication for DNS data. Commonly, there will be two or more digital signature public key pairs (which make up the key set) used to implement DNSSEC in a zone. One key pair is used to sign the zone data (referred to as the Zone Signing Key or ZSK), and a separate, stronger key pair is used to sign the zone key set (known as the Key Signing Key or KSK). This KSK is also known as the Secure Entry Point (SEP) key for the zone – using it, a client can authenticate the ZSK (by validating the signature over the ZSK using the KSK public key), and then use the ZSK to authenticate the zone data. The KSK is also used to link the security chain[46] from the zone to its delegating parent. Since the KSK is used to link security from the zone (e.g. "example.gov") to the delegating parent zone (e.g. ".gov") [RFC 4035], it is often longer lived, and used infrequently (used only to sign the zone key set). Multiple digital signature algorithms can be supported, so there may be multiple keys (one for each algorithm), as there is no algorithm negotiation in DNSSEC, and clients may only understand certain digital signature algorithms. There is one mandatory-to-implement algorithm as defined by the IETF, so there is at least one agreed-upon digital signature algorithm that all servers and clients will understand.

When this document was published, the only mandatory-to-implement algorithm to use with DNSSEC data authentication is RSA using SHA-1 as a hash algorithm.[47] When the use of SHA-256 (or greater) is included in the specification, there **shall** be an immediate transition plan to migrate from the use of RSA with SHA-1 to the use of RSA with SHA-256 [FIPS 180-3] for DNS data authentication. However, both hash algorithms (i.e., SHA-1 and SHA-256) **should** be used to generate digital signatures for DNS data for a period of time to ensure that client systems that cannot validate RSA with SHA-256 can still authenticate DNS data. The length of this transition period depends on the widespread availability and deployment of client system software that understands RSA with SHA-256.

[46] The security chain (also referred to as "chain of authentication") is the collection of digital signatures and public keys that can be used to trace a logical path from the data to be validated back to a trusted, installed public key on the client [SP 800-81]. This chain of public keys and signatures is similar to a PKI certificate chain (see Section 2.1), but entirely contained within the DNS.

[47] Note that the transition away from SHA-1 in 2011 is necessary when the hash function is used for digital signatures, but not when used for HMAC.

8.1.2 DNS Transaction Authentication

Additional authentication mechanisms are used for server-server communication and administrative control. Transaction authentication is performed by computing an HMAC over the entire DNS message and a secret random string that is known by both authoritative DNS servers in the transaction, and transmitting the result in a Transaction Signature (TSIG) RR appended to the original message. Transaction authentication is usually used for special transactions, such as zone transfers or dynamic updates. A zone transfer is a special query type that is used to keep secondary authoritative servers up-to-date with the most recent version of the zone data. Dynamic update is a feature that allows an authorized administrator to add or delete DNS data by sending a specially formatted message. This is frequently used in local area networks where the Dynamic Host Configuration Protocol (DHCP) is used to assign IP addresses dynamically. The DHCP server may update the DNS server by sending a dynamic update message to reflect network changes.

The currently defined algorithm used in TSIG authentication is HMAC using SHA-1. This is deemed acceptable for current security practices when using HMAC with a suitably random secret string [SP 800-90]. Newer implementations allow for stronger hash algorithms to be used (SHA-256 and stronger) and may be used when available. All DNS server administrators taking part in the transaction must agree on which algorithm and secret string size will be used for transaction authentication and must ensure that all parties have the same secret random string (which may include out-of-band transactions to distribute keys).

8.1.3 DNS Cryptographic Algorithms/Schemes, Modes and Combinations

DNS does not support algorithms in isolation, but specifies suites of algorithms and schemes. Algorithm/Scheme combinations for zone data signing and for message authentication are provided in Tables 8-1 and 8-2:

Table 8-1: Recommended Algorithm and Scheme Combinations for Zone Data Signing

Suite	Authentication	Digest	IETF Status	Approved for Federal Use[48]
RSA_SHA1	RSA	SHA-1	Mandatory	YES until 2010
DSA_SHA1	DSA	SHA-1	Optional	YES until 2010
RSA_SHA256	RSA	SHA-256	Optional	YES
RSA_SHA512	RSA	SHA-512	Optional	YES

[48] Using approved key sizes as discussed in Part 1 of this guide.

Table 8-2: Recommended Message Authentication Algorithms

Suite	IETF status	Approved for Federal Use
HMAC_SHA1	Mandatory	YES
HMAC_SHA256	Optional[49]	YES
GSS_TSIG[50]	Optional	YES

It should be noted that HMAC-MD5.SIG-ALG.REG.INT is a suite that is widely implemented and often set as the default choice. However, it **shall not** be used for Federal implementations. Since TSIG message authentication is used between servers where there is an existing trust relationship, the administrators must agree to the method used and the secret (random) string used with the TSIG method.

Currently, there are plans for migrating to SHA-256 and SHA-512 by the IETF community for use with RSA in zone data signing. The registry maintained by the Internet Assigned Number Authority (IANA) function for DNSSEC algorithm codes has several entries available for use, but the use of SHA-256 has yet to be finalized and implemented in current software. However, the use of SHA-256 is encouraged and **should** be considered when supported by DNSSEC software. There **should** be a transition period when SHA-1 and SHA-256 would both be used to ensure that clients that are non-SHA-256-aware could still validate signatures until they are upgraded.

8.1.4 Special Considerations for NSEC3

There is a special varient to DNSSEC that minimizes the risk of information leakage and is known as the Hashed Next Secure (NSEC3) RR [RFC 5155]. In DNSSEC, a client can map the contents of the zone by sending a series of queries for the Next Secure (NSEC) RR type found in error messages. These NSEC RRs provide signed proof that the queried name did not exist, but also provides two names that do exist in the zone as part of that proof. NSEC3 attempts to minimize this information leakage of zone names by using the hash values of the two existing names (currently using SHA-1 only). However, this requires the server and client to be able to perform multiple SHA-1 hash calculations during runtime; note that this method could be used to mount a Denial of Service attack against the server if multiple requests are made.

NSEC3 was designed to solve a specific class of information leakage that could lead to a complete mapping of network resources in a DNS zone. NSEC3 deployment risks are often greater than the usefulness provided by using NSEC3, unless there is an overridding need to deploy NSEC3 beyond zone content protection (examples include protecting personally indentifying information that may be contained in the DNS). However, it is a

[49] As of the time of writing, SHA-256 has begun to be implemented in some DNS servers

[50] Generic Security Service Algorithm for Secret Key Transaction Authentication (GSS-TSIG [RFC3645]) may be found in some server implementations

good idea to use an NSEC3-aware client software, because a client may access a zone that uses NSEC3 RRs with DNSSEC.

As with other issues in DNSSEC involving secure hash algorithms, SHA-1 is the most widely used algorithm. As of the time of writing, SHA-256 has yet to be widely implemented. System installers and administrators **should** develop a transition plan to migrate from SHA-1 to SHA-256 when SHA-256 becomes available in major software distributions to prevent an attacker from performing a brute force attack against the hashed names and obtain the entire contents of a zone database. This would involve deploying both SHA-1 and SHA-256-based NSEC3 RRs until it is observed that SHA-256-aware implementations have become widely used in the Internet community.

8.2 Security/Compliance Issues

1. For digital signatures over DNS data, only SHA-1 (used with RSA or DSA) has both Federal government approval as a hash algorithm and is fully specified in DNSSEC (as of the time of writing). It is the only approved hash algorithm available to use until SHA-256 with RSA and SHA-512 with RSA is implemented in DNSSEC software.

2. Although not strictly necessary to the specification, a Key Signing Key **should** be used to maintain security chains from the parent zone (e.g. .gov) to the zone (e.g. nist.gov). This KSK **should** be securely transmitted to the delegating parent according to the policy and procedure established by the parent zone.

3. TSIG shared secret strings **should** be random for use in providing integrity protection for DNS message transactions, and generated at an appropriate security strength. The system installers using the TSIG secret string **shall** agree on which TSIG algorithm to use.

4. System developers **should** implement SHA-256 [FIPS 180-3] for use with digital signatures when the use of SHA-256 is specified through the IETF standards process.

8.3 Procurement Guidance

A purchase decision maker is any individual that makes a purchasing decision for acquiring a DNS component. Recommendations for purchasing are:

1. DNSSEC utilities **shall** use FIPS140-2 compliant cryptographic modules.
2. DNS server software **should** generate and serve NSEC3 RRs, if required by zone policy.
3. DNS server software **shall** use an approved random bit generator to generate random strings for use with TSIG message authentication using HMAC that is consistent with the hash algorithm security strength recommendations in Part 1.

4. DNSSEC-enabled versions of network applications **shall** be purchased as required by security policy, if available.
5. DNSSEC software implementing SHA-256 **shall** be included in procurements when available. A transition plan deploying digital signatures using both RSA/SHA-1 and RSA/SHA-256 **should** be used for some period of time until wide-scale deployment of SHA-256 is observed.

8.4 Recommendations for System Installers

The system installer is the individual(s) that installs a DNS component and performs the initial configuration of the component.

8.4.1 Recommendations for System Installers (Authoritative Servers)

1. Authoritative server installers **shall** configure a DNS authoritiative server to serve DNSSEC signed zone data.

2. Authoritiative server installers **shall** use an approved random bit generator (as discussed in NIST Special Publication 800-90 [SP 800-90]) to create and configure an initial random secret string for use with TSIG in transactions.

3. Authoritative servers **shall** be configured to generate and sign zone data with a key pair that is consistent with the key size recommendations for digital signatures as specified in Part 1.
4. Authoritative servers **shall** be configured to generate and sign the key set with a key pair that is consistent with the key sizes recommended for digital signatures, as specified in Part 1.
5. Authoritative servers **shall** be configured to generate and use a random secret string for zone transfer message authentication (via TSIG) between primary and secondary servers. The security strength of the random bit generator process **shall** support the security strength required by the servers.
6. Authoritative servers **shall** be configured to generate and use a separate shared secret string for dynamic update message authentication (via TSIG). The security strength of the random bit generator process **shall** support the security strength required by the servers.

8.4.2 Recommendations for System Installers (Caching Recursive Servers)

1. Recursive caching server installers **shall** configure DNS servers to be DNSSEC-aware.

2. Recursive caching server installers **shall** install at least one public key used for DNSSEC validation.

8.4.3 Recommendations for System Installers (Client Systems)

1. Client systems **shall** be configured to send DNS queries to a DNSSEC-enabled caching recursive server.
2. Client systems **should** be configured to use DNSSEC-enabled applications, if they are available.

8.5 Recommendations for System Administrators

The System Administrator is the individual who runs the DNS application on a day-to-day basis and interacts with the end user.

8.5.1 Recommendations for System Administrators (Authoritative Server)

1. Organization security policy regarding the Authoritative servers **shall** be enforced.
2. Cryptographic keys **shall** be protected as specified in Part 1.
3. System administrators **shall** replace zone data signature Resource Records before the end of their validity period.
4. Zone data signature Resource Records **shall** be replaced if the private key associated with the public key is compromised, when the administrator of the DNS zone leaves the organization, and for other reasons listed in the organization's security policy.
5. Administrators **shall** utilize methods for handling and protecting the private key (e.g., using a smart card that requires appropriate user authentication).
6. Administrators **shall** follow the key lifecycle procedures found in NIST Special Publication 800-81 Secure Domain Name System (DNS) Deployment Guide [SP 800-81]

8.5.2 Recommendations for System Administrators (Caching Recursive Servers)

1. Server administrators **shall** ensure that there is an organization security policy for using the Caching Recursive Server.
2. Cryptographic keys **shall** be adequately protected (see Part 1).
3. The trust anchors for DNS validating caches **should** be kept up to date.

8.5.3 Recommendations for System Administrators (Client Systems)

1. Server administrators shall ensure that there is an organization security policy for using the client systems.
2. Server administrators shall ensure that users are properly trained.

8.6 Recommendations for End Users

An end user is the individual using a client to access the system.

1. End users shall operate their client systems as instructed by their organization and system administrator.

9 Encrypted File Systems (EFS)

9.1 Description

The encryption of data files and complete disk volumes presents a somewhat different set of key management issues than those for encryption of network-communicated data. While network communications security focuses on the privacy and integrity of information in transit, storage (e.g., file) security focuses on the privacy and integrity of persistent data and the secure sharing of this data. The key management guidance surrounding file and volume encryption are sufficiently similar to consolidate into a single section.

Unlike the previous sections where the protocols and/or standards have been thoroughly studied by the network security community, the commercial solutions for file encryption utilize a wide variety of security schemes and methods for storing keys. Due to this range of solutions, this section will not be comprehensive, but will cover a variety of methods used for file encryption. One group that is currently working to develop standards for encrypted file systems is the Trusted Computing Group (TCG).

The most important questions that designers of a file encryption system must answer are:

- How are keys used in the system, and what protection are they afforded?
- Where are the keys stored on the system?
- Does the method scale upward for numerous user communities (without requiring an impractical number of keys to be stored)?

9.1.1 Number of Keys Required

For an Encrypted File System, keys are used to encrypt a file or group of files. The system can either encrypt each file with a distinct symmetric key or encrypt a set of files using the same symmetric key. In the first case, it is very easy to provide access to a sharing user, for example, by simply giving the key to that user. In this way, only that single file can be accessed by the sharing user without providing access to any of the other files in the system. The drawback with this first method is that if many files are encrypted, the model can quickly become unwieldy, since a key is required for each encrypted file and must be provided to the sharing user.

In the second case, many fewer keys are required, which eases the key distribution process. However, when a sharing user requires access to a single file, giving access to that user is more problematic. By simply sending a key to the sharing user, access would be provided to all of the owner's[51] files that were encrypted using that key, rather than to the single file.

However, there are several cryptographic management actions that could be used to grant access to an individual file. One option to limit access in this second system is for the

[51] An owner could be an individual or a group of individuals or processes that share the key.

owner or system to decrypt the file and re-encrypt it using a new key for transmission to the sharing user (e.g., using network security mechanisms and session keys). In the increasingly rare case of both users sharing a common file server, the decrypted and re-encrypted file would be placed in the sharing user's file space. This would require significant processing overhead and a key management protocol for exchanges between the owner and the sharing user or between the file system management process and the sharing user. This process requires proper protection for these new keys, at the same security level as the original key protecting that file.

Another option is for the sharing user to be provided the encrypted file and the owner's decryption key that could be used to decrypt all of the owner's files, including the one provided to the sharing user. System overhead is reduced, and it may be possible to protect the key from third party administrators, but it is unlikely that the owner would agree that the requester should be granted access to all files protected under the common key.

Having provided more extreme examples in the previous two cases, the following is a more common approach that is used. In this case, each user on the system has an asymmetric key pair. Each owner's file is encrypted under different randomly generated symmetric File Encrypting Key (FEK). The FEK is then encrypted using the public key of the owner and stored with the encrypted file. When the owner of a file wants to share it with another user, the owner decrypts the encrypted FEK (using her private key) and then encrypts the FEK using the sharing user's public key. The encrypted file and re-encrypted FEK can then be provided to the requester. This system has several advantages. First, the owner needs to manage only a single asymmetric key pair. Second, it permits easier file sharing between users. Third, it is very efficient because files do not have to be re-encrypted in order to be shared. Finally, the system need not manage any keys separately, since the asymmetric keys are managed by the owners, and the FEKs are stored in encrypted form with the files.

The owner can, of course, use different keys to encrypt different files or sets of files. The fewer files that the owner chooses to encrypt with a given key results in more keys and associations (e.g., associations of keys with file identities, file groups, individual identities, or access groups) that would need to be maintained.

Another important concept within an encrypted file system that must be considered is how data recovery is implemented. If a user loses his keys, without a data recovery capability within the system, the user's data is permanently lost. As such, it is vital that some form of data recovery, such as master administrator passwords, be included.[52] This requires a file encryption system that allows multiple passwords to decrypt the file, one of which is provided to the user, and the second is provided to the administrator, in the form of a master administrator password. Another possibility is that the administrator has a method of storing the user's passwords for use only when the user has lost or forgotten their password.

[52] The specifics of how data recovery is accomplished are beyond the scope of this document.

When the scope of the system expands from a pair of individuals to a large network or internet-work, the factors associated with the management of keys can become unwieldy. Key management challenges associated with large systems include the following:

- Maintaining context in the face of global data placement (many owners and large quantities of data)
- Very large numbers of keys to manage and distribute
- If numerous keys are stored in a single location within an EFS, that site provides an attractive target for an adversary, a single point of trust for a large domain.
- Difficulty in accounting for revoked users (individuals who have left an organization, whose subscriptions have expired, or otherwise **should not** be authorized to access the file any longer).
- Reassignment of ownership of protected data to another individual or organization.
- Recovery of data in the event of lost keys (e.g., the case of archived encrypted data that is encrypted and stored by an individual who has left the organization and cannot be found).
- A sharing user who has been provided the keys to a number of the owner's files, then provides the keys, and the owner's data, to additional users.

9.1.2 Access to Symmetric Keys used in File Encryption

After the decision has been made about the number of files to be protected with a single symmetric key, key management questions can be considered. How does the File Encryption System generate the encryption keys? How will the keys be stored and protected? This section identifies common ways of answering these questions, as well as discussing their strengths and weaknesses. As technology advances, additional techniques will be developed, and as such, the list below is not complete nor should be considered mandatory.

Consider common answers to the three important questions above. First, how do file encryption systems generate symmetric keys? A simple method is to derive the key from a password as described in [PKCS-5]. In this case, the security of the system depends on the randomness of the password; normally, passwords do not contain enough randomness to be used for generating keys (i.e., they can be guessed relatively easily). It is preferable to utilize a good random bit generator within the system to generate keys. Good random bit generation schemes can be found in NIST Special Publication 800-90 [SP 800-90]. Next, consider the question of how to protect the keys. When the key is derived from a password, the security of the key is dependent upon the strength (i.e., the randomness and secrecy) of the password used. A standard dictionary attack can often recover weak passwords, so a strong password is vital for the security of this type of system. There is a great deal of effort underway by the Trusted Computing Group (TCG) to develop secure storage of keys on computers. As this effort continues to mature, the Trusted Platform Module chip, through its key cache management, offers another format for protecting keys used in EFS.

Next, consider where these keys will be stored. If random keys need to be stored, they could simply be stored on the computer itself or on a hardware token. Alternatively, the key could be split into two (or more) key components with, for example, one component stored on a hardware token and the other key component stored on the computer itself. If a split key is employed, the method used to combine the key components is important; performing an XOR operation on equal length key components is better than simply concatenating the components. Common hardware tokens include PCMCIA cards and smartcards. The advantage to using a hardware token is that if the user stores the hardware token away from the computer, and the key is split between the token and the computer, an adversary needs to recover both pieces of hardware to recover the key. Additional security may be provided by encrypting the key splits, perhaps by using a password.

There are many permutations of answers to the questions above. Four examples of how these questions can be answered will be considered, along with the pros and cons of each system. It is important to consider the specific environment in which the File Encryption System will be used, as that will usually point to a specific type of system that is preferable for that case.

The first example that will be considered is a file encryption system that uses a single symmetric key to encrypt every file on the system. This single key is generated using PKCS #5 from a user's password.

The second example is a system that utilizes per-file encryption keys, which are stored on the hard disk, encrypted by a key encryption key. The key encryption key (which is also used to decrypt each file encryption key) is securely stored on the hard drive (e.g., using the Trusted Platform Module [TPM]).

The third example is a system that utilizes per-file encryption keys that are split into two key components that will be XORed to recreate the key, with one key component stored on a hardware token and the other component derived from a password (e.g., using PKCS-5 to derive the key).

The fourth example uses per-file encryption keys, which are encrypted under the file owner's asymmetric private key as previously described. This system is common in current file encryption packages, while the previous three are extreme to show more clearly the pros and cons of the systems.

Method	Pros	Cons
Example 1: PKCS #5	- Least expensive solution. - Utilizing a strong password can result in reasonable security.	- Less secure because the security is dependent only on the strength of the password.
Example 2: Key Encryption Key	- Secure storage directly in the computer. - Secure from external software attack and physical threat.	- Relatively new technology. - Keys are stored on the same computer as the file.
Example 3: Hardware Token	- Splits key. - Requires two hardware pieces to decrypt. - Highly secure if implemented correctly. - If the files or token are lost, the files will stay secure.	- More expensive. - If the token is lost, the files cannot be decrypted.
Example 4: Asymmetric user owned Key Encryption Key	-No plaintext keys stored in the computer. - Efficient file sharing. - Highly secure if token is used. - Compromise of a user's private key compromises only the user's files.	-Requires the user to manage their own RSA key pair. - Requires either a user password or a user token.

9.2 Security and Compliance Issues

1. Any encrypted file system **shall** employ approved cryptography if it is to be used for the protection of Federal government information.
2. Keys derived from passwords **shall** use strong passwords to maximize the difficulty of an off-line exhaustion attack [SP 800-118] [39]

9.3 Recommendations for Procurement Officials

The following recommendations are for any individual(s) that make a purchasing decision for acquiring an EFS component.

1. An encrypted file system **shall** include a data recovery capability (e.g., master administrator password) so that the data is not lost in the event that a user forgets his password or the user is unavailable. Data recovery is vital in this type of system.

2. Any EFS system that derives keys from passwords **should** have the capability of enforcing the use of strong passwords.Error! Bookmark not defined.
3. To increase the security of encrypted file systems, the system **should** use a hardware token for storage of the key or the Trusted Platform Module (TPM) for storage of the keys.

9.4 Recommendations for System Installers

The system installer is the individual(s) that installs EFS components and performs the initial configuration of the components.

1. When an EFS system that utilizes passwords for security is installed, the installer **shall** require that strong passwords be enforced by the EFS.Error! Bookmark not defined. This maximizes the difficulty of an off-line exhaustion attack.
2. The system installer **should** ensure that the database of keys is protected by encryption to ensure the security of the system. In addition, if the key is split by the EFS system, the key component stored on a hardware token **should** be protected.

9.5 Recommendations for System Administrators

The system administrator is the individual that manages the EFS on a day-to-day basis and interacts with the end user.

1. The system administrator **shall** ensure that the organization's security policy is enforced.
2. Key recovery procedures **shall** be in place to ensure that users can recover their data if they lose their authentication information (password, token data, etc). A method for data recovery personnel to authenticate these users **should** be in place prior to the recovery of the user's keys or files.
3. If the EFS includes a master administrator password for use in data recovery by the system administrator, the system administrator **shall** utilize a strong password.
4. System administrators **shall** provide training and security guidance to the end users of the system that, at a minimum, focuses on passwords, data recovery procedures, and user configuration of their system to utilize the authentication features within the system.
5.

9.6 Recommendations for End Users

An end user is the individual that uses the EFS to secure and share their information.

1. If user-selected passwords are used within the product, end users **shall** utilize strong passwords to maximize the difficulty of an off-line exhaustion attack.

2. End users **shall** follow guidance provided by the system administrator regarding the use of an Encrypted File System product.
3. End users **shall** inform a system administrator if they have lost their hardware token or forgotten their password.

10 Secure Shell (SSH)

(Topic under consideration)

11 IEEE 802.1x

(Topic under consideration)

Appendix A: Glossary

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

Term	Definition
Access control	Restricts access to resources only to privileged entities.
Access Control Mechanism	A method for restricting access to some resource.
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 or 3) specified in a list of NIST approved security functions.
Archive	See Key management archive.
Asymmetric key algorithm	See Public key cryptographic algorithm.
Authentication	A process that establishes the origin of information, or determines an entity's identity.
Authentication code	See Message Authentication Code.
Authorization	Access privileges granted to an entity; conveys an "official" sanction to perform a given security function or activity.
Availability	Timely, reliable access to information by authorized entities.
Backup	A copy of information to facilitate recovery, if necessary.
CBC-MAC	A mode of operation for block cipher algorithms.
Certificate	See public key certificate.
Certification authority	The entity in a Public Key Infrastructure (PKI) that is responsible for issuing certificates, and exacting compliance to a PKI policy.
Checksum	A method used to protect the integrity of data by detecting errors in that data.
Ciphertext	Data in its encrypted form.
Compromise	The unauthorized disclosure, modification, substitution or use of sensitive data (e.g., keying material and other security related information).

Confidentiality
The property that sensitive information is not disclosed to unauthorized entities.

Contingency plan
A plan that is maintained for disaster response, backup operations, and post-disaster recovery to ensure the availability of critical resources and to facilitate the continuity of operations in an emergency situation.

Cryptographic Boundary
An explicitly defined continuous perimeter that establishes the physical bounds of a cryptographic module and contains all hardware, software, and/or firmware components of a cryptographic module.

Cryptographic Hash Function
See Hash function.

Cryptographic key (key)
A parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce or reverse the operation, while an entity without knowledge of the key cannot. Examples include:

1. the transformation of plaintext data into ciphertext data,
2. the transformation of ciphertext data into plaintext data,
3. the computation of a digital signature from data,
4. the verification of a digital signature,
5. the computation of an authentication code from data,
6. the verification of an authentication code from data and a received authentication code,
7. the computation of a shared secret that is used to derive keying material.

Cryptographic module
The set of hardware, software, and/or firmware in which approved security functions are implemented; a cryptographic module is contained within a 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 integrity
A property whereby data has not been altered in an unauthorized manner since it was created, transmitted or stored. In this Recommendation, the statement that a cryptographic algorithm "provides data integrity" means that the algorithm is used to detect unauthorized alterations.

Decryption
The process of changing ciphertext into plaintext using a cryptographic algorithm and key.

DES
The Data Encryption Standard that was specified in FIPS 46 (now withdrawn).

Digital signature

The result of a cryptographic transformation of data that, when properly implemented, provides the services of:

1. origin authentication,
2. data integrity, and
3. signer non-repudiation.

Distribution

See key distribution.

Encryption

The process of changing plaintext into ciphertext using a cryptographic algorithm and key.

Entity

An individual (person), organization, device or process.

Hash algorithm

See Hash function.

Hash function

A function that maps a bit string of arbitrary length to a fixed length bit string. Approved hash functions satisfy the following properties:

1. (One-way) It is computationally infeasible to find any input that maps to any pre-specified output, and
2. (Collision free) It is computationally infeasible to find any two distinct inputs that map to the same output.

Hash value

The result of applying a hash function to information.

Hash-based message authentication code (HMAC)

A message authentication code that uses an approved keyed-hash function.

Identifier

A bit string that is associated with a person, device or organization. It may be an identifying name, or may be something more abstract (for example, a string consisting of an IP address and timestamp), depending on the application.

Initialization vector (IV)

A vector used in defining the starting point of a cryptographic process within a cryptographic algorithm.

Integrity

The property that sensitive data has not been modified or deleted in an unauthorized and undetected manner. In this Recommendation, the statement that a cryptographic algorithm "provides integrity" means that the algorithm is used to detect unauthorized modifications or deletions.

Key

See cryptographic key.

Key agreement

A key establishment scheme whose resultant keying material is a function of information contributed by two or more participants, so that no party can predetermine the value of the keying material.

Key Bundle

A set of keys used during one operation, typically a TDEA operation.

Key component

A parameter used in conjunction with other key components in an approved security function to form a plaintext cryptographic key or perform a cryptographic function.

Key distribution

The transport of a key and other keying material from an entity that either owns or generates the key to another entity that is intended to use the key.

Key encrypting key

A cryptographic key that is used for the encryption or decryption of other keys.

Key establishment

A function 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.

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 function in the lifecycle of keying material; a repository containing keying material of historical interest.

Key pair

A public key and its corresponding private key; a key pair is used with a public key algorithm.

Key recovery

Mechanisms and processes that allow authorized entities to retrieve keying material from key backup or archive.

Key revocation

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

A key establishment procedure whereby one entity (the sender) selects a value for secret keying material and then securely distributes that value to another party (the receiver).

Key update

A function performed on a cryptographic key in order to compute a new, but related, key.

Key wrapping

A method of encrypting keys (along with associated integrity information) that provides both confidentiality and integrity protection using a symmetric key.

Keying material

The data (e.g., keys and IVs) necessary to establish and maintain cryptographic keying relationships.

Message Authentication Code (MAC)

A cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modifications of data.

Nonce

A time-varying value that has, at most, a negligible chance of repeating. For example, a nonce could be a random value that is generated anew for each instance of a nonce, a timestamp, a sequence number, or some combination of these.

Non-repudiation

A service that is used to provide assurance of the integrity and origin of data in such a way that the integrity and origin can be verified by a third party as having originated from a specific entity in possession of the private key of the claimed signatory.

Owner

For an asymmetric key pair, the entity that owns the private key, whether that entity generated the key pair or a trusted party generated the key pair for the entity.

In Encrypted File Systems, the file owner has control over the file and grants access of the file to others. A file may have one or more owners. An owner could be an individual or a group of individuals or processes

Password

A string of characters (letters, numbers and other symbols) that are used to authenticate an identity or to verify access authorization.

Payload

A part of the data stream representing the user information and user overhead in a communication.

Plaintext

Intelligible data that has meaning and can be understood without the application of decryption.

Private key

A cryptographic key, used with a public key cryptographic algorithm, that is uniquely associated with an entity and is not made public. In an asymmetric (public) cryptosystem, the private key is associated with a public key. Depending on the algorithm, the private key may be 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.

Protocol

A special set of rules used by two or more entities that describe the message order and data structures for information exchanged between the entities.

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, depending on the algorithm, may be used to: 1. Verify a digital signature that is signed by the corresponding private key, 2. Encrypt data that can be decrypted by the corresponding private key, or 3. Compute a piece of shared data.
Public key (asymmetric) cryptographic algorithm	A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that determining the private key from the public key is computationally infeasible.
Public key certificate	A set of data that uniquely identifies an entity, contains the entity's public key and possibly other information, and is digitally signed by a trusted party, thereby binding the public key to the entity.
Public Key Infrastructure (PKI)	A framework that is established to issue, maintain and revoke public key certificates.
Reconstitute	Rebuilding a service provider (possibly with a different infrastructure) using previously saved security information and/or key material rather than simply restarting a service from a backup.
Rekey	A new key replaces another key; the “value” of the new key is entirely independent of the “value” of the old key.
Relying party	An individual or organization that relies on the certificate and the CA that issued the certificate to verify the identity of the user; the validity of the public key, associated algorithms and any relevant parameters; and the user’s possession of the corresponding private key.
Secret key	A cryptographic key that is used with a secret key (symmetric) cryptographic algorithm that is uniquely associated with one or more entities and is not made public. The use of the term “secret” in this context does not imply a classification level, but rather implies the need to protect the key from disclosure.
Security association	A relationship between two network entities in which security information is shared and used to support secure communication between the two entities.
Security services	Mechanisms used to provide confidentiality, data integrity,

authentication or non-repudiation of information.

Self-signed certificate
A public key certificate whose digital signature may be verified by the public key contained within the certificate. The signature on a self-signed certificate protects the integrity of the data, but does not guarantee authenticity of the information. The trust of self-signed certificates is based on the secure procedures used to distribute them.

Shall
This term is used to indicate a requirement of a Federal Information Processing Standard (FIPS) or a requirement that must be fulfilled to claim conformance to this Recommendation. Note that **shall** may be coupled with **not** to become **shall not**.

Shared secret
A value that is generated during a key agreement scheme; the shared secret is typically used to derive keying material for a symmetric key algorithm.

Should
This term is used to indicate a very important requirement. While the "requirement" is not stated in a FIPS, ignoring the requirement could result in undesirable results. Note that **should** may be coupled with **not** to become **should not**.

Signature verification
Uses a digital signature algorithm and a public key to verify a digital signature.

Symmetric key
A single cryptographic key that is used with a secret (symmetric) key algorithm.

Symmetric key algorithm
A cryptographic algorithm that uses one shared key, a secret key.

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.

Triple DES/Triple DEA (TDEA)
Triple Data Encryption Algorithm [specified in SP 800-67]

3-TDEA
Three key TDEA as specified in SP 800-67.

Unauthorized disclosure
An event involving the exposure of information to entities not authorized access to the information.

User name (in a certificate)
The name of the party authorized to use the private key associated with the public key in the certificate; the subject of the certificate.

User registration
A function in the lifecycle of keying material; a process whereby an entity becomes a member of a security domain.

X.509 public key certificate

The public key for a user (or device) and a name for the user (or device), together with some other information, rendered unforgeable by the digital signature of the certification authority that issued the certificate, encoded in the format defined in the ISO/ITU-T X.509 standard.

Appendix B: Acronyms

AES	Advanced Encryption Standard
AH	Authentication Header
AS	Authentication Server
CA	Certificate Authority
CBC	Cipher Block Chaining
CBC-MAC	Cipher Block Chaining Method Authentication Code
CMVP	Cryptographic Module Validation Program
COTS	Commercial Off-the-Shelf
CRL	Certificate Revocation List
CTR	Counter Mode
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSA	Digital Signature Algorithm
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EFS	Encrypted File System
ESP	Encapsulating Security Protocol
FEK	File Encryption Key
GCM	Galois Counter Mode
GMAC	Galois Message Authentication Code
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
ICV	Integrity Check Value
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IV	Initialization Vector
KDC	Key Distribution Center
KMF	Key Management Facility
KMM	Key Management Message
KSK	Key Signing Key
KWK	Key Wrapping Key
LDAP	Lightweight Directory Access Protocol

MAC	Message Authentication Code
MD5	Message-Digest Algorithm 5
NSEC	Next Secure
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
OMB	Office of Management and Budget
OTAR	Over The Air Rekeying
PIN	Personal Identification Number
PKCS#1	Public Key Cryptography Standard #1
PKI	Public Key Infrastructure
PRF	Pseudorandom Function
PVM	Path Validation Module
RA	Registration Authority
RFC	Request for Comment
RR	Resource Record
RSA	Rivest, Shamir, and Adleman.
SA	Security Association
SEP	Secure Entry Point
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transfer Protocol
S/MIME	Secure/Multipart Internet Mail Extensions
SSL	Secure Socket Layer
TDEA	Triple Data Encryption Algorithm
TEK	Traffic Encryption Key
TCG	Trusted Computing Group
TGS	Ticket Granting Service
TLS	Transport Layer Security
TPM	Trusted Platform Module
TS	Target Server
TSIG	Transaction Signature
URL	Uniform Resource Locator
VPN	Virtual Private Network
XOR	Exclusive-Or operation
ZSK	Zone Signing Key

Appendix C: A Word to Novice End Users

Cryptographic keys are frequently categorized by the algorithms they are used with, the operations they are used to perform and by the number of times they can be used. Keys will either be asymmetric and used with an asymmetric algorithm, or symmetric and used with a symmetric algorithm. Asymmetric keys are generated as key pairs: a *private key* that must be kept secret, and a related *public key*, that may not be a secret. In general, the private key performs one operation, for example, a digital signature, and the public key does the complementary operation, in this case, signature verification. In the case of symmetric keys, end users need to treat the key as a shared secret value and use the same value for complementary cryptographic functions, such as encryption and decryption.

Some asymmetric keys are static and intended for long-term use, while others are ephemeral and expire after use with one message or session. The public key of a static asymmetric key pair is often provided in a public key certificate, while an ephemeral public key is not. Although the concept of static vs. ephemeral also applies to symmetric keys, a short term symmetric key is often called a session key, rather than an ephemeral key. No specific terminology is used for a long-term symmetric key. An application or protocol may be supported by some combination of such keys.

A PKI is the foundation of many current key management processes and is used in many of the protocols or applications described in this Recommendation, as well as other security protocols and applications. Some understanding of the role of a PKI and *public key certificates* in key management is very helpful to setting up security protocols or applications properly. A long-term, or "static" public key is generally combined with the name of the key's "owner" in an electronic document called a public key certificate. While certificates can be self-issued and signed (that is, the party that created the key pair can sign his name and the public key with the corresponding private key), most certificates are digitally signed with the private key of a trusted entity called a Certification Authority.

An end user may typically has one or more certificates, and may have different certificates for different applications, e.g., for e-mail and for authentication to web sites. As Federal Personal Identity (PIV) cards are issued to Federal employees and contractors, most Federal users will have a personal smart card that contains one or more personal certificates issued to them by their agency. Other specific applications may require "soft" certificates, usually kept on the user's computer, and possibly issued by commercial CAs. For example, browser products, such as Internet Explorer, Firefox or Safari, typically implement a mechanism to generate key pairs, send the public keys to a CA and return a public key certificate for that key to the browser. The certificates are then kept in a user certificate store on the user's computer, which can be managed in a manner similar to a root certificate store (see Section 2). The process and interfaces for generating keys, and for requesting, downloading and installing certificates are specific to both the individual user client product and, sometimes, to the CA itself; however, the websites of CAs often have pages that "walk" the user through the key generation and certificate issuance process for the common browser products. Users can share certificates via email or public key infrastructures, smart cards or other memory tokens.

Similarly, secure web servers have TLS/SSL server certificates, with certain specific characteristics; note that, although Federal users are required to use TLS rather than SSL, SSL is really just an earlier version of TLS, and the certificates are identical). The *Subject Name* in these certificates follows specific rules so that the domain name of the server is included in the Subject Name field of the certificate. Commercial CAs sell SSL certificates, and, where it is important to reach a general population of non-government users, it may be desirable to get an SSL certificate from a CA that has its root certificate widely distributed "out of the box" in the certificate stores of Microsoft Windows, Macintosh OS X, and the various Mozilla browsers. This will allow most users to verify the server certificate. However, it is important to review the certificate policy and choices that may be available from the CAs selling SSL certificates for use on Federal agency web servers in order to ensure that the certificates meet the requirements stated herein and in SP 800-57.

Appendix D: References

COMMON “X.509 Certification Policy for the U.S. Federal PKI Common Policy Framework”, Federal Public Key Infrastructure Policy Authority, Version 3647-1.3, December 2007.

COMMON PROF “X.509 Certificate and Certificate Revocation List (CRL) Extensions Profile for the Shared Secret Providers (SSP) Program” Federal PKI Policy Authority Shared Service Provider Working Group, January 2008

DENN Dorothy E. Denning and Giovanni Maria Sacco, “Time-stamps in Key Distribution Protocols,” Communications of the ACM, Vol. 24 (8), p. 533-536, August 1981.

DES ANSI X3.106, "American National Standard for Information Systems-Data Link Encryption," American National Standards Institute, 1983.

FIPS 46-3 “Data Encryption Standard (DES)”, Federal Information Processing Standard, National Institute of Standards and Technology, U.S. Department of Commerce, October 25, 1999.

FIPS 180-2 “Secure Hash Standard (SHS)”, Federal Information Processing Standard, National Institute of Standards and Technology, US Department of Commerce, August 2002.

FIPS 180-3 “Secure Hash Standard (SHS)”, Federal Information Processing Standard, National Institute of Standards and Technology, US Department of Commerce, June 2007.

FIPS 186-3 “Digital Signature Standard”, Federal Information Processing Standard, National Institute of Standards and Technology, U.S. Department of Commerce, March 2006.

FIPS 197 FIPS 197, “Advanced Encryption Standard (AES)”, Federal Information Processing Standard, National Institute of Standards and Technology, U.S. Department of Commerce, November 2001.

FIPS 201 “Personal Identity Verification (PIV) of Federal Employees and Contractors”, Federal Information Processing Standard, National Institute of Standards and Technology, U.S. Department of Commerce, March 2006.

FPKI PROF “Federal Public Key Infrastructure (PKI) X.509 Certificate and CRL Extensions Profile” Booz-Allen & Hamilton and National Institute of Standards and Technology, October 2005.

KEA "SKIPJACK and KEA Algorithm Specifications", Version 2.0, Department of Defense, June 1998.

NEED Roger M. Needham and Michael D. Schroeder, “Using Large Networks of Computers,” Communications of the ACM, Vol. 21 (12), pp. 993-999, December 1978.

NEUM B. Clifford Neuman and Theodore Y. Ts’o, “An Authentication Service for Computer Networks,” IEEE Communications Magazine, Vol. 32 (9), p. 33-38, September 1994.

OMB 04-04 “MEMORANDUM TO THE HEADS OF ALL DEPARTMENTS AND AGENCIES RE: E-Authentication Guidance for Federal Agencies”, M-04-04, December 16, 2003

OTAR “Project 25 Digital Radio Over The Air Rekeying (OTAR) Protocol”, TIA/EIA-102.AACA-2001, April 2001.

OTAR1 “Project 25 Digital Radio Over The Air Rekeying Protocol: Addendum 1 - Key Management Security Requirements for Type 3 Block Encryption Algorithms”, TIA/EIA 102.AACA-1, November 2002.

PKCS-5 "PKCS #5 v2.0: Password-Based Cryptography Standard", RSA Laboratories, March 25, 1999.

PKCS-7 Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, IETF, March 1998.

PKCS-10 “PKCS #10 v1.7: Certification Request Syntax Standard” RSA Laboratories May 2000

RFC 791 “Internet Protocol” RFC 791, September 1981.

RFC 1034 P. Mockapetris “Domain Names – Concepts and Facilities” RFC 1034 November 1987.

RFC 1035 P. Mockapetris “Domain Names – Implementation and Specification” RFC 1035 November 1987

RFC 1510 J. Kohl and C. Neuman “The Kerberos Network Authentication Service (V5)”, RFC 1510, September 1993.

RFC 2045 N. Freed and N. Borenstein, “Multipurpose Internet Mail Extensions (MIME) Part One:Format of Internet Message Bodies” RFC 2045, November 1996.

RFC 2046 N. Freed and N. Borenstein, “MIME Part 2: Media Types” RFC 2046, November 1996.

RFC 2047 K. Moore, “MIME Part Three: Message Header Extensions for Non-ASCII Text” RFC 2047, November 1996.

RFC 2048 N. Freed, J. Klensin , J. Postel, “MIME Part Four: Registration Procedures” RFC 2048, November 1996.

RFC 2049 N. Freed and N. Borenstein, “MIME Part Five: Conformance Criteria and Examples” RFC 2049, November 1996

RFC 2246 T. Dierks and C. Allen, “The TLS Protocol Version 1.0”, RFC 2246, January 1999.

RFC 2401 S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

RFC 2402 S. Kent and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

RFC 2404 C. Madson and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.

RFC 2405 C. Madson and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm with Explicit IV", RFC 2405, November 1998.

RFC 2406 S. Kent, and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

RFC 2407 D. Piper, "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998.

RFC 2408 D. Maughan, et. al., "The Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

RFC 2409 Harkins, D., Carrel, D., “The Internet Key Exchange (IKE)”, RFC 2409, November 1998.

RFC 2410 R. Glenn and S. Kent, “The NULL Encryption Algorithm and its Use with IPsesc”, RFC 2410, November 1998.

RFC 2451 R. Pereira, and R. Adams, "The ESP CBC-Mode Cipher Algorithms", Internet Advisory Board, Internet Engineering Task Force, RFC 2451, November 1998.

RFC 2460 S. Deering and R. Hinden. “Internet Protocol, Version 6 (IPv6) Specification” RFC 2460 December 1998.

RFC 2560 M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP”, RFC 2560, June 1999.

RFC 2631 E. Rescorla, “Diffie-Hellman Key Agreement Method”, RFC 2631, June 1999.

RFC 2634 P. Hoffman “Enhanced Security Services for S/MIME” RFC 2634, June 1999.

RFC 3394 R. Housley and J. Schaad, “Advanced Encryption Standard (AES) Key Wrap Algorithm”, RFC 3394, September 2002.

RFC 3447 J. Jonsson, and B. Kaliski, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1”, RFC 3447, February 2003.

RFC 3566 S. Frankel and H. Herbert, “The AES-XCBC-MAC-96 Algorithm and its Use with IPsec”, RFC 3566, September 2003.

RFC 3602 S. Frankel, R. Glenn, and S. Kelly, “The AES-CBC Cipher Algorithm and its Use with IPsec”, RFC 3602, September 2003.

RFC 3686 R. Housley, “Using Advanced Encryption Standard (AES) Counter Mode with IPsec Encapsulating Security Payload (ESP)”, RFC 3686, January 2004.

RFC 3851 B. Ramsdell, “S/MIME Version 3 Message Specification”, RFC 3851, June 1999.

RFC 3852 R. Housley, “Cryptographic Message Syntax (CMS)”, RFC 3852, July 2004.

RFC 3962 K. Raeburn, “Advanced Encryption Standard (AES) Encryption for Kerberos 5”, RFC 3962, February 2005.

RFC 4033 R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “DNS Security Introduction and Requirements”, RFC 4033, March 2005.

RFC 4034 R. Arends, R. Austein, M. Larson, D. Massey, D., and S. Rose, “Resource Records for the DNS Security Extensions”, RFC 4034, March 2005.

RFC 4035 R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, “Protocol Modifications for the DNS Security Extensions”, RFC 4035, March 2005.

RFC 4106 J. Viega and D. McGrew, “The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)”, RFC 4106, June 2005.

RFC 4109 P. Hoffman, “Algorithms for Internet Key Exchange version 1 (IKEv1), RFC 4109, May 2005.

RFC 4120 Neuman, C., Yu, T., Hartman, S., Raeburn, K., “The Kerberos Network Authentication Service (V5)”, RFC 4120, July 2005.

RFC 4210 Adams, C., Farrell, S., Kause, T., Mononen, T., “Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)”, RFC 4210, September 2005.

RFC 4301 S. Kent and K. Seo, “Security Architecture for the Internet Protocol", RFC 4301, December 2005.

RFC 4302 S. Kent, "IP Authentication Header", RFC 4302, December 2005.

RFC 4303 S. Kent, "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

RFC 4306 C. Kaufman, ed., “Internet Key Exchange (IKEv2) Protocol”, RFC 4306, December 2005.

RFC 4307 J. Schiller, "Cryptographic Algorithms for use in the Internet Key Exchange Version 2 (IKEv2)" RFC 4307, December 2005.

RFC 4308 P. Hoffman, "Cryptographic Suites for IPsec", RFC 4308, December 2005.

RFC 4309 R. Housley, “Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)”, RFC 4309, December 2005.

RFC 4434 P. Hoffman, “The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)”, RFC 4434, February 2006.

RFC 4543 D. McGrew, and J. Viega, “The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH”, RFC 4543, May 2006.

RFC 4556 L. Zhu and B. Tung, “Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)”, RFC 4556, June 2006.

RFC 4718 P. Eronen and P. Hoffman, “IKEv2 Clarification and Implementation Guidelines”, RFC 4718, October 2006.

RFC 4753 D. Fu and J. Solinas, “ECP Groups for IKE and IKEv2”, RFC 4753, January 2007.

RFC 4754 D. Fu and J. Solinas, “IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)”, RFC 4754, January 2007.

RFC 4835 V. Manral, “Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)”, RFC 4835, April 2007.

RFC 4868 S. Kelly and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, May 2007.

RFC 4869 L. Law and J. Solinas, “Suite B Cryptographic Suites for IPsec”, RFC 4869, May 2007.

RFC 5008 R. Housley, and J. Solinas, “Suite B in Secure/Mulitpurpose Internet Mail Extensions (S/MIME) “, RFC 5008, September 2007.

RFC 5019 A. Deacon and R. Hurst, “ The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments”, RFC 5019, September 2007.

RFC 5035 P. Hoffman, “Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility”, RFC 5035, August 2007.

RFC 5155 B. Laurie, G. Sisson, R. Arends and D. Blacka. “DNSSEC Hashed Authenticated Denial of Existence”. RFC 5155, March 2008.

RFC 5246 T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

RFC 5272 M. Myers and J. Schaad, “Certificate Management Messages over CMS “, RFC 5272, June 2008.

RFC 5280 D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”,RFC 5280, May 2008.

SEC1 Standards for Efficient Cryptography Group, “Elliptic Curve Cryptography”, 2000. [See http://www.secg.org/collateral/sec1.pdf.]

SP 800-15 W. Burr, D. Dodson, N Nazario, and W. Polk, “MISPC Minimum Interoperability Specifications for PKI Components, Version 1”, National Institute of Standards and Technology, NIST Special Publication 800-15, September 1997.

SP 800-32 D. Kuhn, V. Hu, W. Polk, and S. Chang, “Introduction to Public Key Technology and the Federal PKI Infrastructure”, National Institute of Standards and Technology, NIST Special Publication 800-32, February 2001.

SP 800-38A M. Dworkin, “Recommendations for Block Cipher Modes of Operation” National Institute of Standards and Technology, NIST Special Publication 800-38A December 2001

SP 800-56A E. Barker, D. Johnson and M. Smid, “Recommendations for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised)” National Institute of Standards and Technology, NIST Special Publication 800-56A March 2007.

SP 800-67 W. Barker, “Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher”, National Institute of Standards and Technology, NIST Special Publication 800-67, May 2004.

SP 800-78-1 W. Polk, D. Dodson and W. Burr, “Cryptographic Algorithms and Key Sizes for Personal Identity Verification” National Institute of Standards and Technology, NIST Special Publication 800-78-1, August 2007.

SP 800-81 R. Chandramouli and S. Rose, “Secure Domain Name System (DNS) Deployment Guide”, National Institute of Standards and Technology, NIST Special Publication 800-81, May 2006.

SP 800-90 E. Barker and J. Kelsey, “Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)”, NIST Special Publication 800-90, March 2007.

TPM TCG TPM Specification Version 1.2 Revision 103 including Design Principles, Structures of the TPM, and Commands.

X9.62 ANSI X9.62, "Public key cryptography for the financial services industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", November 2005.