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 [SP 800-57 Part 1]. *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 provided,
- 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, and
- General recommendations for people making purchasing decisions, 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

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

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 time, 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 [SP 800-57 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 as discussed in this Recommendation.

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 cryptography 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., Internet Engineering Task Force (IETF) Requests for Comment (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 that do not provide adequate security (e.g., Data Encryption Standard (DES) or RC2), and this document will say they **shall not** be used. Similarly, there may be optional-to-implement algorithms that have greater security (e.g., Advanced Encryption Standard (AES)), and this document may say that these algorithms **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 communicating parties or other system limitations. When servers dictate the algorithms used, the server may select the algorithms that optimize overall system performance, rather than the ones 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 e-mail 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., Transport Layer Security (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 Parts 1 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 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 Certificate Authority (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:

[2] The DES algorithm and the 512-bit RSA key size do not provide adequate security (see [SP 800-57 Part 1]).

[3] A 512-bit RSA key does not provide an acceptable security strength (see [SP 800-57 Part 1]).

- 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 **shall not** be used for multiple cryptographic purposes. For example, the same key **shall not** be used to generate a digital signature and to establish other keying material (see [SP 800-57 Part 1, Sec. 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 [SP 800-57 Part 1]. 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 e-mail package before upgrading the cryptographic module. Hence, for a period of time, the system may be running with an e-mail 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 upgrades can be made. Organizations and system administrators must determine the approach for transitioning to stronger security mechanisms within a system.

[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.

2 Public Key Infrastructure (PKI)

2.1 Description

A PKI is the most common key management approach for the distribution of public keys. As described in SP 800-57 Part 1, *Recommendation for Key Management, Part 1: General* [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 (where appropriate), 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 e-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. CA certificates 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, can maintain their own root certificate stores and also have a feature that allows them to share a root certificate store with other applications.[6]

[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 have their own root certificate stores, and they also can share a root certificate store from Mozilla's Network Security Services (NSS) utility.

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 224-bit elliptic curve cryptography, and either, 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, validity (starting and expiring times), 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 some 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.

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:

- An Online Certificate Status Protocol (OCSP) responder [RFC 6960]. An OCSP responder is a trusted system, and provides signed status information for a given CA, 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 **shall** be a formal relationship between the CA and OCSP responder (e.g., a contract).
- An alternative method is to use a certificate revocation list, or CRL. An X.509 CRL contains a list of certificates issued by a CA that have been revoked, indicates when they were revoked, and may include the reason for revocation [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. The distribution of CRLs by web servers has become more common recently.

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. A PKI **should** include a key recovery services system.

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

[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*.

large, general-purpose PKIs, such as the Federal PKI. For PKIs requiring less 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

Table 2-1 below summarizes the recommended key sizes for key pairs used by PKI users and infrastructure components. The PKI uses the term *digital signature key* to refer to a private signature key or public signature verification key (as defined in Part 1) that provides a non-repudiation service. The term *authentication key* is used by the PKI to refer to a private authentication key or public authentication key as defined in Part 1. Note that both a digital signature key and an authentication key are used with a digital signature algorithm.

The dates in this table are consistent with those that appear in Part 1, where:

- A digital signature key is as defined above.
- A key establishment key is an asymmetric key pair used to provide key agreement or key transport, and
- A CA and OCSP responder signing key is an asymmetric key pair used to sign and verify certificates.

Approved algorithms and key sizes specified in **Table 2-1** and certificate expiration dates are two different things. These algorithms and key sizes are approved for use beyond the year 2013. However, a digital signature or key establishment certificate may expire at any time, depending on the organization's security policy.

Table 2-1: Recommended Algorithms and Key Sizes

Key Type	Algorithms and Key Sizes
Digital Signature keys used for authentication (for Users or Devices)	RSA (2048 bits) ECDSA (Curve P-256)
Digital Signature keys used for non-repudiation (for Users or Devices)	RSA (2048 bits) ECDSA (Curves P-256 or P-384)
CA and OCSP Responder Signing Keys	RSA (2048 or 3072bits) ECDSA (Curves P-256 or P-384)
Key Establishment keys (for Users or Devices)	RSA (2048 bits) Diffie-Hellman (2048 bits) ECDH (Curves P-256 or P-384)

Note that some approved algorithms and key sizes, such as DSA 2048, are omitted to enhance interoperability. RSA and ECDSA, which are included in Table 2-1 above, have been widely deployed in PKIs. Therefore, they are recommended for use to enhance interoperability. However, DSA (2048 and 3072) as specified in [FIPS 186-4] are allowed as long as the required security strength is satisfied. For ECDSA, only the two elliptic curves listed in Table 2-1 above of the elliptic curves are recommended for use in PKIs for digital signatures [FIPS 186-4]. Similarly, Elliptic Curve Diffie-Hellman (ECDH) is recommended to support key establishment, rather than Elliptic Curve MQV.

While Table 2-1 is focused on the strength of the public key contained in a certificate, the strength of the digital signature on a certificate itself is equally important. The signature security strength also reflects the security strength of the hash algorithm, and possibly the padding scheme[8], in addition to the strength of the private key used to generate the signature. Table 2-2 below summarizes the recommended algorithms, key sizes, hash functions, and padding schemes for signing certificates and CRLs by CAs, and OCSP status messages by OCSP responders.

[8] RSA has two padding schemes used in the PKI: PKCS #1 v1.5, and PSS. The security strength of a digital signature generated using ECDSA is not affected by a padding scheme.

Table 2-2: Digital Signature Recommendations for CAs and OCSP Responders

Public Key Algorithms and Key Sizes	Hash Algorithms	Padding Scheme
RSA (2048 or 3072 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 the same cryptographic strength. Table 2-3 below shows preferred combinations for user keys.

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 [SP 800-67]. Components that support P-384 elliptic curve keys and the SHA-384 algorithm **shall** support AES-256.

Table 2-3: Recommended Combinations for the Recommended Algorithms and Key Sizes

Authentication Key Type	Signature Key	Key Establishment Key
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

[9] In general, protocols and applications are designed to use cryptographic algorithms from one mathematical family. For example, applications that encounter certificates with ECDSA digital signatures would expect to use elliptic curve Diffie-Hellman for key establishment services. Users that obtain an ECDSA certificate (i.e., a certificate containing an ECDSA public key to be used for verifying digital signatures), and an RSA key establishment certificate (i.e., a certificate containing an RSA public key to be used for key establishment), for example, may find they cannot use the keys together in a single application. Other combinations of certificates are commonly used (see Table 2-3). It is advisable that users obtain authentication, signature, and key establishment certificates that are complementary to ensure that the keys can be used together in applications and protocols.

2.3 Procurement Guidance

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

2.3.1 CA/RA Software and Hardware:

1. Ensure that CA and RA software supports at least one of these protocols: the Certificate Management Protocol (CMP) [RFC 4210], Enrollment over Secure Transport (EST) [RFC 7030] and Certificate Management Using Cryptographic Message Syntax (CMC); see [RFC 5272].

2. Ensure that all CAs 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. Ensure that CAs are capable of issuing multiple certificates to users, and for all such certificates, asserting the key usage extension, including the Extended Key Usage extension.

4. Ensure that CAs are capable of including the CRL distribution points extension.

5. Ensure that CAs support the inclusion of HTTP URLs to specify the location of CRLs.

6. CAs **should** support the inclusion of LDAP to specify the location of CRLs.

7. Ensure that CAs are capable of specifying an authoritative OCSP responder in the Authority Information Access extension.

8. Each PKI has its own certificate profile, identifying certificate extensions that appear in the certificates and CRLs it issues.[10] Ensure that CAs are able to generate all mandatory extensions in the appropriate profiles. For CAs owned or operated on behalf of federal agencies, the following specific guidance applies:

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

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

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

[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.

10. Ensure that CAs 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], DSA or ECDSA. To maximize flexibility, CAs **should** support RSA and ECDSA.[12]

11. Ensure that CAs include backup and archive capabilities to support reconstitution of the CA in the event that the root key is corrupted, destroyed or lost, and it is necessary to rebuild the CA using a backup root key, rather than simply recovering the lost state of the CA. CAs **should** include backup and archive capabilities in order to establish when certificates were issued and revoked, and under whose authority.

12. Ensure that CA/RA components are 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. Ensure that OCSP responders conform to RFC 6960, *Online Certificate Status Protocol* [RFC 6960].

2. Ensure that OCSP responders are capable of processing both signed and unsigned requests and are 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. Ensure that OCSP responders are capable of processing certificate status requests and generating responses for non-error conditions as specified in [RFC 5019].

4. Where supported, the OCSP responder **should** sign the OCSP response with the algorithm and key size used to sign the certificate. Ensure that OCSP responders support at least one of the following algorithms for digitally signing response messages: RSA with PKCS#1 v1.5 padding; RSA with PSS Padding, DSA 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 RSA and ECDSA.[13]

[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] 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.

2.3.3 Cryptographic Modules

1. Ensure that Cryptographic modules for CAs, Key Recovery Servers, and OCSP responders are hardware modules validated as meeting FIPS 140-2 Level 3 or higher [FIPS 140-2].

2. Ensure that cryptographic modules for RAs are hardware cryptographic modules validated as meeting FIPS 140-2 Level 2 or higher [FIPS 140-2].

3. Ensure that relying party and user cryptographic modules are validated as meeting FIPS 140-2 Level 1 or higher [FIPS 140-2].

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.[14]

2.3.5 Relying Party Software

1. Relying party path validation
   a) Ensure that relying party implementations use RFC 5280-conformant path validation [RFC 5280].
   b) Where interoperability outside a single organization is required (e.g., a single federal agency), path validation modules **should** conform to requirements for an Enterprise path validation module (PVM), as specified in NIST Recommendation for X.509 Path Validation [X.509 Path].
   c) Where interoperability across organizations is required, path validation modules **should** conform to requirements for a Bridge-Enabled PVM, as specified in the draft *NIST Recommendation for X.509 Path Validation* [X.509 Path].
   d) Ensure that relying party implementations support CRLs or OCSP for certificate status, and **should** support both.

2. Building certificate paths
   a) Ensure that relying party implementations are able to build certification paths.
   b) At a minimum, implementations **should** support http-based certificate retrieval.
   c) Relying party implementations **should** also 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.

[14] 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.

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. Ensure that client implementations support multiple private keys and certificates for each end user 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. Ensure that client cryptographic modules are validated at FIPS 140-2 Level 1 or higher [FIPS 140-2].

3. Client implementations **should** support the certificate management protocol supported by the organization's CA.[15]

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.

2.4.1 Certificate Issuance

1. CAs **shall** be configured to ensure that certificates specify public keys with approved key sizes, valid domain parameters (if appropriate), 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 using a signing algorithm, hash function and a padding scheme (if the signing algorithm is RSA) combination specified in Table 2-2.

[15] 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.

5. For digital signature certificates, CAs **shall** sign the certificate using a digital signature process (i.e., signature algorithm, hash function and key) whose security strength is equal to or greater than the security strength of the subject public key in the certificate. For key establishment certificates, CAs may sign the certificate using a digital signature process whose security strength is less than the security strength of the subject public key in the certificate[16].
6. Generating key pairs:
    a) Users **should** generate their own digital signature key pairs.
    b) Key establishment key pairs may be generated by the user or by the PKI on the user's behalf; where required, the PKI that generated a user's key pair may retain copies of the key-establishment private key to permit key recovery.
    c) CAs **should** perform proof of possession for 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 **shall** restrict acceptance of the private key to a single cryptographic function: either user/entity authentication and verification of committed data, or key establishment.
9. All certificates **shall** include the CRL distribution-points extension to support the retrieval of status information.
10. If an OCSP responder is supported, a certificate **shall** include an appropriate URL in the Authority Information Access extension.
11. Certificates **should** be renewed before they expire and replaced if there is a change in the certificate's contents, such as the domain name or the embedded e-mail address.

2.4.2 Certificate Revocation Requests

1. CAs **should** be configured to automate revocation processing where practical:

[16] A public key certificate used for key establishment involves two keys: the subject (key establishment) public key, which is used to establish a symmetric key that will protect data, and the signing key of the Certification Authority (CA), which is used to sign the certificate. The CA's signing key needs to be secure only until the key-establishment certificate expires, but the subject (key establishment) public key needs to be secure as long as the data must be secure, which may be long after the key establishment certificate expiration date. As long as the CA's signing key is secure during the certificate's lifetime, and the certificate has been securely archived, any break of the CA signing key after the expiration of the certificate does not affect the validity of the subject (key-establishment) public key or the security that it (the subject public key) can provide. For example, if the security strength of the subject (key establishment) public key is greater than that of the CA's signing key, any break of the signing key after the subject public key is signed does not affect the security of that public key. Therefore, it is acceptable for a key transport or key agreement subject public key to be stronger than the CA key used to sign a certificate containing the key agreement or key transport public key.

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 a particular 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 of the requester.

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 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*.[17]

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, 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 as long as required.

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.

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 Common Policy Root CA or the Federal Bridge as the trust anchor.

[17] Available at http://cio.nist.gov/esd/emaildir/lists/pkits/doc00000.doc.

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 means either the person associated with a public key, or the administrator of a device associated with a public key.

1. Users **should** generate their own key pairs for digital signatures 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.

6. Users **shall** verify all CRLs before rejecting certificates in the CRLs.

3 Internet Protocol Security (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)[18], 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. Hereafter, the terms AH and ESP will be used as shorthand for messages using AH and ESP headers, respectively. 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[20], as it has many enhancements not found in the previous versions. 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, an automated 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

---

[18] See SP 800-77, *Guide to IPsec VPNs* [SP 800-77].

[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], [RFC 4303] and [RFC 4835].

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

(e.g., to rekey). This approach permits a decoupling of the key management mechanism from the 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 5996]; 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. IKEv1 is still implemented in numerous current systems, despite the fact that it is obsolete. These two versions of IKE are not interoperable. IKEv2 was designed to be more reliable and efficient than IKEv1; therefore, IKEv2 **should** be used.

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, RFC 2406	RFC 2407, RFC 2408, RFC 2409
IPsec-v3	RFC 4301	RFC 4303	RFC 4302, RFC 4303	RFC 5996

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, two cipher suites: VPN-A and VPN-B were specified [RFC 4308]. However, these two cipher suites are not NIST-approved cipher suites. Four additional cipher suites have been defined in [RFC 6379]: Suite-B-GCM-128, Suite-B-GCM-256, Suite-B-GMAC-128 and Suite-B-GMAC-256 and they are NIST-approved.

Implementers may allow the individual selection of security algorithms (i.e., rather than selecting one of the pre-specified suites of algorithms) specified in [RFC 6379], 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. Current IETF algorithm guidance is in [RFC 7321].

3.2 Security and Compliance Issues

3.2.1 Cryptographic Algorithms

Table 3-2 below gives cryptographic algorithm recommendations for use within IPsec. The algorithms that are specified for IKE are used to protect IKE's own traffic. The algorithms used in ESP and AH 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.

In Table 3-2 below, column four lists the IETF conformance requirements as specified in the RFCs by using the three IETF requirement levels: MUST, SHOULD and MAY to indicate whether the algorithm needs to be implemented. See [RFC 2119] for definitions of these requirement levels and further information on IETF Conformance language. Column five, however, states federal conformance requirements using two levels: Mandatory and Optional. Mandatory means that the feature is required to be available in an implementation, and Optional means that implementation of the technique is permitted.

Table 3-2: Cryptographic Algorithm Recommendations

Protocol	Cryptographic Service	Algorithm/Mode	IETF Requirement RFC 7321	Federal Requirement
ESP	Encryption	TDEA in CBC mode	MAY	Optional; if used, TDEA **shall** use three distinct keys
ESP	Encryption	AES with 128-bit keys in CBC mode	MUST	Mandatory
ESP	Encryption	AES-128 in Counter mode	MAY	Optional; if used, **shall** be used with integrity protection
ESP or AH	Integrity Protection	HMAC SHA1-96 (key strength **shall** be equal to or greater than 112 bits)	MUST	Mandatory

Protocol	Cryptographic Service	Algorithm/Mode	IETF Requirement	Federal Requirement
ESP or AH	Integrity Protection	HMAC SHA-256-128 (key strength **shall** be equal to or greater than 112 bits)	[RFC 4868] SHOULD	Optional
ESP	Encryption and Integrity Protection	AES-128 in Galois/Counter Mode	[RFC 4106], [RFC 4835]	Optional
ESP	Encryption and Integrity Protection	AES-128 in Counter mode with CBC-MAC	[RFC 4309], [RFC 4835] MAY	Optional
ESP or AH	Integrity Protection	AES-128 in GMAC Mode	[RFC 4543]	Optional
IKEv1 or IKEv2	Encryption	TDEA in CBC mode	MAY	Optional; if used, TDEA **shall** use three distinct keys
IKEv1 or IKEv2	Encryption	AES-128 in CBC mode	MUST	Mandatory
IKEv1 or IKEv2	Pseudo-random function	HMAC-SHA1	[RFC 4109], [RFC 4307] MUST	Mandatory
IKEv1 or IKEv2	Pseudo-random Function	HMAC-SHA-256	[RFC 4868] SHOULD	Optional
IKEv1 or IKEv2	Diffie-Hellman Group 24	2048-bit MODP	[RFC 5114]	Mandatory

Protocol	Cryptographic Service	Algorithm/Mode	IETF Requirement	Federal Requirement
IKEv1 or IKEv2	Diffie-Hellman Group 14	2048-bit MODP	[RFC 4109], [RFC 4307] SHOULD	Optional
IKEv1 or IKEv2	Elliptic Curve Diffie-Hellman	P-256 or P-384	[FIPS 186-4], [RFC 5903] MAY	Should
IKEv1 or IKEv2	Integrity	HMAC-SHA1-96 (key strength **shall** be equal to or greater than 112 bits)	[RFC 4109], [RFC 4307] MUST	Mandatory
IKEv1 or IKEv2	Integrity	HMAC-SHA256-128 key strength **shall** be equal to or greater than 112 bits)	[RFC 4868] SHOULD	Optional
IKEv1	Peer Authentication	2048-bit RSA with SHA256	[RFC 4109] SHOULD	Mandatory
IKEv1	Peer Authentication	DSA with SHA256	[RFC 4109] MAY	Mandatory
IKEv2	Peer Authentication	2048-bit RSA with SHA256	[RFC 5996] MUST	Mandatory
IKEv2	Peer Authentication	DSA with SHA256	[RFC 5996] MAY	Optional
IKEv1 or IKEv2	Peer Authentication	ECDSA-256 or ECDSA-384	[RFC 4754] MAY	Should

In [RFC 4835] and [RFC 2410], ESP provides options for NULL integrity protection or NULL encryption, which means that either no integrity protection would be applied or no encryption would be used, respectively; however, [RFC 4835] specifies that at least one of them MUST be applied. In the RFCs, NULL integrity protection (often referred to as NULL authentication) is intended for use in 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).

However, to be compliant with this Recommendation, IPsec ESP-protected traffic **shall** always be integrity-protected, either through the use of an integrity-protection algorithm, such as HMAC-SHA1-96, or through the use of a combined-mode algorithm, such as AES-128 in Galois/Counter Mode. Therefore, encrypted ESP **shall not** be used with NULL integrity-protection.

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 ([RFC 2406], [RFC 2402]).

When HMAC is used for integrity protection, the length of the ICV value is at most the size of the output value of the hash function. For example, the ICV value for HMAC-SHA1-96 and HMAC-SHA256-128 is 96 and 128 bits, respectively [RFC 4868].

Although the IETF still recommends supporting AES-XCBC-MAC ([RFC 3566], [RFC 4434], [RFC 7321]), it is not approved for use by the Federal Government. As such, AES-XCBC-MAC **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 may 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 an ICV size of 16 bytes for these three algorithms ([RFC 4309], [RFC 4106], [RFC 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, manually distributed keys **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, single 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 [SP 800-57 Part 1], could select the following algorithms to form an IPsec suite with an overall security strength of 112 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: 2048-bit MODP
- IKEv2 Integrity: HMAC-SHA1
- IKEv2 Peer Authentication: 2048-bit RSA with SHA-256.

The Suite-B-GCM-128 and Suite-B-GCM-256 suites are both defined in [RFC 6379]. At present, these cipher suites are not widely available or deployed. [SP 500-267] states that support for these cipher suites is optional. 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. As discussed above, AES-GCM is a combined-mode algorithm that provides both encryption and integrity protection; therefore these suites provide integrity, despite the fact that the integrity mechanism is listed in [RFC 6379] as NULL for both suites.

3.2.2 Additional Recommendations

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

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

3. 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. Ensure that IPsec implementations include approved algorithms for each IPsec security component.

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 **shall** 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)

This section was moved to SP 800-52 Revision 1, *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations* [SP 800-52].

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 5751], [RFC 5652], [RFC 2045], [RFC 2046], [RFC 2047], [RFC 4288], [RFC 4289] 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 e-mail, 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, 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 Section 2 of this Recommendation.

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 e-mail until it is no longer necessary for a recipient to be able to decrypt or verify the integrity of the e-mail.

S/MIME is not restricted to e-mail; 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 e-mail **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 e-mail 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 validated [FIPS 140-2],
- Support cryptographic Cipher Suite 1 (see Table 5-1 below), and
- Support 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 e-mail with security labels and securely binding senders' certificates to their signatures through the signing certificate attribute as described in [RFC 5035].[22]

The most widely accepted, standard S/MIME profile is [RFC 5751]. Not all cryptographic algorithms available for use in support of the features in the profile 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. Tables 5-1 through 5-3 specify a variety of cipher suites that may be used to protect federal information and information systems (based on [SP 800-49] and [RFC 6318]). Any of the algorithms listed in the following tables may be used, in accordance with security strength time frame restrictions given in Part 1 and [SP 800-131A] to protect federal information in combinations other than those displayed.

Table 5-1: Cipher Suite 1

Mechanism	Guidance
Digital Signatures	DSA with key sizes ≥ 2048 bits [FIPS 186-4]
Hash	SHA-256 [FIPS 180-4]
Key Agreement	Diffie-Hellman with key size ≥ 2048bits [SP 800-56A]
Encryption	AES-128 in CBC mode [FIPS 197] and [SP 800-38A]

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

Table 5-2: Cipher Suite B, Level 1*

Mechanism	Guidance
Digital Signatures	ECDSA with P-256 [X9.62]
Hash	SHA-256 [FIPS 180-4]
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] and [SP 800-38A]

*see [RFC 6318]

Table 5-3: Cipher Suite B, Level 2*

Mechanism	Guidance
Digital Signatures	ECDSA with P-384 [X9.62]
Hash	SHA-384 [FIPS 180-4]
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] and [SP 800-38A]

* see [RFC 6318]

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 [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, that 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, ensure that all federal information systems support Cipher Suite 1.

2. Procurements **should** support Cipher Suite B, either Level 1, Level 2, or both.

3. Federal clients **may** support RC2 for use only in the event that users receive correspondence encrypted with this weaker and unapproved algorithm.

4. Ensure that federal agencies do not use SHA-1 for digital signature generation; however, it can be used for the verification of digital signatures signed using SHA-1. Cryptographic algorithm implementations **should** be modular so as to allow for new algorithms.

If an S/MIME client needs to generate a key pair, ensure that the S/MIME client or some related administrative utility or function is 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 Suite 1 for interoperability as described above in Section 5.2.

2. 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.

3. Installers **should** install and configure S/MIME clients so that they default to the use of an approved cipher algorithm suite. Furthermore, installers **should** configure clients so 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.

4. System installers **should** configure clients so 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. 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.

3. Administrators **should** maintain S/MIME clients so 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.

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

5. 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.[23]

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

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

[23] 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 the Massachusetts Institute of Technology (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[24]. The original design and implementation of Kerberos and its first three revisions (i.e., versions 1 through 4) was primarily the work of Steve Miller, Clifford Neuman, Jerome Saltzer and Jeffrey Schiller[25]. 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 his 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.

[24] 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).

[25] The design was based in part upon a protocol proposed 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.

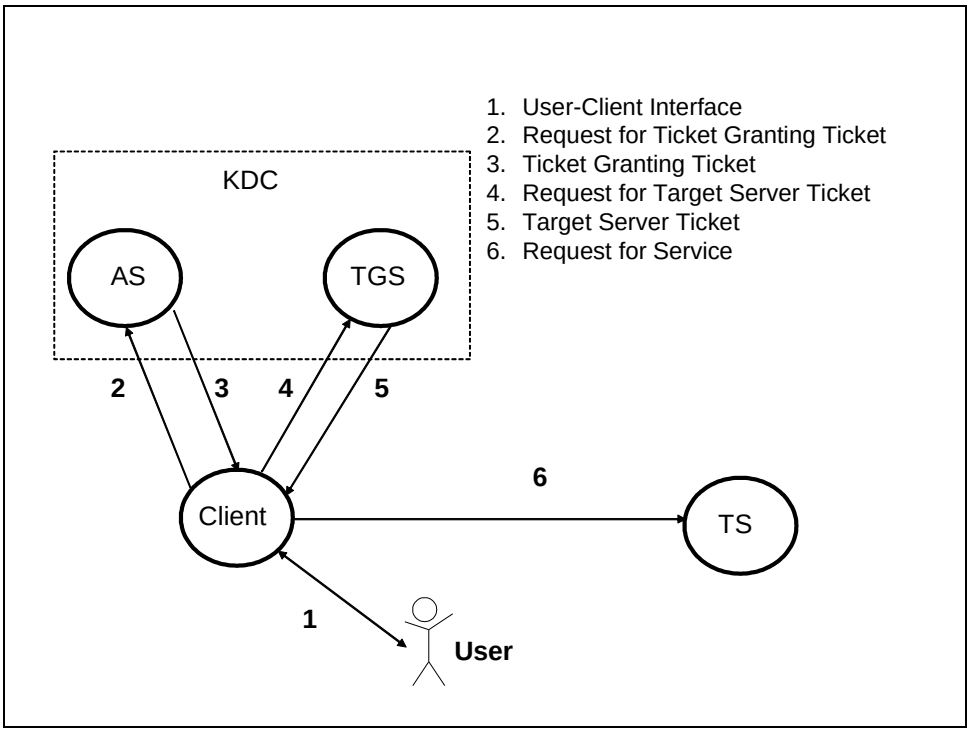

Figure 6-1: The Kerberos Protocol

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.

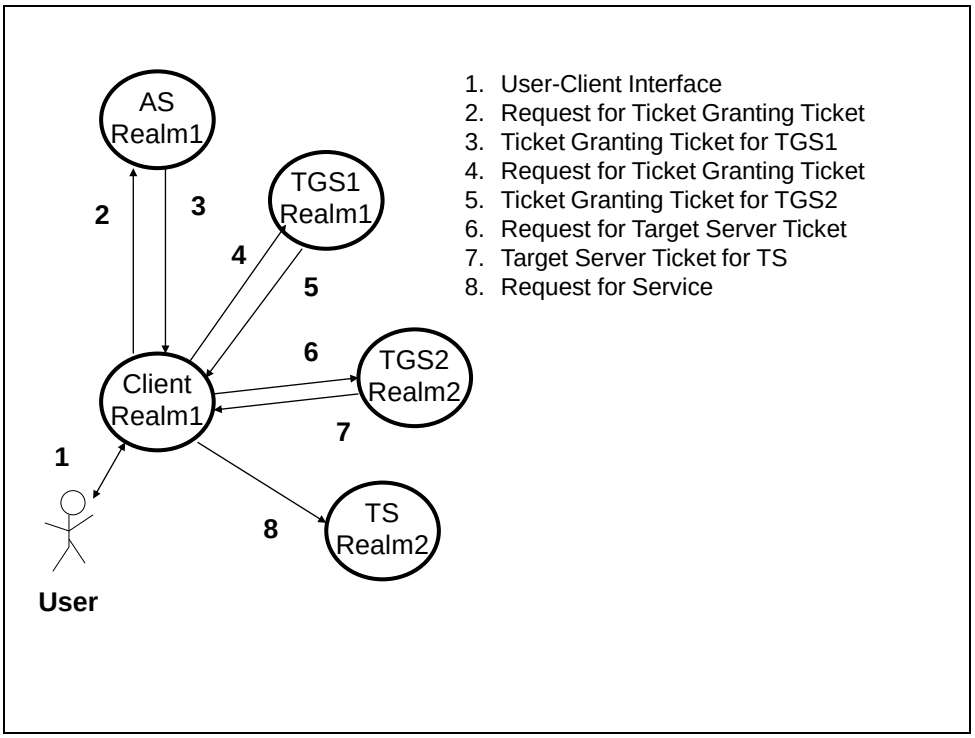

Figure 6-2: Cross-Realm Authentication

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[26], or
2. Using key transport (e.g., RSA), where the AS generates the user symmetric key, and sends the key to the client[27].

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

Kerberos implementations may provide an additional authentication capability that is called pre-authentication, as described in [RFC 6113]. TLS may also be implemented to protect all communication between the clients and KDCs as described in [RFC 6251].

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

[27] 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] and [RFC 6649]); 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 and has been deprecated as well in Kerberos [RFC 6649].
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[28] 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].[29]
5. Compromising the client, KDC, or TS could compromise the symmetric keys that they contain and thereby compromise parts of 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. 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.
7. A failure of the AS or the TGS would prevent all AS users from obtaining new tickets and corresponding new services.
8. Clocks must be synchronized in order to accurately assess the validity of clock authentication information and tickets. If the TS's clock is running behind the clock of the KDC (or AS), then previous authentication information and tickets could be played back to the TS after they have expired.
9. If a Kerberos implementation has a TLS capability, then it **should** be used when the DH key agreement or RSA key transport method discussed above is not used.

[28] 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.

[29] Draft SP 800-118, *Guide to Enterprise Password Management*, is currently under development [SP 800-118].

6.3 Procurement Guidance

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

1. Ensure that new procurements conform to version 5 ([RFC 4120], [RFC 6649]).
2. Ensure that government procurements specify the inclusion of approved symmetric key encryption algorithms (e.g., the Advanced Encryption Standard (AES)) [RFC 3962].
3. Ensure that an approved MAC computation (e.g., HMAC-SHA1 or HMAC-SHA256-128) is available for data integrity with encryption (see [RFC 3962] and the IETF Informational Draft, *AES Encryption with HMAC-SHA2 for Kerberos 5*, at: http://tools.ietf.org/html/draft-ietf-kitten-aes-cts-hmac-sha2-04).
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. Ensure that passwords stored on tokens are randomly generated; therefore, when tokens are used, ensure that a means of generating random passwords and securely writing them on the token is available and the password generation function is a NIST-approved random number generation function [SP 800-90A]. When tokens are used, ensure that the manual entry of passwords is not permitted except to authenticate the user to the token.
5. If passwords are used to form user symmetric keys, ensure that the password mechanism supports the use of strong passwords [SP 800-118], and an approved hash algorithm (e.g., SHA-1 or stronger) is used as the hash algorithm.
6. If passwords are generated by users, ensure that the system software enforces a strong password policy in accordance with [SP 800-118].
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] and [RFC 5349] for further information. Ensure that the key-establishment methods (i.e., key-agreement or key-transport methods) used have at least 112 bits of security; see [SP 800-57 Part 1] and [SP 800-131A] for further information.
8. TLS **should** be used when the system supports it.
9. Procurement officials **should** consider whether inter-realm networking is necessary and include the capability in the software if it's needed.
10. Ensure that cryptographic modules used by CAs, TSs and clients are validated at FIPS 140-2 Level 1 or higher [FIPS 140-2].

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. Government systems **shall** be configured so that approved algorithms (e.g., AES) **shall** be used [RFC 3962], and DES **shall not** be used [RFC 6649].
2. An approved MAC checksum (e.g., HMAC-SHA1 or HMAC-SHA256-128) **shall** be installed and used in all implementations for data integrity purposes (see [RFC 3962] and the IETF Informational Draft, *AES Encryption with HMAC-SHA2 for Kerberos 5*, at: http://tools.ietf.org/html/draft-ietf-kitten-aes-cts-hmac-sha2-04).
3. The AS, TGS, TSs, and clients **shall** use strong access control mechanisms[30] (physical and logical) for protecting and updating keys.
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 by a NIST-approved random number generation function ([SP 800-90A], [SP 800-90B], [SP 800-90C]), 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.
5. 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.
6. TLS **should** be used when the system supports it, see Section 4 for guidelines on using TLS.
7. If user passwords are generated by the system, the system **shall** generate strong passwords [SP 800-118]. 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.
8. If user passwords are used to generate cryptographic keys, the password mechanism **shall** be configured to use and require strong passwords [SP 800-118].[29]
9. A backup AS and TGS **should** be provided so as to minimize the impact in case of operational failure or denial-of-service attacks.
10. Clocks **should** be synchronized periodically and whenever a new system is brought on-line[31].

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.

---

[30] 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).

[31] See http://www.nist.time.gov.

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.
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].[29]
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) [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 [OTAR1][32].

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 and 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).

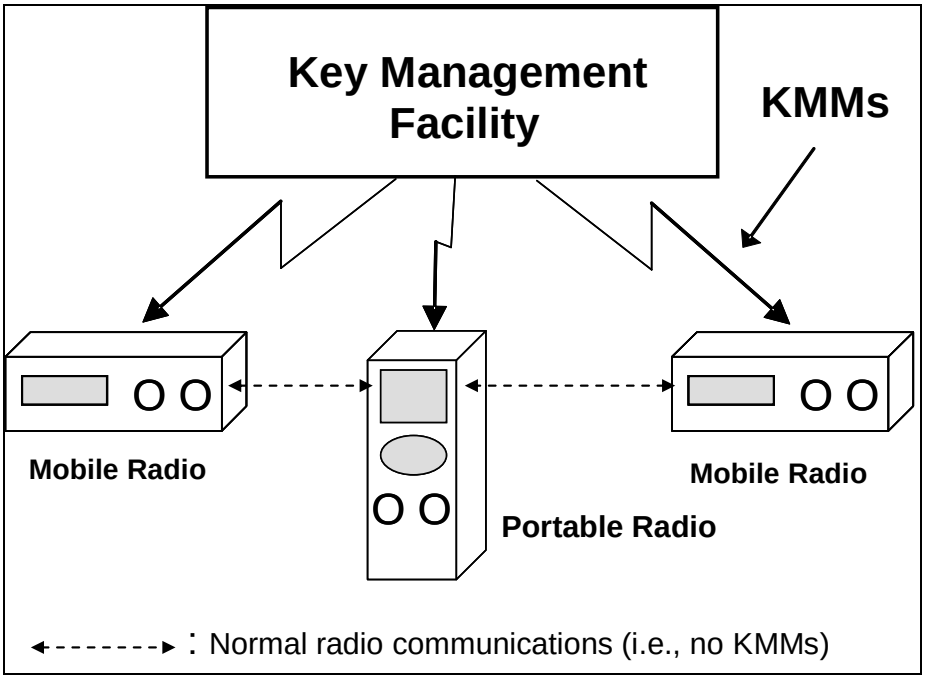

Figure 7-1: Radio Communications with OTAR

[32] Reference [OTAR] provides an overview of the key management techniques and the protocol. Reference [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)[33], 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, as specified in [SP 800-67], 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 and **shall not** be used.

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 **shall** be used for only one purpose.[34] However, OTAR1 states that the key used to generate a MAC must be either a key

[33] A Key Wrapping Key may also be referred to as a Key Encryption Key (KEK).

[34] There is an allowed exception to this rule, but it does not apply to OTAR [SP 800-57 Part 1, Sec. 5.2]).

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 **shall** 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. NIST-approved random bit generation methods are specified in [SP 800-90A], [SP 800-90B] and [SP 800-90C] (under development).

7.3 Procurement Guidance

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

1. Ensure that the AES or TDEA algorithm is included.

2. If TDEA is provided in an implementation, ensure that the three-key version is included, and the implementation is capable of limiting the number of uses of a single TDEA key bundle to 1 000 000.[35]

3. Ensure that KMFs and radios conform to OTAR and OTAR1.

4. When keys are generated within KMFs**,** ensure that they are generated using approved random bit generators.

5. Ensure that cryptographic modules used by the KMF and the mobile radios are validated at FIPS 140-2 Level 1 or higher [FIPS 140-2].

[35] See Section 7.2.2.

6. Ensure that KMFs 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 **shall** be provided.

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. Backup KMFs 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 **shall** be established.

6. 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.

7. 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

The Domain Name System (DNS), as defined in [RFC 1034] and [RFC 1035], is the global hierarchical distributed database system for mapping Internet addresses, Simple Mail Transfer Protocol (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, the geographic location of servers, even encoded digital certificates. DNS data is stored as individual Resource Records (RRs) each 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[36], 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.

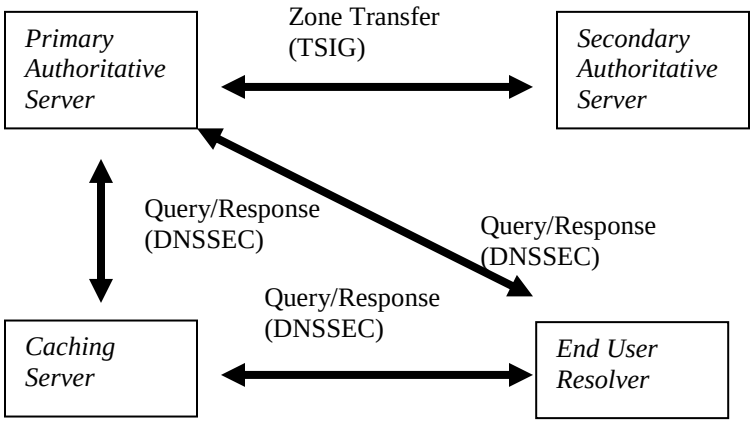

Figure 8-1: DNS Components

[36] 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 RFCs has been developed to provide security enhancements contained in three IETF documents, collectively called the DNS security extensions (DNSSEC) ([RFC 4033], [RFC 4034], [RFC 4035]). 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 key pair is used to sign the zone key set (known as the Key Signing Key or KSK). While some zones may only use one key pair for both ZSK and KSK, it is not recommended for federal agency zones. 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[37] 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.

Currently, both RSA using SHA-1 and SHA-256 have been specified and can be used with DNSSEC zones. Most modern implementations either support both or will do so after an upgrade. Zones deploying DNSSEC for the first time can start with RSA using SHA-256. Zones that initially deployed with RSA using SHA-1 **should** migrate to RSA (2048-bit RSA key) using SHA-256: see [SP 800-131A] for information about guidelines for using RSA with different key sizes and hash algorithms. 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.

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

[37] 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.

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 algorithms used in TSIG authentication are HMAC using SHA-1 and the SHA-2 family of hash algorithms (SHA-224, SHA-256, SHA-384 and SHA-512). The use of SHA-1 is acceptable for current security practices when using HMAC with a suitably random secret string. 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 Table 8-1 ([RFC 6944], [RFC 6605]) and Table 8-2 ([RFC 3645], [RFC 4635]):

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

Suite	Authentication	Digest	IETF Status	Approved for Federal Use[38]
RSA_SHA-256	RSA	SHA-256	Recommended to Implement	YES
RSA_SHA-512	RSA	SHA-512	Recommended to Implement	YES
ECDSAP256SHA256	ECDSA	SHA-256	Recommended to Implement	YES
ECDSAP384SHA384	ECDSA	SHA-384	Recommended to Implement	YES

[38] Refer to Part 1 of this guide for approved key lengths and for algorithm lifetimes [SP 800-57 Part 1].

Table 8-2: Recommended Message Authentication Algorithms

Suite	IETF status	Approved for Federal Use
HMAC_SHA1	Mandatory	YES
HMAC_SHA-224	Optional	YES
HMAC_SHA-256	Mandatory	YES
HMAC_SHA-384	Optional	YES
HMAC_SHA-512	Optional	YES
GSS_TSIG[39]	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 on the method used and the secret (random) string used with the TSIG method.

Due to message size constraints (See Section 8.1.4 below), large RSA keys may result in DNS transaction failures that are often interpreted by clients as DNS failures. It is recommended that, instead, a digital signature algorithm that has the same security strength, but with smaller sized keys be used, such as ECDSA [RFC 6605]. It is recommended that DNS administrators plan to migrate to ECDSA for zone signing by October 1, 2015, or plan to migrate earlier as soon as it becomes available in DNS software components.

8.1.4 Special Considerations for Key Sizes

There are some special considerations needed when choosing the size of the RSA DNSSEC signing keys. Early deployments have shown that large RSA keys can result in protocol issues, such as response messages that are too large to fit in a standard UDP packet. DNSSEC requires the use of larger DNS packet sizes up to 4 KB, but practical limits are around 1500 bytes or less.

It is recommended that DNS administrators maintain 1024-bit RSA/SHA-1 and/or RSA/SHA-256 ZSK's until October 1, 2015, or until it is proven that the majority of routers, caches and other network middle boxes can handle packet sizes over 1500 bytes (if before 2015). However, 1024-bit RSA keys are allowed until the 2015 date to accommodate older versions of DNS clients that may be older network components that cannot handle UDP packets with sizes over 1500 bytes. This is an exception to the security guidance provided in [SP 800-131A]. However, this exception on RSA key sizes

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

does not apply to Key Signing Keys. KSK's **shall** follow the guidance set in Part 1 of this Recommendation [SP 800-57 Part 1].

It is recommended that zone administrators migrate DNSSEC zone signing algorithms to ECDSA by 2015 or when support for ECDSA appears in DNSSEC components, whichever is sooner.

To minimize the risk when using 1024-bit RSA ZSK's with DNSSEC, ZSK's should be changed more frequently: every one to three months, with a signature validity period of five to seven days. The ZSK-rollover sequence discussed in [SP 800-81] is recommended to maintain a valid chain of authentication in DNS data.

8.1.5 Special Considerations for NSEC3

There is a special variant 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 overriding need to deploy NSEC3 beyond zone content protection (examples include protecting personally identifying information that may be contained in the DNS). However, it is a good idea to use NSEC3-aware client software, because a client may access a zone that uses NSEC3 RRs with DNSSEC.

System installers and administrators **should** develop a transition plan to migrate from SHA-1 to SHA-256, and to do so when SHA-256 becomes available in major software distributions. 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. Even though 1024-bit RSA keys are allowed until October 1, 2015 to accommodate older versions of DNS clients, 2048-bit RSA keys are strongly recommended for use.

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.

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

8.3 Procurement Guidance

The following recommendations are for any individual that makes a purchasing decision for acquiring DNSSEC-capable components for their network infrastructure.

1. Ensure that DNSSEC utilities use FIPS140-2 validated cryptographic modules [FIPS 140-2].
2. DNS server software **should** generate and serve NSEC3 RRs, if required by zone policy.
3. Ensure that DNS server software 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. Ensure that DNSSEC-enabled versions of network applications are purchased as required by security policy, if available.
5. Ensure that DNSSEC software implementing SHA-256 is included in procurements when available.

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 [SP 800-90A]) 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. The key(s) **shall** be kept up-to-date to insure successful validations.

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. The 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.

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[40] 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 in the second case above. One option to limit access in this second system is for the 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,

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

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 strength as the original key protecting that file.

Another option is for the sharing user to be provided with 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 a 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.[41] 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.

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:

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

- 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 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 [SP 800-132]. 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). A standard dictionary attack can often recover weak passwords, so a strong password is vital for the security of this type of system. It is preferable to utilize a good random bit generator within the system to generate keys. Approved random bit generators can be found in [SP 800-90A], [SP 800-90B] and [SP 800-90C].

Next, consider the question of how to protect the keys. 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.

Then, consider where these keys will be stored. If 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 smartcards. The advantage of 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 a method in [SP 800-132] 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) [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 a method in [SP 800-132] 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 the pros and cons of the systems more clearly.

Table 9-1: Summary of File Encryption Examples

Method	Pros	Cons
Example 1: SP 800-132	- 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 a token is used. - Compromise of a user's private key compromises only the user's files.	- Requires the user to manage his/her own 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].

9.3 Recommendations for Procurement Officials

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

1. Ensure that an encrypted file system includes 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. Ensure that any EFS system that derives keys from passwords has the capability of enforcing the use of strong passwords.
3. To increase the security of encrypted file systems, the system **should** use a hardware token or a TPM for the 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. 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.

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 The Secure Shell (SSH)

10.1 Description

SSH is a protocol between clients and servers for secure remote login and other secure network services over an insecure network or the Internet. The Internet Engineering Task Force (IETF) governs the SSH protocol [RFC 4251][42]. The SSH protocol consists of three major components: the Transport Layer Protocol [RFC 4253], the User Authentication Protocol [RFC 4252] and the Connection Protocol [RFC 4254].

10.1.1 Transport Layer Protocol (SSH-TLP)

The Transport Layer Protocol (TLP) provides server authentication, confidentiality, and integrity with perfect forward secrecy[43]. The establishment of an SSH connection is initiated using the TLP, which negotiates the algorithms to be used and authenticates the server. TLP does not provide client authentication.

The algorithm-negotiation step of the protocol is used to determine the algorithms to be used by the client and server for key agreement (called key exchange in the protocol), server authentication, encryption and data integrity. The encryption and data-integrity algorithms that protect communications from the client to the server and from the server to the client are independently selected, i.e., the encryption algorithm protecting communication from the client to the server may be different than the encryption algorithm protecting communication from the server to the client.

After the algorithm negotiation is completed, the TLP authenticates the server to the client in a key-exchange process [RFC 4253, Sec. 8]) that provides keys and IVs for the selected cryptographic algorithms. The key-exchange process provides assurance that the server is the owner of the public key, but additional steps may be required to verify the server's identity; see Section 10.2.1.2 for a discussion on the verification of the server's public host key.

The protocol provides data- integrity protection by choosing a MAC algorithm for each communication direction between the server and the client. However, the protocol also allows this service (i.e., data-integrity protection) to be disabled.

10.1.2 The User Authentication Protocol (UAP)

The User Authentication Protocol authenticates the client to the server. After the TLP is completed, the server and the client communicate using an encrypted SSH tunnel[44] that uses the selected encryption algorithm(s) from the negotiation process and the keys from

[42] This section discusses SSH version 2, sometimes called SSH2 or SSH-2. SSH1 never became a standard protocol and was later replaced by the SSH2 protocol, which is the SSH protocol in [RFC 4251].

[43] Perfect forward secrecy is a cryptographic property of a key-establishment method in which the compromise of a currently established session key or long-term private key does not cause the compromise of any earlier established session keys.

[44] An encrypted tunnel is an end-to-end communication connection where all of the data traffic going through the connection is encrypted.

the key exchange (see the SSH TLP protocol discussion in Section 10.1.1). Using the encrypted SSH tunnel, the server can securely perform any client authentication. The UAP provides a single authenticated tunnel for the SSH connection protocol.

10.1.3 Connection Protocol (CP)

The Connection Protocol [RFC 4254] multiplexes the encrypted tunnel used by SSH into several logical channels. The CP defines how interactive login sessions, remote execution of commands, forwarded TCP/IP connections, and forwarded X11 connections [X11] can be run simultaneously over an established SSH Transport Layer and User Authentication connection.

10.2 Security and Compliance Issues

10.2.1 TLP Issues

10.2.1.1 Algorithm Negotiation

In this step of the TLP, the algorithms to be used for the key exchange (i.e., key agreement), public key authentication, data encryption and message authentication are selected.

1. The SSH server and client **should** choose the same NIST-approved cryptographic algorithms for both communication directions (data streams) for a particular cryptographic service. For example, the same encryption algorithm and MAC-generation algorithm **should** be used for both communication directions to provide confidentiality and integrity protection, respectively.

   Note: The cryptographic algorithms selected for use depend on the algorithms that are supported by both the server and the client, and by the client's preference levels for these algorithms in each of its algorithm lists offered in the negotiation; the client's preference level is indicated by the order in which the algorithms are listed. See [RFC 4253, Sec. 7.1] for the defined procedure for selecting the cryptographic algorithms.

2. Suite B cryptographic algorithms are recommended for use when supported by both the client and server systems. Suite B cryptographic algorithms are specified in [RFC 6239]. They are:

   - Key agreement (key exchange): ecdh-sha2-nistp256 and ecdh-sha2-nistp384 (see Section 10.2.1.2).
   - Public key algorithm (for server and client authentications): x509v3-ecdsa-sha2-nistp256 and x509v3-ecdsa-sha2-nistp384 (see Section 10.2.1.3).

     The public keys are conveyed in X.509 version 3 certificates.
   - Encryption and MAC: AEAD_AES_128_GCM and AEAD_AES_256_GCM (see Sections 10.2.1.4 and 10.2.1.5).

10.2.1.2 Key Agreement /Key Exchange Algorithms

Two families of Diffie-Hellman key-agreement/key-exchange algorithms have been specified for use in SSH: those based on finite fields and those based on elliptic curves.

- RFC 4253 specifies two finite-field key-exchange methods: “diffie-hellman-group1-sha1” and “diffie-hellman-group14-sha1” [RFC4253]. Since diffie-hellman-group1-sha1 uses 1024-bit keys, which provide less than 112 bits of security strength, it **shall not** be used. However, note that the use of SHA-1, in this case, is acceptable.
- RFC 4419 specifies two additional finite-field Diffie-Hellman key-exchange methods: “diffie-hellman-group-exchange-sha1” and “diffie-hellman-group-exchange-sha256” [RFC 4419]. Although the modulus length (i.e., key size) that can be supported by these two methods is between 1024 and 8192 bits, a modulus length of at least 2048 bits **shall** be used.
- RFC 5656 specifies key-agreement options based on elliptic curves that can be used instead of the options described above [RFC 5656]. Two of the options are mandatory-to-implement for the Federal Government when supporting Elliptic Curve Cryptography for SSH: “ecdh-sha2-nistp256” and “ecdh-sha2-nistp384”. These options specify the use of elliptic curve Diffie Hellman with the appropriate, NIST-approved hash function and curve: “ecdh-sha2-nistp256” specifies the use of SHA-256 and the nistp256 curve, while “ecdh-sha2-nistp384” specifies the use of SHA-384 and the nistp384 curve [RFC 5656]. See [FIPS 186-4] for information about the curves.

10.2.1.3 Public Key Authentication Algorithms

Public-key authentication algorithms are the digital-signature algorithms that can be used in the key exchange specified in [RFC 4253, Sec. 8] to perform server authentication. RFC 4253 specifies two digital signature algorithms: RSA and DSA (which is called “ssh-dss” in the RFC) using SHA-1 as the hash function for server authentication. It is important to note that according to [SP 800-131A], SHA-1 is no longer allowed for generating digital signatures. However, in this protocol, SHA-1 is allowed for server authentication, as long as the public key size of the signing function (either RSA or DSA) is at least 2048 bits. The protocol allows a server’s public key to be used without validation (i.e., without obtaining assurance of public key validity). For Federal Government use, the client **shall** obtain assurance of public key validity, as required in [SP 800-89], before digital signature verification can be performed.

Additional authentication algorithms for SSH are specified in [RFC 5656]. These use the Elliptic Curve Digital Signature Algorithm (ECDSA) using the curves specified in [FIPS 186-4]. Table 10-1 identifies the requirements for the implementation of the ECDSA algorithm and the curves to be used by the Federal Government.

Table 10-1: Public Key Authentication Methods Using Elliptic Curves

ECDSA	Hash Algorithm	IETF	Federal Government
ecdsa-sha2-nistp256	SHA-256	Mandatory	Mandatory
ecdsa-sha2-nistp384	SHA-384	Mandatory	Mandatory

[RFC 6187] specifies the use of X.509 version 3 certificates to convey public keys for the digital-signature algorithms. It also formally defines 2048-bit RSA with SHA-256 as a method for use in SSH. This combination provides 112 bits of security and is allowed. It is important to note that the hash algorithm used with the digital signature algorithms for authentication can be different than the hash function HASH in the key exchange and the key-derivation functions of SSH TLP.

10.2.1.4 Encryption Algorithms

Several encryption algorithms are available for SSH; Federal Government applications **shall** use NIST-approved algorithms. The currently specified encryption algorithms and mode(s) of operation for SSH that are NIST-approved are 3DES-CBC, AES-128-CBC, AES-192-CBC and AES-256-CBC [RFC 4253], and AEAD_AES_128_GCM and AEAD_AES_256_GCM [RFC 5647]. 3DES-CBC is mandatory for IETF implementation.

In the protocol, when either AEAD_AES_128_GCM or AEAD_AES_256_GCM is selected as the encryption algorithm for protecting an encrypted tunnel (client-to-server or server-to-client), it is also chosen as the MAC algorithm. A different algorithm or set of algorithms may be selected for communication in each direction. It should be noted that when AEAD_AES_128_GCM or AEAD_AES_256_GCM is chosen (for encryption and MAC generation), the algorithm is executed only once (rather than once for encryption and another time for integrity protection), because these modes provide both confidentiality and integrity protections at the same time.

When AEAD_AES_128_GCM or AEAD_AES_256_GCM is used, the SSH packet-length field is not encrypted, but is processed as additional authenticated (plaintext) data. See [RFC 5647] for more details on using these algorithms in SSH.

10.2.1.5 Message Authentication Codes (MAC) Algorithms

Several MAC algorithms are available for SSH; Federal Government applications **shall** use NIST-approved algorithms. The currently specified MAC algorithms for SSH that are NIST-approved are HMAC-SHA1 (with MAC tags of 160 bits), HMAC-SHA1-96 (with MAC tags of 96 bits [RFC 4253]), AEAD_AES_128_GCM and AEAD_AES_256_GCM [RFC 5647].

As explained in Section 10.2.1.4, AEAD_AES_128_GCM or AEAD_AES_256_GCM can only be chosen as the MAC algorithm when also chosen as the encryption algorithm.

10.2.1.6 Public Host-Key Verification

After the algorithm negotiation is completed, the TLP authenticates the server to the client in a key-exchange process [RFC 4253, Sec. 8]. In this key exchange, a Diffie-Hellman (DH) key exchange is performed, producing two values: a DH value[45] *K* and an exchange hash value called *H*. *H* is the hash value of *K* and other data [RFC 4253, Sec. 8] for a complete specification of *H*). The generated *H* and *K* are subsequently used as inputs to a key-derivation function [RFC 4253, Sec. 7.2] for the specification of this function) to generate IVs and keys for the selected cryptographic algorithms.

[45] In the key-agreement schemes in [SP 800-56A], *K* would be considered to be the shared secret.

To prove that the server is actually the owner of the public key (called the public host key in the protocol), the server generates a digital signature using its private key (called the private host key in the protocol) on *H* (i.e., data that is known by both the client and the server). If the client can verify the digital signature using the public host key, then the client has assurance that the server is the owner of the public key. However, the identity of the server has not been verified unless the public host key is verified by the client before the signature on *H* is verified.

Verifying the public host key is performed by 1) comparing the server name and its public host key against a database of trusted server names and public host keys, or 2) using the public host key certificate (i.e., the server's public key certificate), or 3) using an out-of-band verification method using DNSSEC upon initiating SSH TLP[46]. A verification of the association between the public host key and the server name is essential to the security of the SSH connection. If the connection is not verified, then the connection is subject to a man-in-the-middle attack; details of this vulnerability are addressed in [RFC 4251]. Therefore, the association of a public host key (i.e., the server's key) to a server name **shall** be verified for every TLP session, and the TLP **shall** fail if the verification fails.

- When the first method of server authentication is used, and it is the only method available for the client, the protocol **shall** continue only when a match is found. Note that the protocol does not explicitly require the connection to fail when the public host key is not verified, however; this Recommendation requires a discontinuation of the protocol in this case.
- For the second method, the TLP specified in [RFC 4253] allows the client to either a) accept the presented public key certificate without verification of the association between the public key and the server name, or b) verify the certificate and continue only if the public host key is verified (see Section 2 for details on how to verify a public key in a public key certificate). For Federal Government use, option b) **shall** be used; that is, when this second method is used, the protocol **shall not** continue unless the server's certificate has been successfully verified.
- When an out-of-band verification method using DNSSEC is used (method three), the server's identity and public key **shall not** be accepted unless the fingerprint (hash value) of the public host key matches a fingerprint of the public key in the "SSHFP" resource record(s) of the server. "SSHFP" resource record(s) **shall** be verified before the fingerprint is accepted as a legitimate fingerprint. See [RFC 4255] for details about this method.

Other server-authentication methods may be defined later for the protocol.
It is important to note that the key exchange specified in [RFC 4253, Sec. 8] contains a Diffie-Hellman primitive specified in [SP 800-56A] that generates a shared secret. In [SP 800-56A], a complete key-agreement scheme contains a specific key-derivation method (e.g., a key-derivation function) that uses the shared secret to derive keying material. In SSH, the shared secret is provided, instead, to the key-derivation function specified in

[46] Details of the DNSSEC can be found in [RFC 4255].

[RFC 4253, Sec. 7.2]. This key-derivation function has been **approved** in [SP 800-135]. Therefore, the Diffie-Hellman key exchange in [RFC 4253, Sec. 8], combined with the key-derivation function in [RFC 4253, Sec. 7.2] is an **approved** key-agreement method for SSH TLP.

10.2.2 UAP Issues

There are many authentication methods that the server can use to authenticate the client. The required method that the server must support is public-key authentication, named "publickey" in the UAP [RFC 4252, Sec. 7]. This requires the use of a digital-signature algorithm as specified in Section 10.2.1.3 except the digital signatures using SHA-1. SHA-1 digital signatures **shall not** be used for client authentication. See [RFC 4252, Sec. 7] for specific details on how client authentication using one of the digital-signature algorithms is done. For Federal Government applications, the digital-signature algorithms used for client authentication **shall** meet the requirements specified in [SP 800-131A].

In the "publickey" authentication method, proof-of-possession of the valid private-key is considered to be a proof of the identity of the client. Therefore, sharing a private-key among different clients (users) is prohibited. However, in current practice, many organizations allow multiple clients (users) to share a private key. To address this issue and other issues related to identity management, NIST recently published Draft NISTIR 7966, *Security of Automated Access Management Using Secure Shell (SSH)* [NISTIR 7966].

Beside the "publickey" method, two other authentication methods have been defined for this protocol. The first method is using passwords. The second method is using the private key of a host system that is trusted by the server in this SSH connection; the latter method is called "host-based authentication".

- For Federal Government use, authentication using passwords **shall not** be used if the TLP does not provide an encrypted tunnel, because the password would be sent as cleartext.
- For the "host-based authentication" method [RFC 4252, Sec. 9], the client is a user of the host system that is attempting to authenticate to the server. In this method, the client knows the private signature key of the host. The client uses this private key to generate a digital signature on behalf of this host during the authentication process with the server in the SSH connection. The server verifies the digital signature to authenticate the host system. If the authentication is successful, and the client (user) is an authorized user associated with this host, then this user (the client) is considered to be authenticated by the SSH server. This "host-based authentication" method **should not** be used for client authentication because the method does not provide direct, cryptographic assurance of the identity of the client to the server - the server must trust the host system to obtain the correct identity of the client.

Also, client authentication using Kerberos (see Section 6) is also described in [RFC 4462]. It is important to note that RFC 4462 specifies an authentication method called "Authentication Using GSS-API Key Exchange" [RFC 4462, Sec. 4]. This authentication

method **shall not** be used as a stand-alone authentication method because the method does not prove the identity of the client.

10.3 Procurement Guidance

The following recommendation is for any individual that makes a purchasing decision for acquiring SSH client and server implementations.

1. Ensure that client and server implementations support at least one of these NIST-approved encryption algorithms: AES-128-CBC, AES-192-CBC and AES-256-CBC, in addition to 3DES-CBC, which is already mandatory to implement for the protocol. Implementations that support all of these encryption algorithms **should** be selected.
2. When AEAD_AES_128_GCM or AEAD_AES_256_GCM is used in a negotiating encryption-algorithm list, it must also be in the corresponding MAC-algorithm list. Ensure that an implementation either enforces this automatically or provides it as a configurable option.
3. Implementations **should** be chosen that support Suite B cryptographic algorithms as described in Section 10.2.1.1 above.
4. Ensure that client and server implementations support public-key authentication using public-key certificates.
5. Ensure that client and server implementations allow a protocol to be discontinued when the verification of the public-key-certificate fails.
6. Ensure that client implementations allow preference setting for implemented cryptographic algorithms.
7. Ensure that server implementations allow the configuration of the cryptographic algorithms to be used.
8. Ensure that server implementations are able to disallow the password-authentication method when encryption is not used in the client-to-server traffic direction.
9. Ensure that server implementations are able to disallow the "host-based authentication method".

10.4 Recommendations for System Installers

The system installer is the individual(s) that installs the SSH server and client applications and performs the initial configuration of the system. The system installer **shall**:

1. Install and/or configure the server and client to use only **approved** cryptographic algorithms.
2. Set the connection to fail when a public-key certificate is not verified at both sides: server and client.
3. Configure preferences in negotiating cryptographic-algorithm lists according to the organization's security guidelines for the client. For example, if the

organization prefers the use of AES-256, then AES-256 must be configured to have the highest preference among all of the available encryption choices.

4. Configure the use of only **approved** cryptographic algorithms by the server.
5. Consider disabling the "host-based authentication" option at the server if the server is not willing to communicate with all possible users/clients at the host machine.
6. When AEAD_AES_128_GCM or AEAD_AES_256_GCM is in a negotiating encryption-algorithm list, configure it to also be its companion MAC-algorithm if this is not done automatically.
7. Suite B cryptographic algorithms **should** be set with the highest preferences in the client-side application. This will result in the selection of Suite B cryptographic algorithms to be used when the server supports them.

The installer **shall** make sure that all of the cryptographic components and required plug-ins are installed so that cryptographic operations in SSH can function properly when protecting the data traffic.

10.5 Recommendations for System Administrators

System administrators are those individuals responsible for the day-to-day functioning of the SSH server and client applications. System administrators **shall**:

1. Configure the client to verify the server's public-key certificate in the key-exchange protocol in TLP and discontinue the protocol if the verification fails.
2. Configure the server to authenticate the client by verifying the client's certificate, if provided, when a public-key algorithm for authentication is used in UAP.
3. Ensure that end users are properly trained to follow the organization's security policy for the selection, use and protection of passwords.
4. Ensure that the organization's security policy is enforced.
5. Ensure the protection of private key(s) associated with the server's certificate and the client's certificate from disposal, leaks or unauthorized access is/are properly configured.

10.6 Recommendations for End Users

An end user is the individual using the SSH client application to securely connect to the SSH server. 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.