This **DRAFT SPECIAL PUBLICATION (SP)** (Draft SP 800-52, Revision 1) document has been approved as **FINAL**, and has been superseded by the following publication:

Publication Number: **Special Publication 800-52 Revision 1**

Title: **Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations**

Publication Date: **April 2014**

- Final Publication:
  NIST Publication Portal
  http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf
  DOI URL (note: The DOI actually redirects to the URL above):
  http://dx.doi.org/10.6028/NIST.SP.800-52r1

- *Link to NIST SP 800-52 Revision 1 can be found on the CSRC Special Publications page at:*
  http://csrc.nist.gov/publications/PubsSPs.html#800-52

- Information on other NIST Computer Security Division publications and programs can be found at: http://csrc.nist.gov/

The following information was posted to CSRC announcing release of this document:

Sep. 24, 2013
SP 800-52 Rev. 1
DRAFT *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations*

NIST announces the release of draft Special Publication (SP) 500-52 (Revision 1), Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations for public comment. TLS provides mechanisms to protect sensitive data during electronic dissemination across networks. This Special Publication provides guidance to the selection and configuration of TLS protocol implementations while making effective use of Federal Information Processing Standards (FIPS) and NIST-recommended cryptographic algorithms. The revised guidelines include the required support of TLS version 1.1, recommended support of TLS version 1.2, guidance on certificate profiles and validation methods, TLS extension recommendations, and support for a greater variety of FIPS-based cipher suites.

NIST requests comments on draft SP 800-52 Revision 1 by November 30, 2013. Please send comments to SP80052-comments @nist.gov with the subject “Comments NIST SP 800-52”. A template for submitting comments is provided below.

NIST Special Publication 800-52
Revision 1

Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations

Tim Polk
Santosh Chokhani
Kerry McKay

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

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

NIST Special Publication 800-52
Revision 1

Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations

Tim Polk
Kerry McKay
Computer Security Division
Information Technology Laboratory

Santosh Chokhani
CygnaCom Solutions

September 2013

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Patrick D. Gallagher, Under Secretary of Commerce for Standards and Technology and Director

Authority

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

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

National Institute of Standards and Technology Special Publication 800-52 Revision 1
Natl. Inst. Stand. Technol. Spec. Publ. 800-52 Revision 1, 64 pages (September 2013)
http://dx.doi.org/10.6028/NIST.SP.XXX
CODEN: NSPUE2

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

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

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

Public comment period: *September 24, 2013* through *November 30, 2013*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: SP80052-comments@nist.gov

Reports on Computer Systems Technology

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

Abstract

Transport Layer Security (TLS) provides mechanisms to protect sensitive data during electronic dissemination across the Internet. This Special Publication provides guidance to the selection and configuration of TLS protocol implementations while making effective use of Federal Information Processing Standards (FIPS) and NIST-recommended cryptographic algorithms, and requires that TLS 1.1 configured with FIPS-based cipher suites as the minimum appropriate secure transport protocol and recommends that agencies develop migration plans to TLS 1.2 by January 1, 2015. This Special Publication also identifies TLS extensions for which mandatory support must be provided and other recommended extensions.

Keywords

information security; network security; SSL; TLS; Transport Layer Security

Acknowledgements

The authors, Tim Polk and Kerry McKay of NIST, and Santosh Chokhani of CygnaCom Solutions would like to thank the many people who assisted with the development of this document. In particular we would like to acknowledge Matthew J. Fanto and C. Michael Chernick of NIST and Charles Edington III and Rob Rosenthal of Booz Allen and Hamilton who wrote the initial published version of this document.

Table of Contents

Tables

Executive Summary

Office of Management and Budget (OMB) Circular A-130, *Management of Federal Information Resources,* requires managers of publicly accessible information repositories or dissemination systems that contain sensitive but unclassified data to ensure that sensitive data is protected commensurate with the risk and magnitude of the harm that would result from the loss, misuse, or unauthorized access to or modification of such data. Given the nature of interconnected networks and the use of the Internet to share information, protection of this sensitive data can become difficult if proper mechanisms are not employed to protect the data. Transport layer security (TLS) provides such a mechanism to protect sensitive data during electronic dissemination across the Internet.

TLS is a protocol created to provide authentication, confidentiality and data integrity between two communicating applications. TLS is based on a precursor protocol called the Secure Sockets Layer Version 3.0 (SSL 3.0) and is considered to be an improvement to SSL 3.0. SSL 3.0 is specified in [RFC6101]. The Transport Layer Security version 1 (TLS 1.0) specification is an Internet Request for Comments [RFC2246]. Each document specifies a similar protocol that provides security services over the Internet. TLS 1.0 has been revised to version 1.1, as documented in [RFC4346], and TLS 1.1 has been further revised to version 1.2, as documented in [RFC5246]. In addition, some extensions have been defined to mitigate some of the known security vulnerabilities in implementations using TLS. These vulnerabilities are not necessarily weaknesses in TLS, but in how applications use TLS.

This Special Publication provides guidance to the selection and configuration of TLS protocol implementations while making effective use of Approved cryptographic schemes and algorithms. In particular, it requires that TLS 1.1 be configured with cipher suites using Approved schemes and algorithms as the minimum appropriate secure transport protocol[1]. It also recommends that agencies develop migration plans to TLS 1.2, configured using Approved schemes and algorithms, by January 1, 2015. When interoperability with non-government systems is required, TLS 1.0 may be supported. This Special Publication also identifies TLS extensions for which mandatory support must be provided and other recommended extensions.

Use of the recommendations provided in this Special Publication would promote:

- More consistent use of authentication, confidentiality and integrity mechanisms for the protection of information transport across the Internet;
- Consistent use of recommended cipher suites that encompass Approved algorithms and open standards;
- Protection against known and anticipated attacks on the TLS protocol; and

[1] While SSL 3.0 is the most secure of the SSL protocol versions, it is not approved for use in the protection of Federal information because it relies in part on the use of cryptographic algorithms that are not Approved. TLS versions 1.1 and 1.2 are approved for the protection of Federal information, when properly configured. TLS version 1.0 is approved only when it is required for interoperability with non-government systems and is configured according to these guidelines.

- Informed decisions by system administrators and managers in the integration of transport layer security implementations.

While these guidelines are primarily designed for Federal users and system administrators to adequately protect sensitive but unclassified U.S. Federal Government data against serious threats on the Internet, they may also be used within closed network environments to segregate data. (The client-server model and security services discussed also apply in these situations). This Special Publication supersedes NIST Special Publication 800-52. This Special Publication should be used in conjunction with existing policies and procedures.

1 Introduction

Many networked applications rely on the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols to protect sensitive data transmitted over insecure channels. The Internet's client-server model and communication protocol design principles have been described in many books, such as [Rescorla01], [Comer00], and [Hall00]. TLS requires the existence of a Public Key Infrastructure (PKI) that generates public key certificates in compliance with [RFC5280]. Books such as [Adams99] and [Housley01], as well as technical journal articles (e.g., [Polk03]) and NIST publications (e.g., [SP800-32]), describe how PKI can be used to protect information in the Internet.

This document assumes that the reader of these guidelines is familiar with public key infrastructure concepts, including, for example, X.509 certificates; and SSL and TLS protocols. The references cited above and in Appendix E further explain the background concepts that are not fully explained in these guidelines.

1.1 Background

The TLS protocol is used to secure communications in a wide variety of online transactions. Such transactions include financial transactions (i.e., banking, trading stocks, e-commerce), healthcare transactions (i.e., viewing medical records or scheduling medical appointments), and social transactions (i.e., email or social networking). Any network service that handles sensitive or valuable data, whether it is personally identifiable information (PII), financial data, or login information, needs to adequately protect that data. TLS provides a protected channel for sending data between the server and the client. The client is often, but not always, a web browser.

TLS is a layered protocol that runs on top of a reliable transport protocol – typically the transmission control protocol (TCP). Application protocols, such as HTTP and IMAP, can run above TLS. TLS is application independent, and used to provide security to any two communicating applications that transmit data over a network via an application protocol. It can be used to create a virtual private network (VPN) that connects an external system to an internal network, allowing that system to access a multitude of internal services and resources as if it were in the network.

1.2 History of TLS

The Secure Sockets Layer (SSL) protocol was designed by the Netscape Corporation[2] to meet security needs of client and server applications. Version 1 of SSL was never released. SSL 2.0 was released in 1995, but had well-known security vulnerabilities, which were addressed by the 1996 release of SSL 3.0. During this timeframe, Microsoft Corporation released a protocol known as Private Communications Technology (PCT), and later released a higher performance protocol known as the Secure Transport Layer Protocol (STLP). PCT and STLP never commanded the market share that SSL 2.0 and

---

[2] Commercial company names are used for historical reference purposes only. No product endorsement is intended or implied.

SSL 3.0 commanded. The Internet Engineering Task Force (IETF) (a technical working group responsible for developing Internet standards to ensure communications compatibility across different implementations), attempted to resolve, as best it could, security engineering and protocol incompatibility issues between the protocols. The IETF standards track Transport Layer Security Protocol Version 1.0 (TLS 1.0) emerged and was codified by the IETF as [RFC2246]. While TLS 1.0 is based on SSL 3.0, and the differences between them are not dramatic, they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate. TLS 1.0 is also referred to as SSL 3.1.

TLS 1.0 does incorporate a mechanism by which a TLS 1.0 implementation can negotiate to use SSL 3.0 with requesting entities as if TLS were never proposed. However, because SSL 3.0 is not approved for use in the protection of Federal information (Section D.9 of [FIPS140Impl]), TLS must be properly configured to ensure that the negotiation and use of SSL 3.0 never occurs when Federal information is to be protected.

TLS 1.1 was developed to address discovered weaknesses in TLS 1.0, primarily in the areas of initialization vector selection and padding error processing. Initialization vectors were made explicit[3] to prevent a certain class of attacks on the Cipher Block Chaining (CBC) mode of operation used by TLS. The handling of padding errors was altered to treat a padding error as a bad message authentication code, rather than a decryption failure – a technique that mitigates a certain class of attacks on the CBC mode of operation.

TLS 1.2 made several cryptographic enhancements, particularly in the area of hash functions, with the ability to use or specify SHA-2 family algorithms for hash, message authentication code (MAC), and Pseudorandom Function (PRF) computations. TLS 1.2 also adds support for authenticated encryption with associated data (AEAD) cipher suites.

1.3 Scope

Security is not a single property possessed by a single protocol. Rather, security includes a complex set of related properties that together provide the required information assurance characteristics and information protection services. Security requirements are usually derived from a risk assessment to the threats or attacks an adversary is likely to mount against a system. The adversary is likely to take advantage of implementation vulnerabilities found in many system components, including computer operating systems, application software systems, and the computer networks that interconnect them. Thus, in order to secure a system against a myriad of threats, security must be judiciously placed in the various systems and network layers.

These guidelines focus only on security within the network, and they focus directly on the small portion of the network communications stack that is referred to as the transport layer. Several other NIST publications address security requirements in the other parts of the systems and network layers. Adherence to these guidelines only protects the data in transit. Other applicable NIST Standards and guidelines should be used to ensure protection of systems and stored data.

[3] The IV must be sent; it cannot be derived from a state known by both parties, such as the previous message.

These guidelines focus on the common use where clients and servers must interoperate with a wide variety of implementations, and authentication is performed using public key certificates. To promote interoperability, these guidelines (and the RFCs that define the TLS protocol) establish mandatory features and cipher suites that conforming implementations must support. There are, however, much more constrained implementations of TLS servers, where security is needed, but broad interoperability is not required and the cost of implementing unused features may be prohibitive. For example, minimal servers are often implemented in embedded controllers and network infrastructure devices such as routers and then used with browsers to remotely configure and manage the devices. The use of an appropriate subset of the capabilities specified in these guidelines may be acceptable in such cases.

The scope is further limited to TLS when used in conjunction with TCP/IP. For example, Datagram TLS (DTLS) is outside the scope of these guidelines. NIST may issue separate guidelines for DTLS at a later date.

1.4 Document Conventions

Throughout this document, key words are used to identify requirements. The key words "**shall**", "**shall not**", "**should**", and "**should not**" are used. These words are a subset of the IETF Request For Comments (RFC) 2119 key words, and have been chosen based on convention in other normative documents [RFC2119]. In addition to the key words, the words "need", "can", and "may" are used in this document, but are not intended to be normative. The key word "Approved" is used to indicate that a scheme or algorithm is described in a Federal Information Processing Standard (FIPS) or is recommended by NIST.

The recommendations in this document are grouped by server recommendations and client recommendations. Section 3 provides detailed guidance for the selection and configuration of TLS servers. Section 3.9.1 summarizes guidance that applies to the selection of TLS server implementations, Section 3.9.2 summarizes guidance that applies to the configuration of TLS server implementations, and Section 3.9.3 contains guidance for system administrators that are responsible for maintaining the server. Section 4 provides detailed guidance for the selection, configuration, and use of TLS clients. Section 4.9.1 summarizes guidance that applies to the selection of TLS client implementations, Section 4.9.2 summarizes guidance that applies to the configuration of TLS client implementations, Section 4.9.3 summarizes guidance for system administrators responsible for maintaining TLS clients, and Section 4.9.4 contains guidance for end users.

2 TLS Overview

TLS exchanges records over the TLS record protocol. A TLS record contains several fields, including version information, application protocol data, and the higher-level protocol used to process the application data. TLS protects the application data by using a set of cryptographic algorithms to ensure the confidentiality, integrity, and authenticity of exchanged application data. TLS defines several protocols for connection management that sit on top of the record protocol, where each protocol has its own record type. These protocols, discussed in Section 2.1, are used to establish and change security parameters, and communicate error and warning conditions to the server and client. Sections 2.2 through 2.6 describe the security services provided by the TLS protocol and how those security services are provisioned. Section 2.7 discusses key management.

2.1 Handshake Protocol

There are three subprotocols in the TLS protocol that are used to control the session connection: the handshake, change cipher spec[4], and alert protocols. The TLS handshake protocol is used to negotiate the session parameters. The alert protocol is used to notify the other party of an error condition. The change cipher spec protocol is used to change the cryptographic parameters of a session. In addition, the client and the server exchange application data that is protected by the security services provisioned by the negotiated cipher suite. These security services are negotiated and established with the handshake.

The handshake protocol consists of a series of message exchanges between the client and the server. The handshake protocol initializes both the client and server to use optional cryptographic capabilities by negotiating a cipher suite of algorithms and functions, including key establishment, digital signature, confidentiality and integrity algorithms. Clients and servers can be configured so that one or more of the following security services are negotiated during the handshake: confidentiality, message integrity, authentication, and replay protection. A confidentiality service provides assurance that data is kept secret, preventing eavesdropping. A message integrity service provides confirmation that unauthorized data modification is detected, thus preventing undetected deletion, addition, or modification of data. An authentication service provides assurance of the sender or receiver's identity, thereby detecting forgery. Replay protection ensures that an unauthorized user does not capture and successfully replay previous data. In order to comply with these guidelines, both the client and the server **shall** be configured for data confidentiality and integrity services. Note that the anti-replay service is implicit when data contains monotonically increasing sequence number and data integrity is assured.

The handshake protocol is used to optionally exchange X.509 public key certificates[5] to authenticate the server and the client to each other. In order to comply with these

[4] In these guidelines, "change cipher spec" refers to a protocol, and "ChangeCipherSpec" refers to the message used in that protocol

[5] The use of X.509 public key certificates is fundamental to TLS. For a comprehensive explanation of X.509 public key certificates see [Adams99] or [Housley01]. In these guidelines, the terms "certificate" and "public key certificate" are used interchangeably.

guidelines, the server always presents an X.509 public key certificate that complies with the requirements stated elsewhere in these guidelines. For client-authenticated connections, the client also presents an X.509 public key certificate that complies with the requirements stated elsewhere in these guidelines.

The handshake protocol is responsible for establishing the session parameters. The client and server negotiate algorithms for authentication, confidentiality and integrity, as well as derive symmetric keys and establish other session parameters, such as data compression. The negotiated set of authentication, confidentiality, and integrity algorithms is called the cipher suite.

When all the security parameters are in place (i.e., when the handshake is complete), the ChangeCipherSpec message is used to inform the other side to begin using the negotiated security services agreed to during the handshake. All messages sent after the ChangeCipherSpec message are protected (i.e., encrypted and/or integrity protected) using the negotiated cipher suite and derived symmetric keys.

Finished messages, sent immediately following the ChangeCipherSpec messages, provide integrity checks for the handshake messages. Each Finished message is protected using the negotiated cipher suite and the derived session keys. Each side keeps a hash of all of the handshake messages exchanged up to but not including their Finished message (e.g. the Finished message sent by the server includes the Finished message sent by the client in the hash). The hash value is sent through a pseudo random function (PRF) keyed by the master secret key to form the Finished message. The receiving side decrypts the protected Finished message and compares it to its output of the PRF on the hashed messages. If the PRF values differ, the handshake has been modified or an error has occurred in the key management, and the connection is aborted. If the PRF values are the same, there is high assurance that the entire handshake has cryptographic integrity – nothing was modified, added or deleted and all key derivation was done correctly.

Alerts are used to convey information about the session, such as errors or warnings. For example, an alert can be used to signal a decryption error (decrypt_error) or that access has been denied (access_denied). Some alerts are used for warnings, and others are considered fatal and lead to immediate termination of the session. A close_notify alert message is used to signal normal termination of a session. Like all other messages after the handshake protocol is completed, alert messages are encrypted and optionally compressed.

Details of the handshake, change cipher spec and alert protocols are outside the scope of these guidelines; they are described in [RFC5246].

2.2 Shared Secret Negotiation

The client and server establish keying material during the TLS handshake protocol. The derivation of the premaster secret depends on the key exchange method that is agreed upon. For example, when RSA is used for the key exchange, the premaster secret is generated by the client and sent to the server in a ClientKeyExchange message, encrypted with the server's public key. When Diffie-Hellman is used as the key exchange algorithm, the client and server send each other their parameters, and the resulting key is used as the premaster secret. The premaster secret, along with random values exchanged by the client and server in the hello messages, is used to compute the master secret. The master secret is used to derive session keys, described in Sections 2.3 and 2.4, which are used by the negotiated security services to protect the data exchanged between the client and the server, thus providing a secure channel for the client and the server to communicate. Anti-replay protection is implicitly provided, since each packet has a monotonically increasing sequence number.

The establishment of these secrets is secure against eavesdroppers. When the TLS protocol is used in accordance with these guidelines, the application data, as well as the secrets, are not vulnerable to attackers who place themselves in the middle of the connection. The attacker cannot modify the handshake messages without being detected by the client and the server because the Finished message, exchanged after security parameter establishment, provides integrity protection to the entire exchange. In other words, an attacker cannot modify or downgrade the security of the connection by placing itself in the middle of the negotiation.

A premaster secret is securely established by the client using the RSA key transfer, Diffie-Hellman (DH or DHE) key agreement, or Elliptic Curve DH (ECDH or ECDHE).

2.3 Confidentiality

Confidentiality is provided for a communication session by the negotiated encryption algorithm for the cipher suite and the encryption keys derived from the master secret and random values, one for encryption by the client (the client write key), and another for encryption by the server (the server write key). The sender of a message (client or server) encrypts the message using a derived encryption key; the receiver uses the same key to decrypt the message. Both the client and server know these keys, and decrypt the messages using the same key that was used for encryption. The encryption keys are derived from the shared master secret.

2.4 Integrity

The keyed MAC algorithm, specified by the negotiated cipher suite, provides message integrity. Two MAC keys are derived: 1) a MAC key to be used when the client is the message sender and the server is the message receiver (the client write MAC key), and 2) a second MAC key to be used when the server is the message sender and the client is the message receiver (the server write MAC key). The sender of a message (client or server) calculates the MAC for the message using the appropriate MAC key, and encrypts both the message and the MAC using the appropriate encryption key. The sender then transmits the encrypted message and MAC to the receiver. The receiver decrypts the received message and MAC, and calculates its own version of the MAC using the MAC algorithm and sender's MAC key. The receiver verifies that the MAC that it calculates matches the MAC sent by the sender.

Two types of constructions are used for MAC algorithms in TLS. All versions of TLS support the use of HMAC using the hash algorithm specified by the negotiated cipher suite. With HMAC, MACs for server-to-client messages are keyed by the server write MAC key, while MACs client-to-server messages are keyed by the client write MAC key. These MAC keys are derived from the shared master secret.

TLS 1.2 added support for authenticated encryption with associated data (AEAD) cipher modes, such as Counter with CBC-MAC (CCM) and Galois Counter Mode (GCM), as an alternative way of providing integrity and confidentiality. In AEAD modes, the sender uses its write key for both encryption and integrity protection. The client and server write MAC keys are not used. The recipient decrypts the message and verifies the integrity information. Both the sender and the receiver use the sender's write key to perform these operations.

2.5 Authentication

Server authentication is performed by the client using the server's public key certificate, which the server presents during the handshake. The exact nature of the cryptographic operation for server authentication is dependent on the negotiated cipher suite and extensions. In most cases (e.g., RSA for key transport, DH and ECDH), authentication is performed explicitly through verification of digital signatures present in certificates, and implicitly by the use of the server public key by the client during the establishment of the master secret. A successful Finished message implies that both parties calculated the same master secret and thus, the server must have known the private key corresponding to the public key used for key establishment.

Client authentication is optional, and only occurs at the server's request. Client authentication is based on the client's public key certificate. The exact nature of the cryptographic operation for client authentication depends on the negotiated cipher suite's key exchange algorithm and the negotiated extensions. For example, when the client's public key certificate contains an RSA public key, the client signs a portion of the handshake message using the private key corresponding to that public key, and the server verifies the signature using the public key to authenticate the client.

2.6 Anti-Replay

The integrity-protected envelope of the message contains a monotonically increasing sequence number. Once the message integrity is verified, the sequence number of the current message is compared with the sequence number of the previous message. The sequence number of the current message must be greater than the sequence number of the previous message in order to further process the message.

2.7 Key Management

The server public key certificate and corresponding private key, and optionally the client public key certificate and corresponding private key, are used in the establishment of the premaster secret, according to the key exchange algorithm dictated by the selected cipher suite. The premaster secret, server random, and client random are used to determine the master secret, which is then used to derive the symmetric session keys.

The security of the server's private key is critical to the security of TLS. If the server's private key is weak or can be obtained by a third party, the third party can masquerade as the server to all clients. Similarly, if a third party can obtain a public key certificate for a public key corresponding to his own private key in the name of a legitimate server from a certification authority (CA) trusted by the clients, the third party can masquerade as the server to the clients. Requirement and recommendations to mitigate these concerns are addressed later in these guidelines.

Similar threats exist for clients. If a client's private key is weak or can be obtained by a third party, the third party can masquerade as the client to the server. Similarly, if a third party can obtain a public key certificate for a public key corresponding to his own private key in the name of a client from a CA trusted by the server, the third party can masquerade as that client to the server. Requirements and recommendations to mitigate these concerns are addressed later in these guidelines.

The server and client random values are also critical to the security of the protocol, since they form the basis for the master secret, and thus the keys used for encryption and MACs. Both the client and the server must be capable of generating pseudorandom numbers with at least 112 bits of security[6] each.[7] The various TLS session keys derived from these random values and other data are valid for the duration of the session. Because the session keys are only used to protect messages exchanged during an active TLS session, and are not used to protect any data at rest, there is no requirement for recovering TLS session keys. However, servers and clients may (and often do) cache the master secret (but not the session keys) to reduce the significant overhead in session resumption. If both the client and server have the master secret and associated session ID from a previous session in their caches, an abbreviated handshake can be used to resume the session. A resumed session uses the same negotiated parameters as the previous session, but uses new session keys derived from the master secret and new server random and client random values. After some reasonable timeout period, the master secret should be destroyed on both the server and the client. All of the state variables, including the session keys, are destroyed when the session ends. The protocol implementation relies on the operating system to ensure that there is no reuse of the keying material, such as the random values, premaster secret and session keys.

[6] Bits of security provided by Approved algorithms are described in SP 800-57 part 1 [SP800-57p1], Section 5.6.

[7] While the client and server each generate 256-bit (32-byte) random values, 112 bits of security is considered sufficient until 2030.

3 Minimum Requirements for TLS Servers

This section provides a minimum set of requirements that a server must implement in order to meet these guidelines. Requirements are organized in the following sections: TLS protocol version support; server keys and certificates; cryptographic support; TLS extension support; client authentication; session resumption; compression methods; and operational considerations.

Specific requirements are stated as either implementation requirements or configuration requirements. Implementation requirements indicate that Federal agencies **shall not** procure TLS server implementations unless they include the required functionality, or can be augmented with additional commercial products to meet requirements. Configuration requirements indicate that TLS server administrators are required to verify that particular features are enabled, or in some cases, configured appropriately, if present.

3.1 Protocol Version Support

TLS version 1.1 is required, at a minimum, in order to mitigate various attacks on version 1.0 of the TLS protocol. Support for TLS version 1.2 is strongly recommended.

Servers that support government-only applications **shall** be configured to support TLS 1.1, and **should** be configured to support TLS 1.2. These servers **shall not** support TLS 1.0 or any version of SSL. TLS versions 1.1 and 1.2 are represented by major and minor number tuples (3, 2) and (3, 3), respectively[8]. Agencies **shall** develop migration plans to support TLS 1.2 by January 1, 2015.

Servers that support citizen or business-facing applications **shall** be configured to support version 1.1 and **should** be configured to support version 1.2. These servers may also be configured to support TLS version 1.0 in order to enable interaction with citizens and businesses. These servers **shall not** support SSL version 3.0 or earlier. If TLS 1.0 is supported, the use of TLS 1.1 and 1.2 **shall** be preferred over TLS 1.0.

Some server implementations are known to implement version negotiation incorrectly. For example, there are TLS 1.0 servers that terminate the connection when the client offers a version newer than TLS 1.0. Servers that incorrectly implement TLS version negotiation **shall not** be used.

3.2 Server Keys and Certificates

The TLS server **shall** be configured with one or more public key certificates and the associated private keys. TLS server implementations **should** support multiple server certificates with their associated private keys to support algorithm and key size agility.

There are six options for TLS server certificates that can satisfy the requirement for Approved cryptography: an RSA key encipherment certificate; an RSA signature

[8] Historically TLS 1.0 was assigned major, minor tuple (3,1) to align it as SSL 3.1.

certificate; an ECDSA signature certificate; a DSA[9] signature certificate; a Diffie-Hellman certificate; and an ECDH certificate.

At a minimum, TLS servers conforming to this specification **shall** be configured with an RSA key encipherment certificate, and also **should** be configured with an ECDSA signature certificate or RSA signature certificate. If the server is not configured with an RSA signature certificate, an ECDSA signature certificate using a Suite B named curve for the signature and public key in the ECDSA certificate **should** be used.[10]

TLS servers **shall** be configured with certificates issued by a CA, rather than self-signed certificates. Furthermore, TLS server certificates **shall** be issued by a CA that publishes revocation information in either a Certificate Revocation List (CRL) [RFC5280] or in Online Certificate Status Protocol (OCSP) [RFC6960] responses. The source for the revocation information **shall** be included in the CA-issued certificate in the appropriate extension to promote interoperability.

A TLS server that has been issued certificates by multiple CAs can select the appropriate certificate, based on the client specified "Trusted CA Keys" TLS extension, as described in Section 3.4.1.4. A TLS server that has been issued certificates for multiple names can select the appropriate certificate, based on the client specified "Server Name" TLS extension, as described in Section 3.4.1.3. A TLS server may also contain multiple names in the Subject Alternative Name extension of the server certificate in order to support multiple server names of the same name form (e.g., DNS Name) or multiple server names of multiple name forms (e.g., DNS Names, IP Address, etc.)

Section 3.2.1 specifies a detailed profile for server certificates. Basic guidelines for DSA, DH, and ECDH certificates are provided; more detailed profiles may be provided if these algorithms experience broad use in the future. Section 3.2.2 specifies requirements for revocation checking. System administrators **shall** use these sections to identify an appropriate source for certificates. Section 3.5.4 specifies requirements for the "hints list."

3.2.1 Server Certificate Profile

The server certificate profile, described in this section, provides requirements and recommendations for the format of the server certificate. For these guidelines, the TLS server certificate **shall** be an X.509 version 3 certificate; both the public key contained in the certificate and the signature **shall** have at least 112 bits of security. The certificate **shall** be signed with an algorithm consistent with the public key[11]:

- Certificates containing RSA (key encipherment or signature), ECDSA, or DSA public keys **shall** be signed with those same signature algorithms, respectively;

[9] In the names for the TLS cipher suites, DSA is referred to as DSS, for historical reasons.

[10] The Suite B curves are known as P-256 and P-384. These curves are defined in [FIPS186-4] and their inclusion in Suite B is documented in [RFC6460].

[11] Algorithm-dependent rules exist for the generation of public and private key pairs. For guidance on the generation of DH and ECDH key pairs, see [SP800-56A]. For guidance regarding the generation of RSA key pairs, see [SP800-56B]. For guidance regarding the generation of DSA and ECDSA key pairs, see [FIPS186-4].

- Certificates containing Diffie-Hellman public keys **shall** be signed with DSA; and
- Certificates containing ECDH public keys **shall** be signed with ECDSA.

The extended key usage extension limits the operations that keys in a certificate may be used for. There is an extended key usage extension specifically for server authentication, and the server **should** be configured to support it. The use of the extended key usage extension will facilitate successful server authentication, as some clients may require the presence of an extended key usage extension. The extended key usage extension will also indicate that the certificate is not intended to be used for other purposes, such as code signing. The use of the server DNS name in the Subject Alternative Name field ensures that any name constraints on the certification path will be properly enforced.

The server certificate profile is listed in Table 3-1. In the absence of agency-specific certificate profile requirements, this certificate profile **should** be used for the server certificate.

Note that for ECDH, the algorithm OID and the signature OID are identical to those of ECDSA. For interoperability reasons, algorithm OID is not changed and the key usage extension determines if the public key is used for key agreement or signature verification.

Table 3-1: TLS Server Certificate Profile

Field	Critical	Value	Description
Version	N/A	2	Version 3
Serial Number	N/A	Unique positive integer	Must be unique
Issuer Signature Algorithm	N/A	*Values by certificate type:*	
		sha256WithRSAEncryption {1 2 840 113549 1 1 11}, or stronger	RSA key encipherment certificate, RSA signature certificate
		ecdsa-with-SHA256 {1 2 840 10045 4 3 2}, or stronger	ECDSA signature certificate, ECDH certificate
		id-dsa-with-sha256 {2 16 840 1 101 3 4 3 2}, or stronger	DSA signature certificate, DH certificate
Issuer Distinguished Name	N/A	Unique X.500 Issuing CA DN	Single value shall be encoded in each RDN. All attributes that are of directoryString type shall be encoded as a printable string.
Validity Period	N/A	3 years or less	Dates through 2049 expressed in UTCTime
Subject Distinguished Name	N/A	Unique X.500 subject DN per agency requirements	Single value shall be encoded in each RDN. All attributes that are of directoryString type shall be encoded as a printable string. CN={ Host URL | Host IP Address | Host DNS Name }

Field	Critical	Value	Description
Subject Public Key Information	N/A	*Values by certificate type:*	
		rsaEncryption {1 2 840 113549 1 1 1}	RSA key encipherment certificate, RSA signature certificate 2048 bit RSA key modulus Parameters: NULL
		ecPublicKey {1 2 840 10045 2 1}	ECDSA signature certificate, or ECDH certificate Parameters: namedCurve OID for names curve specified in FIPS 186-4. The curve **shall** be P-256 or P-384 SubjectPublic Key: Uncompressed EC Point.
		id-dsa {1 2 840 10040 4 1}	DSA signature certificate Parameters: p, q, g
		dhpublicnumber {1 2 840 10046 2 1}	DH certificate Parameters: p, g, q
Issuer's Signature	N/A	*Values by certificate type:*	
		sha256WithRSAEncryption {1 2 840 113549 1 1 11}, or stronger	RSA key encipherment certificate, RSA signature certificate
		ecdsa-with-SHA256 {1 2 840 10045 4 3 2}, or stronger	ECDSA signature certificate, ECDH certificate
		id-dsa-with-sha256 { 2 16 840 1 101 3 4 3 2}, or stronger	DSA signature certificate, DH certificate
Extensions			
Authority Key Identifier	No	Octet String	Same as subject key identifier in Issuing CA certificate Prohibited: Issuer DN, Serial Number tuple
Subject Key Identifier	No	Octet String	Same as in PKCS-10 request or calculated by the Issuing CA
Key Usage	Yes	*Values by certificate type:*	
		keyEncipherment	RSA key encipherment certificate
		digitalSignature	RSA signature certificate, ECDSA signature certificate, or DSA signature certificate
		keyAgreement	ECDH certificate, DH certificate
Extended Key Usage	No	id-kp-serverAuth {1 3 6 1 5 5 7 3 1}	Required
		id-kp-clientAuth {1 3 6 1 5 5 7 3 2}	Optional
			Prohibited: anyExtendedKeyUsage, all others unless consistent with key usage extension
Certificate Policies	No	Per agency X.509 certificate policy	
Subject Alternative Name	No	DNS Host Name or IP Address if there is no DNS name assigned	Multiple SANs are permitted, e.g., for load balanced environments.
Authority Information Access	No	id-ad-caIssuers	Required. Access method entry contains HTTP URL for certificates issued to Issuing CA
		id-ad-ocsp	Optional. Access method entry contains HTTP URL for the Issuing CA OCSP Responder
CRL Distribution Points	No	See comments	Optional: HTTP value in distributionPoint field pointing to a full and complete CRL. Prohibited: reasons and cRLIssuer fields, and nameRelativetoCRLIssuer CHOICE

3.2.2 Obtaining Revocation Status Information for the Client Certificate

The server **shall** perform revocation checking of the client certificate, when client authentication is used. Revocation information can be obtained by the server from one of the following locations:

1. Certificate Revocation List (CRL) or OCSP [RFC6960] response in the server's local store;
2. OCSP response from a locally configured OCSP Responder;
3. OCSP response from the OCSP Responder location identified in the OCSP field in the Authority Information Access extension in the client certificate; or
4. CRL from the CRL Distribution Point extension in the client certificate.

When the local store does not have the current or a cogent CRL or OCSP response, and the OCSP Responder and the CRL Distribution Point are unavailable or inaccessible at the time of TLS session establishment, the server will either deny the connection or accept a potentially revoked or compromised certificate. The decision to accept or reject a revoked certificate **should** be made according to agency policy.

3.2.3 Server Public Key Certificate Assurance

After the server public key certificate has been verified by a client, it may be trusted by the client on the basis of policies, procedures and security controls used to issue the server public key certificate. The server is required to possess an X.509 version 3 public key certificate. The policy, procedures and security controls are optionally represented in the certificate using the certificatePolicies extension, specified in [RFC5280] and updated in [RFC6818]. When used, one or more certificate policy OIDs are asserted in this extension. The actual policies and procedures and security controls associated with each certificate policy OID are documented in a certificate policy. In the absence of agency-specific policies, Federal agencies **shall** use the Common Policy [COMMON].

The use of a certificate policy that is designed with the secure operation of PKI in mind and adherence to the stipulated certificate policy mitigates the threat that the issuing CA can be compromised or that the registration system, persons or process can be compromised to obtain an unauthorized certificate in the name of a legitimate entity, and thus compromise the clients. With this in mind, the CA Browser Forum, a private sector organization, has carried out some efforts in this area. The guideline was first published as the Extended Validation guideline [EVGUIDE]. Under another effort, the CA Browser Forum published requirements for issuing certificates from publicly trusted CAs in order for those CAs and their trust anchor to remain in browser trust stores [CABBASE].

It should be noted that there are TLS clients that do not perform X.509 certificate policy processing as mandated by [RFC5280]. Thus, they are not able to accept or reject a TLS server certificate based on the assurance level specified by the policy. This may result in the acceptance of a fraudulent certificate and may expose user data to unintended parties. The Federal Government and CA Browser Forum hope that the security requirements in

[COMMON], [EVGUIDE], and [CABBASE] are adopted by all CAs under their purview, mitigating the lack of a policy processing capability.

In order to further mitigate the risk associated with a CA or X.509 certificate registration system, process or personnel compromise, several concepts are under development. These emerging concepts are further discussed in Appendix D.

3.3 Cryptographic Support

Cryptographic support in TLS is provided through the use of various cipher suites. A cipher suite specifies a collection of algorithms for key exchange and for providing confidentiality and integrity services to application data. The cipher suite negotiation occurs during the TLS handshake protocol. The client presents cipher suites that it supports to the server, and the server selects one of them to secure the session data.

Cipher suites have the form:

TLS_*KeyExchangeAlg*_WITH_*EncryptionAlg*_*MessageAuthenticationAlg*

For example, the cipher suite TLS_RSA_WITH_AES_128_CBC_SHA uses RSA for the key exchange, AES-128 in cipher block chaining mode for encryption, and message authentication is performed using HMAC_SHA[12]. For further information on cipher suite interpretation, see Appendix B.

3.3.1 Cipher Suites

The server **shall** be configured to only use cipher suites that are composed entirely of Approved algorithms. A complete list of acceptable cipher suites for general use is provided in this section, grouped by certificate type and TLS protocol version.

In some situations, such as closed environments, it may be appropriate to used pre-shared keys. Pre-shared keys are symmetric keys that are already in place prior to the initiation of a TLS session, which are used in the derivation of the premaster secret. For cipher suites that are acceptable in pre-shared key environments, see Appendix C.

In order to maximize interoperability, TLS server implementations **shall** support the following cipher suites:

- TLS_RSA_WITH_3DES_EDE_CBC_SHA[13]
- TLS_RSA_WITH_AES_128_CBC_SHA[14]

In addition, TLS server implementations **should** support the following cipher suites:

- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

[12] SHA indicates the use of the SHA-1 hash algorithm.

[13] Support of this cipher suite is mandatory for TLS 1.1 [RFC4346]

[14] Support of this cipher suite is mandatory for TLS 1.2 [RFC5246]

TLS version 1.2 adds support for authenticated encryption modes, and support for the SHA-256 and SHA-384 hash algorithms, which are not supported in prior versions of TLS. These cipher suites are described in [RFC5288] and [RFC5289]. In addition to supporting the cipher suites listed above, TLS 1.2 servers **shall** be configured to support the following cipher suite:

- TLS_RSA_WITH_AES_128_GCM_SHA256

TLS 1.2 servers **should** be configured to support the following cipher suites:

- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECHDE_RSA_WITH_AES_128_GCM_SHA256

NIST may define additional mandatory or recommended cipher suites at a later date.

The server **shall** be configured to only support cipher suites for which it has a valid certificate containing a signature providing at least 112 bits of security. The following cipher suite tables are grouped by certificate type and TLS protocol version. The cipher suites in these tables include the cipher suites that **shall** and **should** be supported (as described above), and may be supported. Only cipher suites that are composed of Approved algorithms are acceptable and are listed in this section. The server **shall not** be configured to support cipher suites that do not appear in these tables, unless otherwise stated by agency-specific policies. Cipher suites that do not appear in this section or Appendix C **should not** be used.

In the following tables listing recommended cipher suites, cipher suites shown in bold font **shall** be supported, cipher suites shown in italics **should** be supported, and all others may be supported.

Table 3-2 identifies the recommended cipher suites for a TLS server that has been configured with an RSA private key and a corresponding RSA certificate. Table 3-3 identifies additional acceptable RSA cipher suites that are supported by TLS version 1.2. A server having a RSA certificate may support any cipher suite that appears in Table 3-2 or Table 3-3. The key usage extension in the RSA certificate **shall** specify key encipherment for cipher suites that use RSA key transport to carry out the key exchange, and the key usage extension **shall** specify digital signature for cipher suites using ECDHE for key exchange.

Table 3-2: Cipher Suites for RSA Server Certificates

Cipher Suite Name	Key Exchange	Encryption	Hash Function for HMAC	Hash Function for PRF[15]
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES_EDE_CBC	SHA-1	Per RFC
TLS_RSA_WITH_AES_128_CBC_SHA	RSA	AES_128_CBC	SHA-1	Per RFC
TLS_RSA_WITH_AES_256_CBC_SHA	*RSA*	*AES_256_CBC*	*SHA-1*	*Per RFC*

[15] In TLS versions 1.0 and 1.1, the hash function used in the PRF is a parallel application of MD5 and SHA-1, as defined in [RFC2246] and [RFC4346]. For TLS 1.2, the PRF hash function is SHA-256, unless otherwise stated.

Cipher Suite Name	Key Exchange	Encryption	Hash Function for HMAC	Hash Function for PRF[15]
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	ECDHE	3DES_EDE_CBC	SHA-1	Per RFC
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	ECDHE	AES_128_CBC	SHA-1	Per RFC
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	ECDHE	AES_256_CBC	SHA-1	Per RFC

Table 3-3: Additional TLS 1.2 Cipher Suites for RSA Server Certificates

Cipher Suite Name	Key Exchange	Encryption	Hash Function for HMAC	Hash Function for PRF
TLS_RSA_WITH_AES_128_GCM_SHA256	RSA	AES_128_GCM	N/A	SHA-256
TLS_RSA_WITH_AES_256_GCM_SHA384	RSA	AES_256_GCM	N/A	SHA-384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	ECDHE	AES_128_CBC	N/A	SHA-256
TLS_ECHDE_RSA_WITH_AES_128_GCM_SHA256	ECDHE	AES_128_GCM	N/A	SHA-256
TLS_RSA_WITH_AES_128_CBC_SHA256	RSA	AES_128_CBC	SHA-256	SHA-256
TLS_RSA_WITH_AES_256_CBC_SHA256	RSA	AES_256_CBC	SHA-256	SHA-256
TLS_RSA_WITH_AES_128_CCM[16]	RSA	AES_128_CCM	N/A	SHA-256
TLS_RSA_WITH_AES_256_CCM	RSA	AES_256_CCM	N/A	SHA-256

Table 3-4 identifies the recommended cipher suites for a TLS server that has been configured with an elliptic curve private key and a corresponding ECDSA certificate. These cipher suites are described in [RFC4492]. Table 3-5 identifies additional acceptable ECDSA cipher suites, described in [RFC5289], that are supported by TLS version 1.2. A server that is configured with an ECDSA certificate may support any of the cipher suites listed in Table 3-4 or Table 3-5.

Table 3-4: Cipher Suites for ECDSA Server Certificates

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	ECDHE	3DES_EDE_CBC	SHA-1	Per RFC
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	ECDHE	AES_128_CBC	SHA-1	Per RFC
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	ECDHE	AES_256_CBC	SHA-1	Per RFC

Table 3-5: Additional TLS 1.2 Cipher Suites for ECDSA Server Certificates

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	ECDHE	AES_128_CBC	SHA-256	SHA-256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	ECDHE	AES_256_CBC	SHA-384	SHA-384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ECDHE	AES_128_GCM	N/A	SHA-256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDHE	AES_256_GCM	N/A	SHA-384

DHE is the preferred Diffie-Hellman key exchange algorithm, as it provides perfect forward secrecy[17]. Table 3-6 identifies acceptable cipher suites for a server that has been

[16] AES-CCM cipher suites are defined in [RFC6655].

configured with a DSA private key and a corresponding DSA certificate. Table 3-7 identifies additional acceptable DSA cipher suites supported by TLS version 1.2. A server that is configured with a DSA certificate may support any of the cipher suites listed in Table 3-6 or Table 3-7.

Table 3-6: Cipher Suites for DSA Server Certificates

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DHE	3DES_EDE_CBC	SHA-1	Per RFC
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	DHE	AES_128_CBC	SHA-1	Per RFC
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	DHE	AES_256_CBC	SHA-1	Per RFC

Table 3-7: Additional TLS 1.2 Cipher Suites for DSA Server Certificates

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256	DHE	AES_128_CBC	SHA-256	SHA-256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256	DHE	AES_256_CBC	SHA-256	SHA-256
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256	DHE	AES_128_GCM	N/A	SHA-256
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384	DHE	AES_256_GCM	N/A	SHA-384

Table 3-8 identifies acceptable cipher suites for a server that has been configured with a DH private key and a corresponding DH certificate signed using DSA. Table 3-9 identifies acceptable additional DH cipher suites supported by TLS version 1.2 [RFC5246], [RFC5288].

Table 3-8: Cipher Suites for DH Server Certificates

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH	3DES_EDE_CBC	SHA-1	Per RFC
TLS_DH_DSS_WITH_AES_128_CBC_SHA	DH	AES_128_CBC	SHA-1	Per RFC
TLS_DH_DSS_WITH_AES_256_CBC_SHA	DH	AES_256_CBC	SHA-1	Per RFC

Table 3-9: Additional TLS 1.2 Cipher Suites for DH Server Certificates

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_DH_DSS_WITH_AES_128_CBC_SHA256	DH	AES_128_CBC	SHA-256	SHA-256
TLS_DH_DSS_WITH_AES_256_CBC_SHA256	DH	AES_256_CBC	SHA-256	SHA-256
TLS_DH_DSS_WITH_AES_128_GCM_SHA256	DH	AES_128_GCM	N/A	SHA-256
TLS_DH_DSS_WITH_AES_256_GCM_SHA384	DH	AES_256_GCM	N/A	SHA-384

Table 3-10 identifies acceptable cipher suites that may be used for a server that has been configured with an elliptic curve private key and a corresponding ECDH certificate

[17] Perfect forward secrecy is the condition in which the compromise of a long-term private key used in deriving a session key subsequent to the derivation does not cause the compromise of the session key.

signed using ECDSA. Table 3-11 identifies additional acceptable ECDH cipher suites supported by TLS 1.2 that may be used. These cipher suites are defined in [RFC5289].

Table 3-10: Cipher Suites for ECDH Server Certificate

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA	ECDH	3DES_EDE_CBC	SHA-1	Per RFC
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	ECDH	AES_128_CBC	SHA-1	Per RFC
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	ECDH	AES_256_CBC	SHA-1	Per RFC

Table 3-11: Additional TLS 1.2 Cipher Suites for ECDH Server Certificate

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	ECDH	AES_128_CBC	SHA-256	SHA-256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	ECDH	AES_256_CBC	SHA-384	SHA-384
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	ECDH	AES_128_GCM	N/A	SHA-256
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	ECDH	AES_256_GCM	N/A	SHA-384

Appendix B provides further details on cipher suite name interpretation. While the cipher suite name is used in descriptions, the actual protocol uses assigned numbers to identify cipher suites.

When negotiating a cipher suite, the client sends a handshake message with a list of cipher suites it will accept. The server chooses from the list and sends a handshake message back indicating which cipher suite it will accept. Although the client may order the list with the strongest cipher suites listed first, the server may choose <u>*any*</u> of the cipher suites proposed by the client. Therefore there is <u>*no*</u> guarantee that the negotiation will settle on the strongest suite in common. If no cipher suites are in common the connection is aborted.

Cipher suites using ephemeral DH and ephemeral ECDH (i.e., those with DHE or ECDHE in the second mnemonic) provide perfect forward secrecy, ensuring long-term confidentiality of the session. While support of these cipher suites is not required by these guidelines, it is strongly recommended.

There is no mechanism to specify the minimum key size for the server or client certificate or for the CAs that are in the certification path.

3.3.1.1 Implementation Considerations

System administrators need to fully understand the ramifications of selecting cipher suites and configuring applications to support only those cipher suites. The security guarantees of the cryptography are limited to the weakest cipher suite supported by the configuration. When configuring an implementation, there are several factors that affect supported cipher suite selection.

3.3.1.1.1 Algorithm Support

Most TLS servers and clients support RC4 [Schneier96] cipher suites. RC4 is not an Approved algorithm. If the server were configured to support RC4 cipher suites, they may be chosen over the recommended cipher suites composed of Approved algorithms. Therefore it is important that the server is configured only to use recommended cipher suites.

Server implementations may not allow the server administrator to specify preference order. In such servers, the only way to ensure that a server uses Approved algorithms for encryption is to disable cipher suites that use other encryption algorithms (such as RC4 and Camellia [RFC3713]).

3.3.1.1.2 Cipher Suite Scope

The selection of a cryptographic algorithm may be system-wide and not application specific for some implementations. For example, disabling an algorithm for one application on a system might disable that algorithm for all applications on that system.

3.3.2 Validated Cryptography

The cryptographic module used by the server **shall** be a FIPS 140-validated cryptographic module. All cryptographic algorithms that are included in the configured cipher suites **shall** be within the scope of the validation, as well as the random number generator. Note that the TLS 1.1 pseudorandom function (PRF) uses MD5 and SHA-1 in parallel so that if one hash function is broken, security is not compromised. While MD5 is not an Approved algorithm, the TLS 1.1 PRF is specified as acceptable in [FIPS140Impl] and [SP800-135]. In TLS 1.2, the hash function is either SHA-256 or is indicated by the cipher suite and must be at least as strong as SHA-256.

The random number generator **shall** be tested and validated in accordance with [SP800-90A] under the NIST Cryptographic Algorithm Validation Program (CAVP) and successful results of this testing **shall** be indicated on the cryptographic module's FIPS 140 validation certificate. The validated random number generator **shall** be used to generate the server random value used in the TLS protocol.

3.4 TLS Extension Support

Several TLS extensions are described in [RFC6066]. Servers are encouraged to support these extensions, except where discouraged as specified in Section 3.4.3. Additional extensions are described in [RFC4492], [RFC5246], and [RFC5746]. This section contains recommendations for a subset of the TLS extensions that the Federal agencies **shall**, **should**, or **should not** use as they become prevalent in commercially available TLS servers and clients.

Some servers will refuse the connection if any TLS extensions are included in the ClientHello message. Interoperability with servers that do not properly handle TLS extensions may require multiple connection attempts by the client.

3.4.1 Mandatory TLS Extensions

The server **shall** support the following TLS extensions.

1. Renegotiation Indication

2. Certificate Status Request
3. Server Name Indication
4. Trusted CA Indication

3.4.1.1 Renegotiation Indication

TLS session renegotiation is vulnerable to an attack in which the attacker forms a TLS connection with the target server, injects content of his choice, and then splices in a new TLS connection from a legitimate client. The server treats the legitimate client's initial TLS handshake as a renegotiation of the attacker's negotiated session and thus believes that the initial data transmitted by the attacker is from the legitimate client. The session renegotiation extension is defined to prevent such a session splicing or session interception. The extension uses the concept of cryptographically binding the initial session negotiation and session renegotiation.

Servers **shall** perform initial and subsequent renegotiations in accordance with [RFC5746].

3.4.1.2 Certificate Status Request

When the client wishes to receive the revocation status of the TLS server certificate from the TLS server, the client includes the Certificate Status Request (status_request) extension in the ClientHello message. The server **should** include the certificate status along with its certificate by sending a CertificateStatus message immediately following the Certificate message. While the extension itself is extensible, only OCSP type certificate status is defined in [RFC6066]. This extension is also called OCSP stapling.

3.4.1.3 Server Name Indication

Multiple virtual servers may exist at the same network address. The server name indication extension allows the client to specify which of the servers located at the address it is trying to connect with. The server **shall** be able to process and respond to the server name indication extension received in a ClientHello message as described in [RFC6066].

3.4.1.4 Trusted CA Indication

The trusted CA indication (trusted_ca_keys) extension allows a client to specify which CA root keys it possesses. This is useful for sessions where the client is memory-constrained and possesses a small number of root CA keys. The server **shall** be able to process and respond to the trusted CA indication extension received in a ClientHello message as described in [RFC6066].

3.4.2 Conditional TLS Extensions

A TLS server may be able to support the following TLS extensions under the circumstances described in the following paragraphs:

1. The Supported Elliptic Curves TLS extension **shall** be supported if the server supports EC cipher suite(s).
2. The EC Point Format TLS extension **shall** be supported if the server supports EC cipher suite(s).

3. The Signature Algorithms TLS extension **shall** be supported when the server is operating in TLS 1.2.
4. The Multiple Certificate Status extension **shall** be supported if the extension is supported by the server implementation.
5. The Truncated HMAC extension may be supported if the server communicates with constrained device clients and the server implementation does not support variable-length padding.

3.4.2.1 Supported Elliptic Curves

Servers that support elliptic curve cipher suites **shall** be able to process the elliptic curves received in the ClientHello message. The curves P-256 and P-384 **shall** be supported. The servers **shall** process this extension in accordance with Section 5.1 of [RFC4492].

3.4.2.2 EC Point Format

The servers that support EC cipher suites **shall** be able to process the supported EC point format received in the ClientHello message by the client. The servers **shall** process this extension in accordance with Section 5.1 of [RFC4492].

The servers that support EC cipher suites **shall** also be able to send the supported EC point format in the ServerHello message as described in Section 5.2 of [RFC4492].

3.4.2.3 Signature Algorithms

The servers that support TLS 1.2 **shall** support the processing of the signature algorithms extension received in a ClientHello message. The extension, its syntax, and processing rules are described in Sections 7.4.1.4.1, 7.4.2, and 7.4.3 of [RFC5246].

3.4.2.4 Multiple Certificate Status

The multiple certificate status extension improves on the Certificate Status Request extension described in Section 3.4.1.2 by allowing the client to request the status of all certificates provided by the server in the TLS handshake. When the server returns the revocation status of all the certificates in the server certificate chain, the client does not need to query any revocation service providers, such as OCSP responders. This extension is documented in [RFC6961]. Server implementations that have this capability **shall** be configured to support this extension.

3.4.2.5 Truncated HMAC

The Truncated HMAC extension allows a truncation of the HMAC output to 80 bits for use as a MAC tag. An 80-bit MAC tag complies with the recommendations in [SP800-107], but reduces the security provided by the integrity algorithm. Because forging a MAC tag is an online attack, and the TLS session will terminate immediately when an invalid MAC tag is encountered, the risk introduced by supporting this extension is low. However, truncated MAC tags **shall not** be used in conjunction with variable-length padding, due to attacks described in [Paterson11].

3.4.3 Discouraged TLS Extensions

The following extensions **should not** be used:

1. Client Certificate URL

The Client Certificate URL extension allows a client to send a URL pointing to a certificate, rather than sending a certificate to the server during mutual authentication. This can be very useful for mutual authentication with constrained clients. However, this extension can be used for malicious purposes. The URL could belong to an innocent server on which the client would like to perform a denial of service attack, turning the TLS server into an attacker. A server that supports this extension also acts as a client while retrieving a certificate, and therefore becomes subject to additional security concerns. For these reasons, the Client Certificate URL extension **should not** be supported. However, if an agency determines the risks to be minimal, and this extension is needed for environments where clients are in constrained devices, the extension may be supported. If the client certificate URL extension is supported, the server **shall** be configured to mitigate the security concerns described above and in Section 11.3 of [RFC6066].

3.5 Client Authentication

Where strong cryptographic client authentication is required, TLS servers may use the TLS protocol client authentication option to request a client certificate to cryptographically authenticate the client. For example, the PIV Authentication Certificate [FIPS201-1] (and the associated private key) provides a suitable option for strong authentication of Federal employees and contractors with on-site access. To ensure that agencies are positioned to take full advantage of the PIV card, all TLS servers that perform client authentication **shall** support certificate-based client authentication.

The client authentication option requires the server to implement the X.509 path validation mechanism and a trust anchor store. Requirements for these mechanisms are specified in Sections 3.5.1 and 3.5.2, respectively. To ensure that cryptographic authentication actually results in strong authentication, client keys **shall** contain at least 112 bits of security. Section 3.5.3 describes mechanisms that can contribute, albeit indirectly, to enforcing this requirement. Section 3.5.4 describes the client's use of the server hints list.

The TLS server **shall** be configurable to terminate the connection with a fatal "handshake failure" alert when a client certificate is requested, and the client does not have a suitable certificate.

3.5.1 Path Validation

The client certificate **shall** be validated in accordance with the certification path validation rules specified in Section 6 of [RFC5280]. In addition, the revocation status of each certificate in the certification path **shall** be validated using a Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP). OCSP checking **shall** be in compliance with [RFC6960] and **should** use only one of the following options:

- The OCSP Responder is trusted by the server, i.e., the OCSP Responder public key is the same as that of one of the public keys in the server's trust anchor store; or
- The OCSP Response is signed using the same key as for the certificate whose status is being checked; or

- The OCSP Response is signed by a designated/delegated OCSP Responder as described in [RFC6960], and the OCSP Responder certificate is signed using the same key as for the certificate whose status is being checked.

Revocation information **shall** be obtained as described in Section 3.2.2.

Federal agencies **shall** perform a risk assessment to determine acceptable grace periods for revocation information, as well as whether a grace period should be applied to the time found in the “thisUpdate” or “nextUpdate” field. If the determined grace period has elapsed relative to the selected time field, then the revocation information **shall** be considered stale, and the stale revocation information **shall not** be used to determine the validity of the certificate. If fresh revocation information cannot be obtained through another source, the certificate **shall** be considered invalid.

The server **shall** be able to determine the certificate policies that the client certificate is trusted for by using the certification path validation rules specified in Section 6 of [RFC5280]. Server and backend applications may use this determination to accept or reject the certificate. Checking certificate policies assures the server that only client certificates that have been issued with acceptable assurance, in terms of CA and registration system and process security, are accepted.

Not all commercial products may support the public key certification path validation and certificate policy processing rules listed and cited above. When implementing client authentication, the Federal agencies **shall** either use the commercial products that meet these requirements or augment commercial products to meet these requirements.

The server **shall** be able to provide the client certificate, and the certificate policies for which the client certification path is valid, to the applications in order to support access control decisions.

3.5.2 Trust Anchor Store

Having an excessive number of trust anchors installed in the TLS application can expose the application to all the PKIs emanating from these trust anchors. The best way to minimize the exposure is to only include the trust anchors in the trust anchor store that are absolutely necessary for client public key certificate authentication.

The server **shall** be configured with only the trust anchors that the server trusts, and of those, only the ones that are required to authenticate the clients, in the case where the server supports client authentication in TLS. These trust anchors are typically a small subset of the trust anchors that may be included on the server by default. Also note that this trust anchor store is distinct from the machine trust anchor store. Thus, the default set of trust anchors **shall** be examined to determine if any of them are required for client authentication. Some specific enterprise and/or PKI service provider trust anchor may need to be added.

In the U.S. Federal environment, in most situations, the Federal Common Policy Root or the Agency Root (if cross certified with the Federal Bridge Certification Authority) should be sufficient to build a certification path to the client certificates.

System administrators of a TLS server that supports certificate-based client authentication **shall** perform an analysis of the client certificate issuers and use that information to determine the minimum set of trust anchors required for the server. The server **shall** be configured only to include those trust anchors.

3.5.3 Checking the Client Key Size

The only direct mechanism for a server to check whether the key size and algorithms presented in a client public certificate are acceptable is for the server to examine the public key and algorithm in the client's certificate. An indirect mechanism is to check that the certificate policies extension in the client public key certificate indicates the minimum cryptographic strength of the signature and hashing algorithms used, and for the server to perform certificate policy processing and checking. A more scalable and more robust alternative that is standards-based, but has not gained widespread commercial deployment, is described in Appendix D. The server **shall** check the client key length if client authentication is performed, and the server implementation provides a mechanism to do so.

3.5.4 Server Hints List

Clients may use the list of trust anchors sent by the server in the CertificateRequest message to determine if the client's certification path terminates at one of these trust anchors. The list sent by the server is known as a "hints list." When the server and client are in different PKI domains, and the trust is established via direct cross certification between the two PKI domains (i.e., the server PKI domain and the client PKI domain) or via transitive cross certification (i.e., through cross certifications among multiple PKI domains), the client may erroneously decide that its certificate will not be accepted by the server, since the client's trust anchor is not sent in the hints list. To mitigate this failure, the server **shall** maintain the trust anchors of the various PKIs whose subscribers are the potential clients for the server, and include them in the hints list. Alternatively, the server **should** be configured to send an empty hints list so that the client can always provide a certificate it possesses. However, this list **shall** be distinct from the server's trust anchor store. In other words, the server **shall** continue to only populate its trust anchor store with the trust anchor of the server's PKI domain and the domains it needs to trust directly for client authentication. Note that the distinction between the server hints list and the server's own trust store are the trust anchors of PKI domains that the server trusts only through the cross certificates issued by the trust anchors in the server's trust store.

3.6 Session Resumption

During the initial handshake between the client and server, the server generates a session identifier (ID) and passes this value to the client during the handshake. Both the server and client store the session ID (along with the keying material and cipher suite) after completion of the handshake for later use. If the server is willing to resume a session at the request of a client, the server responds with the original session ID and cipher suite at the start of the handshake. In the event that the server is unwilling to resume the session, the server generates and responds with a new session ID.

Typical server implementations are agreeable to resuming a previous session. This is a secure mode of operation, as the master secret is known only to the client and server, and is coupled with the initial client authentication, if client authentication was required.

However, if there is a requirement to authenticate each client as it initiates a connection session, the server **shall** be configured to ignore requests to resume a session, and generate a new session ID, which forces the entire handshake procedure (including client authentication) to proceed.

3.7 Compression Methods

The use of compression may enable attackers to perform attacks using compression-based side channels. Because of this, only the null compression method, which disables TLS compression, **should** be used. If compression is used, the methods defined in [RFC3749] **shall** be used. If the client population served is known to support the compression method in [RFC3943], that method may be used instead. Other compression methods **shall not** be used. Compression method recommendations are based on the TLS standards. Limitations are recommended to ensure interoperability.

3.8 Operational Considerations

The sections above specify TLS-specific functionality. This functionality is necessary, but is not sufficient, to achieve security in an operational environment.

Federal agencies **shall** ensure that TLS servers include appropriate network security protections as specified in other NIST guidelines, such as [SP800-53].

The server **shall** operate on a secure operating system. Where the server relies on a FIPS 140 Level 1 cryptographic module, the software and private key **shall** be protected using the operating system identification, authentication and access control mechanisms. In some highly sensitive applications, server private keys may require protection using a FIPS 140 Level 2 or higher hardware cryptographic module.

The server and associated platform **shall** be kept up-to-date in terms of security patches. This is critical to various aspects of security, including the black list of certificates pushed by the product vendors. The black list of certificates is useful when an upstream CA certificate or client certificate is declared to be invalid or not operating with appropriate security measures, and the server does not perform revocation checking, does not have access to the latest revocation information, or the certificate has not been revoked.

3.9 Server Recommendations

This section contains summarized recommendations from Section 3.1 through Section 3.8 for the selection, configuration, and maintenance of a TLS server.

3.9.1 Recommendations for Server Selection

The following summary of recommendations is for individuals tasked with selecting a TLS server implementation for procurement. TLS server implementations **shall not** be procured unless they include the required functionality. Recommendations for server selection are:

1. Server implementations **shall** support TLS version 1.1.
2. Server implementations **should** support TLS version 1.2.

3. Server implementations may support TLS version 1.0.
4. Server implementations that incorrectly implement TLS version negotiation **shall not** be selected.
5. Server implementations **should** support multiple server certificates with their private keys to support algorithm and key size agility.
6. Server implementations **shall** use an Approved random bit generator specified in [SP800-90A].
7. Server implementations **shall** be able to terminate the connection with a “fatal handshake failure” alert when the client does not have a certificate or an acceptable certificate.
8. Server implementations **shall** be configurable to support Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP), or both.
9. Server implementations **shall** either support the path validation recommendations in Section 3.5.1 or be augmented to support them.
10. The server **shall** be able to provide the client certificate, and the certificate policies for which the client certification path is valid, to the applications in order to support access control decisions.

3.9.2 Recommendations for Server Installation and Configuration

The following summary of recommendations is for individuals tasked with the installation and initial configuration of a TLS server implementation. Recommendations for TLS server configuration are:

1. Version support
    a. The server **shall** be configured to support TLS version 1.1.
    b. The server **should** be configured to support TLS version 1.2.
    c. If the server supports government-only applications, it **shall not** be configured to support TLS version 1.0.
    d. If the server supports citizen or business facing applications, it may be configured to support TLS version 1.0.
    e. If TLS 1.0 is supported, TLS 1.1 and 1.2 **shall** be preferred over TLS 1.0.
    f. The server **shall not** be configured to support SSL 2.0 or SSL 3.0.
2. Certificates
    a. The server **shall** be configured with one or more public key certificates and the associated private keys.
    b. The server **shall** be configured with an RSA key encipherment certificate.
    c. The server **should** be configured with an ECDSA signature certificate or RSA signature certificate.
    d. If the server is not configured with an RSA signature certificate, an ECDSA signature certificate using a Suite B named curve for the signature and public key in the ECDSA certificate **should** be used.
    e. The server **shall** be configured with certificates issued by a CA, rather than self-signed certificates.
    f. Server certificates **shall** be issued by a CA that publishes revocation information in either CRLs or OCSP responses.
    g. The source for the revocation information **shall** be included in the certificate in the appropriate extension to promote interoperability.

h. All server certificates **shall** be X.509 version 3 certificates.
i. Both the public key contained in the certificate and the signature **shall** have at least 112 bits of security.
j. The certificate **shall** be signed with an algorithm consistent with the public key, as described in Section 3.2.1.
k. The server **should** be configured to support the server authentication extended key usage extension.
l. In the absence of agency-specific server certificate profile requirements, the certificate profile of Table 3-1 **should** be used for the server certificate.
m. The server **shall** perform revocation checking of the client certificate, when client authentication is used.
n. In the absence of agency-specific policies, Federal agencies **shall** use the Common Policy.

3. Cryptographic support
   a. The server **shall** be configured for data confidentiality and integrity services.
   b. The server **shall** be configured to only support cipher suites that are composed entirely of Approved algorithms.
   c. The server **shall** be configured to support the following cipher suites:
      TLS_RSA_WITH_3DES_EDE_CBC_SHA
      TLS_RSA_WITH_AES_128_CBC_SHA
   d. The server **should** be configured to support the following cipher suites:
      TLS_RSA_WITH_AES_256_CBC_SHA
      TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
      TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
      TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
      TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
   e. If the server is configured to support TLS version 1.2, then the server **shall** be configured to support the following cipher suite:
      TLS_RSA_WITH_AES_128_GCM_SHA256
   f. If the server is configured to support TLS version 1.2, then the server **should** be configured to support the following cipher suites:
      TLS_RSA_WITH_AES_256_GCM_SHA384
      TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
      TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
      TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
      TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
      TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
   g. The server may be configured to support other acceptable cipher suites, as described in Section 3.3.1.
   h. The server **shall** only support cipher suites for which it has a valid certificate containing a signature providing at least 112 bits of security.
   i. The server **shall not** be configured to support cipher suites other than those recommended in Section 3.3.1, unless otherwise stated by agency-specific policies.
   j. The server **should not** be configured to use cipher suites that do not appear in Section 3.3.1or Appendix C.

   k. For the RSA certificates, the key usage extension **shall** specify key encipherment for cipher suites that carry out the key exchange with RSA, and the key usage extension **shall** specify digital signature for cipher suites using ECDHE key exchange.
   l. The cryptographic module used by the server **shall** be a FIPS 140-validated cryptographic module.
   m. All cryptographic algorithms that are included in the cipher suites **shall** be within the scope of the validation, as well as the random number generator.
   n. The random number generator **shall** be tested and validated in accordance with [SP800-90A] under the NIST Cryptographic Algorithm Validation Program (CAVP) and successful results of this testing **shall** be indicated on the cryptographic module's FIPS 140 validation certificate.
   o. The validated random number generator **shall** be used to generate the server random value used in the TLS protocol.

4. Extensions
   a. The TLS server **shall** support the following TLS extensions, as described in Section 3.4.1:
      - Renegotiation Indication
      - Certificate Status Request
      - Server Name Indication
      - Trusted CA Indication
   b. The TLS server **shall** support the following TLS extensions when the conditions stated in Section 3.4.2 are met:
      - Supported Elliptic Curves
      - EC Point Format
      - Signature Algorithms
      - Multiple Certificate Status
   c. If the Supported Elliptic Curves extension is supported, the curves P-256 and P-384 **shall** be supported.
   d. The TLS server may support the following TLS extensions when the conditions stated in Section 3.4.2 are met:
      - Truncated HMAC
   e. The TLS server **should not** support the following TLS extensions:
      - Client Certificate URL
   f. If the Client Certificate URL extension is supported, the server **shall** be configured to mitigate attacks described in Section 3.4.3.

5. Client Authentication
   a. If the server supports client authentication, it **shall** support certificate-based client authentication.
   b. If possible, the server **shall** verify that client keys contain at least 112 bits of security.
   c. The server **shall** be configured to terminate the connection with a fatal "handshake failure" alert when a client certificate is requested, and the client does not have a suitable certificate.
   d. The server **shall** be configured such that each certificate in the certification path **shall** be validated using a Certificate Revocation List (CRL) or Online

Certificate Status Protocol (OCSP).

e. If the server supports OCSP, then OCSP checking **shall** be in compliance with [RFC6960] and **should** use only one of the options described in Section 3.5.1 of this document.

f. The server **shall** be configured to consider any revocation information in the CRL or OCSP responses whose grace period has elapsed relative to the selected time field ("thisUpdate" or "nextUpdate") as stale, where the grace period and applicable time field are determined by the agency.

g. The server **shall** be configured such that stale revocation information **shall not** be used to determine the validity of a certificate.

h. The server **shall** be configured to consider a certificate invalid if fresh revocation information cannot be obtained.

i. The server **shall** be able to determine the certificate policies that the client certificate is trusted for by using the certification path validation rules specified in Section 6 of [RFC5280].

j. The server **shall** be configured with only the trust anchors the server trusts, and of those, only the ones that are required to authenticate the clients, in the case where the server supports client authentication in TLS.

k. The default set of trust anchors for the server **shall** be examined to determine if any of them are required for client authentication.

l. The server **shall** check the client key length if client authentication is performed, and the server implementation provides a mechanism to do so.

m. The server **shall** be configured to maintain the trust anchors of the various PKI whose subscribers are the potential clients for the server, and include them in the hints list.

    i. Alternatively, the server **should** be configured to send an empty hints list so that the client can always provide a certificate it possesses.

n. The server hints list **shall** be distinct from the server trust anchor store.

o. The server **shall** continue to only populate its trust anchor store with the trust anchor of the server PKI domain and the domains it needs to trust directly for client authentication.

6. Session Resumption

    a. If there is a requirement to authenticate each client as it initiates a connection session, the server **shall** be configured to ignore requests to resume a session, and generate a new session ID, which forces the entire handshake procedure (including client authentication) to proceed.

7. Compression Methods

    a. The server **should** be configured to only support the null compression method, which disables TLS compression.

    b. If compression is used, the server **shall** be configured to only support the methods defined in [RFC3749].

        i. If the client population served is known to support the compression method in [RFC3943], that method may be used instead.

    c. The server **shall not** be configured to support other compression methods.

8. Operational Considerations

    a. The server **shall** operate on a secure operating system.

   b. Where the server relies on a FIPS 140 Level 1 cryptographic module, the software and private key **shall** be protected using the operating system identification, authentication and access control mechanisms.

3.9.3 Recommendations for Server System Administrators

A Server System Administrator is an individual who is responsible for maintaining the TLS server on a day-to-day basis.

1. Version support
   a. System administrators **shall** develop migration plans to support TLS 1.2 by January 1, 2015.
2. Certificates
   a. System administrators **shall** use Sections 3.2.1 and 3.2.2 to identify an appropriate source for certificates.
   b. System administrators **shall** install, maintain, and update certificates in accordance with the certificate recommendations of Section 3.9.2.
3. Cryptographic support
   a. System administrators **shall** maintain confidentiality and integrity service configurations in accordance with the recommendations of Section 3.9.2.
4. Client Authentication
   a. System administrators **shall** work with the agency to perform a risk assessment to determine acceptable grace periods for revocation information, as well as whether a grace period should be applied to the time found in the “thisUpdate” or “nextUpdate” field.
   b. System administrators of a TLS server that supports certificate-based client authentication **shall** perform an analysis of the client certificate issuers and use that information to determine the minimum set of trust anchors required for the server.
      i. The server **shall** be configured only to include only the minimum set of trust anchors needed.
5. Operational Considerations
   a. System administrators **shall** ensure that TLS servers include appropriate network security protections as specified in other NIST guidelines.
   b. The server **shall** operate on a secure operating system.
   c. Where the server relies on a FIPS 140 Level 1 cryptographic module, the system administrator **shall** ensure that the software and private key are protected using the operating system identification, authentication and access control mechanisms.
   d. The system administrator **shall** ensure that the server and associated platform are kept up-to-date in terms of security patches.

4 Minimum Requirements for TLS Clients

This section provides a minimum set of requirements that a TLS client must meet in order to adhere to these guidelines. Requirements are organized in the following sections: TLS protocol version support; client keys and certificates; cryptographic support; TLS extension support; server authentication; session resumption; compression methods; and operational considerations.

Specific requirements are stated as either implementation requirements or configuration requirements. Implementation requirements indicate that Federal agencies **shall not** procure TLS client implementations unless they include the required functionality. Configuration requirements indicate that system administrators are required to verify that particular features are enabled, or in some cases, configured appropriately if present.

4.1 Protocol Version Support

The client **shall** be configured to support TLS 1.1, and **should** be configured to support TLS 1.2. The client may be configured to support TLS 1.0 to facilitate communication with private sector servers, where necessary. If TLS 1.0 is supported, the use of TLS 1.1 and 1.2 **shall** be preferred over TLS 1.0. The client **shall not** support SSL version 3.0 or earlier. Agencies **shall** develop migration plans to support TLS 1.2 by January 1, 2015.

4.2 Client Keys and Certificates

4.2.1 Client Certificate Profile

When client authentication is needed, the client **shall** be configured with a certificate that adheres to the recommendations presented in this section. A client certificate may be configured on the system, or located on an external device (e.g., a PIV card). For this specification, the TLS client certificate **shall** be an X.509 version 3 certificate; both the public key contained in the certificate and the signature **shall** have at least 112 bits of security. The certificate **shall** be signed with an algorithm consistent with the public key:

- Certificates containing RSA (signature), ECDSA, or DSA public keys **shall** be signed with those same signature algorithms, respectively;
- Certificates containing Diffie-Hellman certificates **shall** be signed with DSA; and
- Certificates containing ECDH public keys **shall** be signed with ECDSA.

The extended key usage extension limits the operations that keys in a certificate may be used for. There is a key usage extension specifically for client authentication. The use of the extended key usage extension will ensure that the servers accept the certificate as a client certificate. The extended usage extension can also indicate that the certificate is not to be used for other purposes, such as code signing. The client certificates **should** include an extended key usage extension that specifies the client authentication key purpose object identifier[18].

[18] Absence of extended key usage extension in some implementation is known to be interpreted as having special permission such as code signing, even though not specifically indicated in the certificate.

The client certificate profile is listed in Table 4-1. In the absence of an agency-specific client certificate profile, this profile **should** be used for client certificates.

Note that for ECDH, the algorithm OID and the signature OID are identical to those of ECDSA. For interoperability reasons, algorithm OID is not changed and the key usage extension determines if the public key is used for key agreement or signature verification.

Table 4-1: TLS Client Certificate Profile

Field	Critical	Value	Description
Version	N/A	2	Version 3
Serial Number	N/A	Unique positive integer	Must be unique
Issuer Signature Algorithm	N/A	*Values by certificate type:*	
		sha256WithRSAEncryption {1 2 840 113549 1 1 11}, or stronger	RSA key encipherment certificate, RSA signature certificate
		ecdsa-with-SHA256 {1 2 840 10045 4 3 2}, or stronger	ECDSA signature certificate, ECDH certificate
		id-dsa-with-sha256 {2 16 840 1 101 3 4 3 2}, or stronger	DSA signature certificate, DH certificate
Issuer Distinguished Name	N/A	Unique X.500 Issuing CA DN	Single value shall be encoded in each RDN. All attributes that are of directoryString type shall be encoded as a printable string.
Validity Period	N/A	3 years or less	Dates through 2049 expressed in UTCTime
Subject Distinguished Name	N/A	Unique X.500 subject DN per agency requirements	Single value shall be encoded in each RDN. All attributes that are of directoryString type shall be encoded as a printable string.
Subject Public Key Information	N/A	*Values by certificate type:*	
		rsaEncryption {1 2 840 113549 1 1 1}	RSA key encipherment certificate, RSA signature certificate 2048 bit RSA key modulus Parameters: NULL
		ecPublicKey {1 2 840 10045 2 1}	ECDSA signature certificate, or ECDH certificate Parameters: namedCurve OID for names curve specified in FIPS 186-4. The curve **shall** be P-256 or P-384 SubjectPublic Key: Uncompressed EC Point.
		id-dsa {1 2 840 10040 4 1}	DSA signature certificate Parameters: p, q, g
		dhpublicnumber {1 2 840 10046 2 1}	DH certificate Parameters: p, g, q
Issuer's Signature	N/A	*Values by certificate type:*	
		sha256WithRSAEncryption {1 2 840 113549 1 1 11}, or stronger	RSA key encipherment certificate, RSA signature certificate
		ecdsa-with-SHA256 {1 2 840 10045 4 3 2}, or stronger	ECDSA signature certificate, ECDH certificate
		id-dsa-with-sha256 { 2 16 840 1 101 3 4 3 2}, or stronger	DSA signature certificate, DH certificate
Extensions			
Authority Key Identifier	No	Octet String	Same as subject key identifier in Issuing CA certificate Prohibited: Issuer DN, Serial Number tuple

Field	Critical	Value	Description
Subject Key Identifier	No	Octet String	Same as in PKCS-10 request or calculated by the Issuing CA
Key Usage	Yes	digitalSignature	RSA certificate, DSA certificate, ECDSA certificate
		keyAgreement	ECDH certificate, DH certificate
Extended Key Usage	No	id-kp-clientAuth {1 3 6 1 5 5 7 3 2}	Required
		anyExtendedKeyUsage {2 5 29 37 0}	Prohibited[19]
			Prohibited: all others unless consistent with key usage extension
Certificate Policies	No	Per agency X.509 certificate policy	
Subject Alternative Name	No	RFC 822 e-mail address, Universal Principal Name (UPN), DNS Name, and/or others	Optional
Authority Information Access	No	id-ad-caIssuers	Required. Access method entry contains HTTP URL for certificates issued to Issuing CA
		id-ad-ocsp	Optional. Access method entry contains HTTP URL for the Issuing CA OCSP Responder
CRL Distribution Points	No	See comments	Optional: HTTP value in distributionPoint field pointing to a full and complete CRL. Prohibited: reasons and cRLIssuer fields, and nameRelativetoCRLIssuer CHOICE

Multiple client certificates may be present that meet the requirements of the TLS server. The TLS client (e.g., a browser) may ask users to select from a list of certificates. The use of the Extended Key Usage (EKU) extension may eliminate this request.

Client certificates are also filtered by TLS clients on the basis of an ability to build a path to one of the trust anchors in the hints list sent by the server, as described in Section 3.5.4.

4.2.2 Obtaining Revocation Status Information for the Server Certificate

The client **shall** perform revocation checking of the server certificate. Revocation information can be obtained by the client from one of the following locations:

1. Certificate Revocation List (CRL) or OCSP [RFC6960] response in the client's local certificate store;
2. OCSP response from a locally configured OCSP responder;
3. OCSP response from the OCSP responder location identified in the OCSP field in the Authority Information Access extension in the server certificate; or
4. CRL from the CRL Distribution Point extension in the server certificate.

When the local certificate store does not have the current or a cogent CRL or OCSP response, and the OCSP Responder and the CRL Distribution Point are unavailable or

[19] The presence of anyExtendedKeyUsage {2 5 29 37 0} in some implementation is known to be interpreted as having special permission such as code signing, even though not specifically indicated in the certificate.

inaccessible at the time of TLS session establishment, the client will either terminate the connection or accept a potentially revoked or compromised certificate. The decision to accept or reject a revoked certificate **should** be made according to agency policy. In order to mitigate the risk of revocation information unavailability, the OCSP stapling extension [RFC6961] may be used. This extension is further described in Section 4.4.2.5.

Other emerging concepts that can be useful in lieu of revocation checking are further discussed in Appendix D.

4.2.3 Client Public Key Certificate Assurance

The client public key certificate may be trusted by the servers on the basis of the policies, procedures and security controls used to issue the client public key certificate as described in Section 3.5.1. For example, as the implementation of Personal Identify Verification (PIV) [FIPS201-1] becomes more established in Federal Agencies, these guidelines recommend that the PIV Authentication certificate be the norm for authentication of Federal employees and long-term contractors. For users who do not have PIV Cards, such as external users, the set of certificate policies to accept should be determined as specified in Appendix B of [SP800-63], based on the level of assurance required by the application. PIV Authentication certificate policy is defined in [COMMON] and PIV-I Authentication certificate policy is defined in [FBCACP]. Depending on the requirements of the server-side application, other certificate policies defined in [COMMON] may also be acceptable. Guidance regarding the acceptable certificate policies is outside the scope of these guidelines.

4.3 Cryptographic Support

4.3.1 Cipher Suites

The acceptable cipher suites for a TLS client are the same as those for a TLS server. General-purpose cipher suites are listed in Section 3.3.1, and cipher suites appropriate for pre-shared key environments are listed in Appendix C.

The client **should not** be configured to use cipher suites other than those listed in Section 3.3.1 or Appendix C.

4.3.2 Validated Cryptography

Clients **shall** use validated cryptography, as described for the server in Section 3.3.2.

4.4 TLS Extension Support

4.4.1 Mandatory TLS Extensions

The client **shall** support the following extensions:

1. Renegotiation Indication
2. Server Name Indication
3. Trusted CA Indication

4.4.1.1 Renegotiation Indication

The Renegotiation Indication extension is required by these guidelines as described in Section 3.4.1.1. Clients **shall** perform initial and subsequent renegotiations in accordance with [RFC5746].

4.4.1.2 Server Name Indication

The server name indication extension is described in Section 3.4.1.3. The client **shall** be capable of including this extension in a ClientHello message, as described in [RFC6066].

4.4.1.3 Trusted CA Indication

The client **shall** be capable of including the trusted CA indication (trusted_ca_keys) extension in a ClientHello message as described in [RFC6066].

4.4.2 Conditional TLS Extensions

A TLS client supports the following TLS extensions under the circumstances described:

1. The Supported Elliptic Curves TLS extension **shall** be supported if the client supports EC cipher suite(s).
2. The EC Point Format TLS extension **shall** be supported if the client supports EC cipher suite(s).
3. The Signature Algorithms TLS extension **shall** be supported when the client is operating in TLS 1.2.
4. The Certificate Status Request extension **shall** be supported when the client is not able to obtain revocation information.
5. The Multiple Certificate Status extension **shall** be supported if the extension is supported by the client implementation.
6. The Truncated HMAC extension may be supported by clients that run on constrained devices when variable-length padding is not supported.

4.4.2.1 Supported Elliptic Curves

The clients that support EC cipher suites **shall** be capable of listing the elliptic curves supported in the ClientHello message, in accordance with Section 5.1 of [RFC4492].

4.4.2.2 EC Point Format

The clients that support EC cipher suites **shall** be capable of specifying the supported EC point format in the ClientHello message, in accordance with Section 5.1 of [RFC4492].

Clients that support EC cipher suites **shall** support the processing of at least one[20] of the EC point formats received in the ServerHello message, as described in Section 5.2 of [RFC4492].

4.4.2.3 Signature Algorithms

[20] The uncompressed point format must be supported, as described in Sections 5.1.2 and 5.2 of [RFC4492].

The clients that support TLS 1.2 **shall** be able to assert acceptable hashing and signature algorithm pairs in this extension in a ClientHello message. The extension, its syntax, and processing rules are described in Sections 7.4.1.4.1, 7.4.4, 7.4.6 and 7.4.8 of [RFC5246].

4.4.2.4 Certificate Status Request

When the client wishes to receive the revocation status of the TLS server certificate from the TLS server, the client **shall** include the “status_request” extension in the ClientHello message.

4.4.2.5 Multiple Certificate Status

The multiple certificate status extension is described in Section 3.4.2.4. This extension improves on the Certificate Status Request extension described in Section 3.4.1.2 by allowing the client to request the status of all certificates provided by the Server in the TLS handshake. This extension is documented in [RFC6961]. Client implementations that have this capability **shall** be configured to support this extension.

4.4.2.6 Truncated HMAC

The Truncated HMAC extension is described in Section 3.4.2.5. Clients running on constrained devices may support this extension. The Truncated HMAC extension **shall not** be used in conjunction with variable-length padding, due to attacks described in [Paterson11].

4.4.3 Discouraged TLS Extensions

The following extension **should not** be used:

1. Client Certificate URL

The reasons for discouraging the use of this extension can be found in Section 3.4.3.

4.5 Server Authentication

The client **shall** be able to build the certification path for the server certificate presented in the TLS handshake with at least one of the trust anchors in the client trust store, if an appropriate trust anchor is present in the store. The client may use all or a subset of the following resources to build the certification path: local certificate store, LDAP, resources declared in CA Repository field of the Subject Information Access extension in various CA certificates, and resources declared in the CA Issuers field of the Authority Information Access extension in various certificates.

4.5.1 Path Validation

The client **shall** validate the server certificate in accordance with the certification path validation rules specified in Section 6 of [RFC5280]. In addition, the revocation status of each certificate in the certification path **shall** be checked using the Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP). OCSP checking **shall** be in compliance with [RFC6960] and **should** use only one of the following options:

- The OCSP Responder is trusted by the client, i.e., the OCSP Responder public key is the same as that of one of the public keys in the client's trust anchor store; or
- The OCSP Response is signed using the same key as that of the certificate whose status is being checked; or
- The OCSP Response is signed by a designated/delegated OCSP Responder as described in [RFC6960], and the OCSP Responder certificate is signed using the same key as that of the certificate whose status is being checked.

Revocation information **shall** be obtained as described in Section 4.2.2.

Federal agencies **shall** perform a risk assessment to determine acceptable grace periods for revocation information, as well as whether a grace period should be applied to the time found in the "thisUpdate" or "nextUpdate" field. If the determined grace period has elapsed relative to the selected time field, then the revocation information **shall** be considered stale, and the stale revocation information **shall not** be used to determine the validity of the certificate. If fresh revocation information cannot be obtained through another source, the certificate **shall** be considered invalid.

Not all commercial products support the public key certification path validation and certificate policy processing rules listed and cited above. Specifically, revocation checking in some instances may not be available, or the client could accept a server public key certificate if the latest revocation information is inaccessible. Similarly, some clients are not able to provide inputs related to acceptable certificate policy or initial values for requiring policies, and inhibiting policy mapping. In the absence of clients that are fully certificate policy aware, Federal agencies may use other mechanisms to decide if a server certificate has been issued with due diligence.

Not all clients support checking name constraints. The Federal agencies **shall** only procure clients that perform name constraint checking in order to obtain assurance that unauthorized certificates are properly rejected. As an alternative, the Federal agency may procure clients that use one or more of the features discussed in Appendix D.

The client **shall** terminate the TLS connection if path validation fails.

Federal agencies **shall** only use clients that check that the DNS name or IP addresses presented in the client TLS request matches a DNS name or IP address contained in the server certificate's subject alternative name extension. If the name presented in the client TLS request is absent from the server certificate's subject alternative name extension, then the client **shall** check the server certificate's subject distinguished name field to determine if the subject distinguished name contains the requested name. The client **shall** terminate the TLS connection if the name check fails.

4.5.2 Trust Anchor Store

Having an excessive number of trust anchors installed in the TLS client can increase the chances for the client to be spoofed. As the number of trust anchors increase, the number of CAs that the client trusts increases, and the chances that one of these CAs or their registration system or process will be compromised to issue TLS server certificates also increases. In the minimal case, a Federal Agency relying party client can have a single trust anchor: an agency legacy trust anchor or the Common Policy trust anchor.

Federal Agencies **shall** perform a trade-off between the risk associated with and need to access commercial web sites to determine the trust anchor store in the various client machines. Federal agencies **shall** administer this trust anchor store through centralized management applications. Federal agency systems and clients **shall** be configured such that an update to the trust anchor store is a privileged system administrative function requiring appropriate agency security approval.

To mitigate the client certificate selection and path-building problem at the client end described in Section 3.5.4, clients **shall not** overpopulate their trust stores with various CA certificates that can be verified via cross-certification. Direct trust of these certificates can expose the clients unduly to a variety of situations, including but not limited to, revocation or compromise of these trust anchors. Direct trust also increases the operational and security burden on the clients to promulgate addition and deletion of trust anchors. Instead, the client **shall** rely on the server overpopulating or not providing the hints listed as discussed in Section 3.5.4.

4.5.3 Checking the Server Key Size

The only direct mechanism for a client to check if the key size presented in a server public certificate is acceptable is for the client to examine the server public key in the certificate. An indirect mechanism is to check that the certificate policies extension in the server public key certificate indicates the minimum cryptographic strength of the signature and hashing algorithms used and for the client to perform certificate policy processing and checking. A more scalable and more robust alternative that is standards-based is described in Appendix D. The client **shall** check the server public key length if the client implementation provides a mechanism to do so.

The length of each write key is determined by the negotiated cipher suite. Restrictions on the length of the shared session keys can be enforced by configuring the client to only support cipher suites that meet the key length requirements.

4.5.4 User Interface

When the TLS client is a browser, the browser interface can be used to determine if a TLS session is in effect. The indication that a TLS session is in effect varies by browser. Examples of indicators include a padlock in the URL bar, or a different color for the URL bar. Some clients, such as browsers, may allow further investigation of the server certificate and negotiated session parameters by clicking on the lock (or other indicator). Users **should** examine the interface for the presence of the indicator to ensure that the TLS session is in force and **should** also visually examine the web site URL to ensure that the user intended to visit the indicated web site. Users **should** be aware that URLs can appear to be legitimate, but still not be valid. For example, the numeric “1” and the letter “l” appear quite similar or the same to the human eye. If the user navigates to a URL that appears to be correct, the browser software could defeat these threats by matching the requested URL with the DNS name in the server certificate.

Client authentication keys may be located outside of the client (e.g., PIV cards). Users **shall** follow the policies and procedures for protecting client authentication keys outside of the client.

4.6 Session Resumption

The client **shall** follow the same session resumption recommendations as the server, which are described in Section 3.6.

4.7 Compression Methods

The client **shall** follow the same compression recommendations as the server, which are described in Section 3.7.

4.8 Operational Considerations

The client and associated platform **shall** be kept up-to-date in terms of security patches. This is critical to various aspects of security, including the black list of certificates pushed by the product vendors. The black list of certificates is useful when an upstream CA certificate or server certificate is declared to be invalid or not operating with appropriate security measures, and the client does not perform revocation checking, does not have access to the latest revocation information, or the certificate has not been revoked.

Once the TLS-protected data is received at the client, and decrypted and authenticated by the TLS layer of the client system, the unencrypted data is available to the applications on the client platform.

These guidelines also do not mitigate the threats against the misuse or exposure of the client credential that resides on the client machine. These credentials could contain the private key used for client authentication or other credentials (e.g., one-time password (OTP) or user ID and password) for authenticating to server side application.

For these reasons, the use of TLS does not obviate the need for the client to use appropriate security measures, as described in applicable Federal Information Processing Standards and NIST Special Publications, to protect computer systems and applications. Users **shall** operate client systems in accordance with agency and administrator instructions.

4.9 Client Recommendations

This section contains summarized recommendations from Section 4.1 through Section 4.8 for the selection, configuration, maintenance, and use of a TLS client.

4.9.1 Recommendations for Client Selection

The following summary of recommendations is for individuals tasked with selecting a TLS client implementation for procurement. TLS clients **shall not** be procured unless they include the required functionality. Recommendations for client selection are:

1. Client implementations **shall** support TLS version 1.1.
2. Client implementations **should** support TLS version 1.2.

3. Client implementations may support TLS version 1.0.
4. Client implementations **shall** be configurable to prefer TLS 1.1 and TLS 1.2 over TLS 1.0.
5. Client implementations **shall** support the client authentication extended key usage extension.
6. Client implementations **shall** support name constraint checking in order to ensure that unauthorized certificates are properly rejected.
7. Client implementations **shall** check that the DNS name or IP addresses presented in the client TLS request matches a name or IP address contained in the server certificate's subject distinguished name field or subject alternative name extension.
8. Client implementations **shall** terminate the TLS connection if the path validation fails.

4.9.2 Recommendations for Client Installation and Configuration

The following summary of recommendations is for individuals tasked with the installation and initial configuration of a TLS client implementation. Recommendations for TLS client configuration are:

1. Version Support
    a. The client **shall** be configured to support TLS version 1.1.
    b. The client **should** be configured to support TLS version 1.2.
    c. The client may be configured to support TLS version 1.0.
    d. If TLS version 1.0 is supported, the client **shall** be configured to prefer TLS 1.1 and TLS 1.2 over TLS 1.0.
    e. The client **shall not** be configured to support SSL version 3.0 or earlier.
2. Certificates
    a. All client certificates **shall** be X.509 version 3 certificates.
    b. Both the public key contained in the certificate and the signature **shall** have at least 112 bits of security.
    c. The certificate **shall** be signed with an algorithm consistent with the public key, as described in Section 4.2.1.
    d. The client certificate **should** include an extended key usage extension that specifies the client authentication key purpose object identifier.
    e. In the absence of an agency-specific client certificate profile, the profile in Table 4-1 **should** be used for client certificates.
    f. The client **shall** perform revocation checking of the server certificate, as described in Section 4.2.2.
    g. The client **should** be configured to make the decision to accept or reject a revoked certificate according to agency policy.
    h. The OCSP stapling extension may be used.
3. Cryptographic support
    a. The client **shall** be configured to support the following cipher suites:
        TLS_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_RSA_WITH_AES_128_CBC_SHA
    b. The client **should** be configured to support the following cipher suites:
        TLS_RSA_WITH_AES_256_CBC_SHA
        TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA

        TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
        TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
        TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

   c. If the client is configured to support TLS 1.2, then the client **shall** be configured to support the following cipher suites:
      - TLS_RSA_WITH_AES_128_GCM_SHA256
   c. If the client is configured to support TLS 1.2, then the client **should** be configured to support the following cipher suites:
      - TLS_RSA_WITH_AES_256_GCM_SHA384
      - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
      - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
      - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
      - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
      - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
   d. The client **should not** be configured to support cipher suites other than those listed above and in Section 3.3.1 or Appendix C.
   e. Clients **shall** use validated cryptography, as described for the server in Section 3.3.2.

4. Extensions
   a. The TLS client **shall** support the following TLS extensions, as described in Section 4.4.1:
      - Renegotiation Indication
      - Server Name Indication
      - Trusted CA Indication
   b. The TLS client **shall** support the following TLS extensions, as described in Section 4.4.2, when the conditions stated in Section 4.4.2 are met:
      - Supported Elliptic Curves
      - EC Point Format
      - Signature Algorithms
      - Certificate Status Request
      - Multiple Certificate Status
   c. The TLS client may support the following TLS extension when the condition stated in Section 4.4.2 is met:
      - Truncated HMAC
   d. The TLS client **should not** support the following TLS extension:
      - Client Certificate URL
5. Server Authentication
   a. The client **shall** be able to build the certification path for the server certificate presented in the TLS handshake with at least one of the trust anchors in the client trust store, if an appropriate trust anchor is present in the store.
   b. The client **shall** validate the server certificate in accordance with the certification path validation rules specified in Section 6 of [RFC5280].
   c. The client **shall** be configured such that the revocation status of each certificate in the certification path **shall** be checked using the Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP).

   d. If the client supports OCSP, then OCSP checking **shall** be in compliance with [RFC6960] and **should** use only one of the options described in Section 4.5.1 of this document.
   e. The client **shall** be configured to consider any revocation information in the CRL or OCSP responses whose grace period has elapsed relative to the selected time field (“thisUpdate” or “nextUpdate”) as stale, where the grace period and applicable time field are determined by the agency.
   f. The client **shall** be configured such that stale revocation information **shall not** be used to determine the validity of a certificate.
   g. The client **shall** be configured to consider a certificate invalid if fresh revocation information cannot be obtained.
   h. The client **shall** terminate the TLS connection if path validation fails.
   i. The client **shall** check that the DNS name or IP addresses presented in the client TLS request matches a name or IP address contained in the server certificate’s subject alternative name extension.
   j. If the name presented in the client TLS request is absent from the server certificate’s subject alternative name extension, then the client **shall** check the server certificate’s subject distinguished name field to determine if the subject distinguished name contains the requested name.
   k. The client **shall** terminate the TLS connection if the name check fails.
   l. Clients **shall not** overpopulate their trust stores with various CA certificates that can be verified via cross-certification.
   m. The client **shall** rely on server trust store overpopulating or not providing the hints list as discussed in Section 3.5.4.
   n. The client **shall** check the server public key length if the client implementation provides a mechanism to do so.
6. Session Resumption
   a. If there is a requirement to authenticate the server for each connection session, the client **shall** generate a new session ID, which forces the entire handshake procedure (including server authentication) to proceed.
7. Compression Methods
   a. The client **should** support the null compression method, which disables TLS compression.
   b. If compression is used, the client **shall** support the methods defined in [RFC3749].
      i. If the server population served is known to support the compression method in [RFC3943], that method may be used instead.
   c. The client **shall not** support other compression methods.

4.9.3 Recommendations for Client System Administrators

A Client System Administrator is an individual who is responsible for maintaining the TLS client on a day-to-day basis.

1. Version support
   a. System administrators **shall** develop migration plans to support TLS 1.2 by January 1, 2015.
2. Certificates

    a. System administrators shall install, maintain, and update certificates in accordance with the certificate recommendations of Section 4.9.2.

3. Server Authentication
    a. System administrators **shall** perform a risk assessment to determine acceptable grace periods for revocation information, as well as whether a grace period should be applied to the time found in the "thisUpdate" or "nextUpdate" field.
    b. System administrators **shall** perform a trade-off between risk associated with and need to access commercial web sites to determine the trust anchor store in the various client machines.
    c. System administrators **shall** administer the trust anchor store through centralized management applications.
    d. System administrators **shall** configure clients such that an update to the trust anchor store is a privileged system administrative function requiring appropriate agency security approval.
    e. Administrators **shall** ensure that client trust stores are not overpopulated with various CA certificates that are otherwise to be trusted via cross-certification.
4. Operational Considerations
    a. The client and associated platform **shall** be kept up-to-date in terms of security patches.

4.9.4 Recommendations for End Users

An end user is an individual using a client to establish a TLS connection. Recommendations for end users are:

1. If the client is a browser, users **should** examine the interface to ensure that the TLS session is in force and also to visually examine the web site URL to ensure that the user intended to visit the web site.
2. Users **should** be aware that URLs can appear to be legitimate, but still not be valid.
3. Users **shall** operate client systems in accordance with agency and administrator instructions.
4. Users **shall** follow appropriate policies and procedures for protecting client authentication keys outside of the client (e.g., PIV cards).

Appendix A Acronyms

Selected acronyms and abbreviations used in these guidelines are defined below.

3DES	Triple DES (TDEA)
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
CA	Certification Authority
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CRL	Certificate Revocation List
DES	Data Encryption Standard
DH	Diffie-Hellman key exchange
DHE	Ephemeral Diffie-Hellman key exchange
DNS	Domain Name System
DNSSEC	DNS Security Extensions
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard (implies DSA)
EC	Elliptic Curve
ECDHE	Ephemeral Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
GCM	Galois Counter Mode
IETF	Internet Engineering Task Force
MAC	Message Authentication Code
OCSP	Online Certificate Status Protocol
OID	Object Identifier
PIV	Personal Identity Verification
PKI	Public Key Infrastructure
PRF	Pseudo-random Function
PSK	Pre-shared Key
RFC	Request for Comments
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Uniform Resource Locator

Appendix B Interpreting Cipher Suite Names

The cipher suite name consists of a set of mnemonics separated by underscores (i.e., "_"). The first mnemonic is the protocol name, i.e., TLS.

One or two mnemonics follow the protocol name. If there is only one mnemonic, it must be RSA or PSK, based on the recommendations in these guidelines. The single mnemonic RSA signifies that the public key in the server certificate is an RSA key transport public key that should be used by the client for sending the premaster secret to the server. The single mnemonic PSK indicates that the premaster secret is established using only symmetric algorithms with pre-shared keys, as described in [RFC4279]. Pre-shared key cipher suites that are approved for use are listed in Appendix C. If there are two mnemonics, the first mnemonic should be DH, ECDH, DHE or ECDHE. When the first mnemonic is DH or ECDH, it indicates that the server public key in its certificate is for either DH or ECDH key exchange, and the second mnemonic indicates the signature algorithm that was used by the issuing CA to sign the server certificate. When the first mnemonic is DHE or ECDHE, it indicates that ephemeral DH or ECDH will be used for key exchange, with the second mnemonic indicating the server signature public key type[21] that will be used to authenticate the server's ephemeral public key.

Next is the word WITH and the mnemonic for the symmetric encryption algorithm and associated mode of operations.

The last mnemonic is generally the hashing algorithm to be used for HMAC, if applicable[22]. In cases where HMAC is not applicable (e.g., AES-GCM), and the cipher suite is defined after the release of the TLS 1.2 RFC, this mnemonic represents the hashing algorithm for the PRF.

The following examples illustrate how to interpret the cipher suite names:

- TLS_RSA_WITH_3DES_EDE_CBC_SHA: The server is using an RSA public key that the client would use for key exchange. The CA signature algorithm is not specified. Once the handshake is completed, the messages are encrypted using triple DES in CBC mode. In TLS versions 1.0 and 1.1, a combination of SHA-1 and MD5 is used in the PRF, and SHA-1 is used for HMAC computations on the messages. In TLS 1.2, SHA-256 is used for the PRF, and SHA-1 is used for HMAC computations on the messages.
- TLS_DH_DSS_WITH_AES_256_CBC_SHA256: The server is using a DH certificate. If the connection is using TLS 1.2, and the signature algorithms extension is provided by the client, then the certificate is signed using the algorithm specified by the extension. Otherwise, the certificate is signed using DSA. Once the handshake is completed, the messages are encrypted using AES 256 in CBC mode. SHA-256 is used for both the PRF and HMAC computations.

[21] In this case, the signature algorithm used by the CA to sign the certificate is not articulated in the cipher suite.

[22] HMAC is not applicable when the symmetric encryption mode of operation is authenticated encryption, i.e., CCM or GCM. Separately, note that the CCM mode cipher suites do not specify the last mnemonic and require that SHA-256 be used for the PRF.

Cipher suites that specify secure hash algorithms other than SHA-1 are not supported prior to TLS 1.2.

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384: Ephemeral ECDH is used for key exchange. The server's ephemeral public key is authenticated using the server's ECDSA public key. The CA signature algorithm used to certify the server's ECDSA public key is not specified. Once the handshake is completed, the messages are encrypted and authenticated using AES-256 in GCM mode, and SHA-384 is used for the PRF. Since an authenticated encryption mode is used, messages neither have nor require an HMAC message authentication code.

Appendix C Pre-shared Keys

Pre-shared keys (PSK) are symmetric keys that are already in place prior to the initiation of a TLS session (e.g., as the result of a manual distribution). The use of PSKs in the TLS protocol is described in [RFC4279], [RFC5487], and [RFC5489]. In general, pre-shared keys **should not** be used. However, the use of pre-shared keys may be appropriate for some closed environments that have adequate key management support. For example, they might be appropriate for constrained environments with limited processing, memory, or power. If PSKs are appropriate and supported, then the following additional guidelines **shall** be followed.

Recommended pre-shared key (PSK) cipher suites are listed in Table C-1; pre-shared keys **shall** be distributed in a secure manner, such as a secure manual distribution or using a key establishment certificate. These cipher suites employ a pre-shared key for entity authentication (for both the server and the client) and may also use RSA or ephemeral Diffie-Hellman (DHE) algorithms for key establishment. For example, when DHE is used, the result of the Diffie-Hellman computation is combined with the pre-shared key and other input to determine the premaster secret.

The pre-shared key **shall** have a minimum security strength of 112-bits. Because these cipher suites require pre-shared keys, these suites are not generally applicable to classic secure web site applications and are not expected to be widely supported in TLS clients or TLS servers. NIST suggests that these suites be considered in particular for infrastructure applications, particularly if frequent authentication of the network entities is required. These cipher suites may be used with TLS versions 1.1 or 1.2. Note that cipher suites using GCM, SHA-256, or SHA-384 are only available in TLS 1.2.

Pre-shared key cipher suites may only be used in networks where both the client and server are government systems. Cipher suites using pre-shared keys **shall not** be supported when TLS 1.0 is supported, and **shall not** be supported where the client or server communicates with non-government systems.

Table C-1: Pre-shared Key Cipher Suites

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_PSK_WITH_3DES_EDE_CBC_SHA	PSK	3DES_EDE_CBC	SHA-1	Per RFC
TLS_PSK_WITH_AES_128_CBC_SHA	PSK	AES_128_CBC	SHA-1	Per RFC
TLS_PSK_WITH_AES_256_CBC_SHA	PSK	AES_256_CBC	SHA-1	Per RFC
TLS_PSK_WITH_AES_128_GCM_SHA256	PSK	AES_128_GCM	N/A	SHA-256
TLS_PSK_WITH_AES_256_GCM_SHA384	PSK	AES_256_GCM	N/A	SHA-384
TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA	DHE_PSK	3DES_EDE_CBC	SHA-1	Per RFC
TLS_DHE_PSK_WITH_AES_128_CBC_SHA	DHE_PSK	AES_128_CBC	SHA-1	Per RFC
TLS_DHE_PSK_WITH_AES_256_CBC_SHA	DHE_PSK	AES_256_CBC	SHA-1	Per RFC
TLS_DHE_PSK_WITH_AES_128_GCM_SHA256	DHE_PSK	AES_128_GCM	N/A	SHA-256
TLS_DHE_PSK_WITH_AES_256_GCM_SHA384	DHE_PSK	AES_256_GCM	N/A	SHA-384
TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA	RSA_PSK	3DES_EDE_CBC	SHA-1	Per RFC
TLS_RSA_PSK_WITH_AES_128_CBC_SHA	RSA_PSK	AES_128_CBC	SHA-1	Per RFC
TLS_RSA_PSK_WITH_AES_256_CBC_SHA	RSA_PSK	AES_256_CBC	SHA-1	Per RFC
TLS_RSA_PSK_WITH_AES_128_GCM_SHA256	RSA_PSK	AES_128_GCM	N/A	SHA-256
TLS_RSA_PSK_WITH_AES_256_GCM_SHA384	RSA_PSK	AES_256_GCM	N/A	SHA-384
TLS_ECDHE_PSK_WITH_3DES_EDE_CBC_SHA	ECDHE_PSK	3DES_EDE_CBC	SHA-1	Per RFC

Cipher Suite Name	Key Exchange	Encryption	Hash function for HMAC	Hash Function for PRF
TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA	ECDHE_PSK	AES_128_CBC	SHA-1	Per RFC
TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA	ECDHE_PSK	AES_256_CBC	SHA-1	Per RFC
TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256	ECDHE_PSK	AES_128_CBC	SHA-256	SHA-256
TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384	ECDHE_PSK	AES_256_CBC	SHA-384	SHA-384

Appendix D Future Capabilities

This section identifies emerging concepts and capabilities that are applicable to TLS. As these concepts mature, and commercial products are available to support them, these guidelines will be revised to provide specific recommendations.

D.1 Additional/Alternate Web Server Certificate Validation Mechanisms

In order to deal with the threat associated with the compromise of a CA, registration system, or process, new ideas about how to gain assurance of the legitimacy of the server certificate presented in a TLS session have been developed.

In addition, new standards are emerging in the use of public key technology to secure the DNS. These DNSSEC standards can be used to replace or augment the traditional PKI approach to establishing trust in the server certificate.

The following sections describe these concepts. In some cases, these concepts are not fully standardized, and in most cases, they are not widely available in commercial products. As these concepts mature and become widely available, these guidelines will be revised to describe them further and to recommend how they can used to augment or replace traditional mechanisms to establish trust in the server certificate and associated revocation checking.

D.1.1 Sovereign Keys

The sovereign key approach has been developed by the Electronic Frontier Foundation. Under this approach, the server public key certificates and, optionally, intermediate CA certificates are claimed by the server domain holder, and these claims are countersigned by one or more trusted third parties. When client systems are shipped with these trusted third-party public keys, clients can query the records and obtain the claims to verify that the server certificate being presented in the TLS handshake is legitimate (i.e., has been signed by a trusted third party). The concept is further described in [SOVER]. While the concept is still in the development stage, its use can obviate the need for public key certification path development, validation and revocation checking, and replace the server authentication requirements listed in Section 4.5.

D.1.2 Certificate Transparency

Google's Certificate Transparency project [RFC6962] strives to reduce the impact of certificate-based threats by making the issuance of CA-signed certificates more transparent. This is done through the use of public logs of certificates, public log monitoring, and public certificate auditing. Certificate logs are cryptographically assured records of certificates that are open to public scrutiny. Certificates may be appended to logs, but they cannot be removed, modified, or inserted into the middle of a log. Monitors watch certificate logs for suspicious certificates, such as those that were not authorized by the domain they claim to represent. Auditors have the ability to check the membership of a particular certificate in a log, as well as verify the integrity and consistency of logs.

D.1.3 Perspectives and Convergence

Perspectives is a project undertaken at Carnegie Mellon University [PERSP]. Perspectives takes a different approach to establish trust in a TLS server public key certificate than using trust in certification authorities and the public key certificate trust model in X.509 and [RFC5280]. Perspectives has a decentralized model that uses "network notary servers." A network notary server is connected to the Internet and regularly monitors websites to build a history of the TLS certificate used by each site. Rather than validating a TLS server certificate as described in [RFC5280] and in Section 4.5, with Perspectives, the TLS client validates a certificate by checking for consistency with the certificates observed by the network notaries over time. A client has the network notaries' public keys embedded in it and decides which and how many notary servers to trust. Clients can also decide how many notaries must provide a positive response before trusting a TLS server public key certificate and can augment the decision with trust history and user input. [PERSP] further describes Perspectives. The decentralized model used by Perspectives provides a high degree of reliability and availability, while protecting against single or even a few compromised "network notaries". Implementations of Perspectives are available at [Perspectives].

Convergence [Convergence] is another effort to implement concepts from the Perspectives project, as well as to augment those ideas to form a comprehensive solution. In particular, it addresses the problems of completeness, privacy, and responsiveness that existed in the original Perspectives work. Convergence notaries can also employ additional methods beyond network perspectives to decide whether a certificate should be trusted.

The Perspectives/Convergence approach can be used to establish confidence in a self-signed TLS server certificate, and in doing so, reduce the amount of certificate warnings that are presented to users.

D.1.4 DANE

Standards and products are still emerging in the area of DNS-based Authentication of Named Entities (DANE), and some of the standards are informational [RFC6394]. However, one of the following mechanisms can aid in the security of TLS server authentication and protect the clients from accepting unauthorized certificates issued due to the errors or compromise in CA or registration system and processes:

1. In addition to the server public key certificate validation as specified in Section 4.5, the client verifies that the TLS server certificate matches the one provided in the DNS records. Digital signatures on the DNS records are verified in accordance with the DNS Security Extensions (DNSSEC), as described in [RFC4033].

2. The client forgoes server public key certificate validation as specified in Section 4.5. Instead, the client verifies that the TLS server certificate matches the one provided in the DNS Records. Digital signatures on the DNS records are verified in accordance with the DNS Security Extensions (DNSSEC), as described in [RFC4033].

3. In addition to the server public key certificate validation, as specified in Section 4.5, the client verifies that the CA certificate in the certificate list provided by the server during a handshake matches the certificate provided in the DNS records and is part of the certification path verified as specified in Section 4.5. Digital signatures on the DNS Records are verified in accordance with the DNS Security Extensions (DNSSEC), as described in [RFC4033].

4. The client verifies that the TLS server certificate can be validated by the trust anchor provided in the DNS records. Digital signatures on the DNS records are verified in accordance with the DNS Security Extensions (DNSSEC), as described in [RFC4033].

D.2 Checking Server/Client Key Size

If the clients or servers wish to require certain key sizes or algorithms, they can implement cryptographic algorithm policy using the concept defined in [RFC5698]. The specification and processing of cryptographic algorithms policy as described in [RFC5698] can ensure that, regardless of the cipher suite specification in the TLS handshake, unacceptable algorithms and key sizes are not accepted by the entity (client or the server) who implements the cryptographic algorithms policy.

Appendix E References

The following list of documents, publications, and organizations provide a wide variety of information on varying aspects of Transport Layer Security.

[Adams99] Adams, C. and Lloyd, S., *Understanding PKI: Concepts, Standard, and Deployment Considerations*, (Macmillan Technology Publishing, Indianapolis, IN, ISBN 1-57870-166-X, 1999).

[CABBASE] Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates, CA Browser Forum, Version 1.1.6, 29 July 2013. https://www.cabforum.org/Baseline_Requirements_V1_1_6.pdf

[Comer00] Comer, D. E., *Internetworking with TCP/IP, Principles, Protocols, and Architectures*, Fourth Edition, (Prentice Hall, Upper Saddle River, NJ 07458, ISBN: 0-13- 018380-6, 2000).

[COMMON] X.509 Certificate Policy for the U.S. Federal PKI Common Policy Framework, Version 1.21, 18 December 2012. http://idmanagement.gov/documents/common-policy-framework-certificate-policy

[Convergence] Thoughtcrime Labs, *Convergence*, http://convergence.io/

[EVGUIDE] Guidelines For The Issuance and Management of Extended Validation Certificates, CA Browser Forum, Version 1.4.3, 9 July 2013. https://www.cabforum.org/Guidelines_v1_4_3.pdf

[FBCACP] X.509 Certificate Policy for the Federal Bridge Certification Authority, Version 2.26, 26 April 2012. http://www.idmanagement.gov/sites/default/files/documents/FBCA%20Certificate%20Policy%20v2.26_s.pdf

[FIPS140-1] FIPS 140-1, *Security Requirements For Cryptographic Modules*, http://csrc.nist.gov/publications/fips/fips140-1/fips1401.pdf

[FIPS140-2] FIPS 140-2, *Security Requirements For Cryptographic Modules*, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[FIPS140Impl] National Institute of Standards and Technology, *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*, 25 July 2013, http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

[FIPS180-4] National Institute of Standards and Technology, *Secure Hash Standard*, Federal Information Processing Standards Publication 180-4, March 2012, http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

[FIPS186-4] National Institute of Standards and Technology, *Digital Signature Standard*, Federal Information Processing Standard 186-4, July 2013, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[FIPS197] National Institute of Standards and Technology, *Advanced Encryption Standard (AES)*, Federal Information Processing Standard 197, November 26, 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[FIPS198-1] National Institute of Standards and Technology, *The Keyed-Hash Message Authentication Code (HMAC)*, Federal Information Processing Standard 198-1, July 2008, http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

[FIPS201-1] National Institute of Standards and Technology, *Personal Identification Verification (PIV) of Federal Employees and Contractors*, Federal Information Processing Standard 201-1, March 2006, http://csrc.nist.gov/publications/fips/fips201-1/FIPS-201-1-chng1.pdf

[Hall00] Hall, E. A., *Internet Core Protocols, The Definitive Guide*, (O'Reilly & Associates, ISBN: 1-56592-572-6, February 2000).

[Housley01] Housley, R. and Polk, T., *Planning for PKI, Best Practices Guide for Deploying Public Key Infrastructure*, (John Wiley & Sons, New York, NY, ISBN 0-471-39702-4, 2001).

[KeyMgmt03] National Institute of Standards and Technology's Computer Security Resource Center, *Key Management Information*, http://csrc.ncsl.nist.gov/CryptoToolkit/tkkeymgmt.html.

[Paterson11] Paterson, K. G., Ristenpart, T., and Shrimpton, T., *Tag Size* Does *Matter: Attacks and Proofs for the TLS Record Protocol*, in ASIACRYPT 2011, (Springer Lecture Notes in Computer Science, volume 7073, ISBN 978-3-642-25384-3).

[PERSP] Wendlandt D., Andersen D.G. and Perrig A., *Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing*, 2011 USENIX. http://perspectivessecurity.files.wordpress.com/2011/07/perspectives_usenix08.pdf.

[Perspectives] Perspectives Project, http://perspectives-project.org/

[Polk03] Polk, W., Hastings, N., and Malani, A., *Public Key Infrastructures that Satisfy Security Goals*, IEEE Internet Computing, Volume 7, Number 4, July-August, 2003.

[Rescorla01] Rescorla, E., *SSL and TLS – Designing and Building Secure Systems*, (Addison- Wesley, Upper Saddle River NJ, 07458, ISBN 0-201-61598, March 2001).

[RFC2119] Bradner, S., *Key words for use in RFCs to Indicate Requirement Levels*, Internet Engineering Task Force, Request for Comment 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2246] Dierks, T. and Allen, C., *The TLS Protocol Version 1.0*, Internet Engineering Task Force, Request for Comment 2246, January 1999, http://www.ietf.org/rfc/rfc2246.txt

[RFC3279] W. Polk et. al., *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate Revocation List (CRL) Profile*, Internet Engineering Task Force, Request for Comment 3279, April 2002, http://www.ietf.org/rfc/rfc3279.txt

[RFC3447] J. Jonsson and B. Kaliski, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, February 2003, http://www.ietf.org/rfc/rfc3447.txt

[RFC3713] M. Matsui et al. *A Description of the Camellia Encryption Algorithm*, Internet Engineering Task Force, Request for Comment 3713, April 2004, http://www.ietf.org/rfc/rfc3713.txt

[RFC3749] Hollenbeck, S., *Transport Layer Security Protocol Compression Methods*, Internet Engineering Task Force, Request for Comment 3749, May 2004, http://www.ietf.org/rfc/rfc3749.txt

[RFC3943] Friend, R., *Transport Layer Security (TLS) Protocol Compression Using Lempel-Ziv-Stac (LZS)*, Internet Engineering Task Force, Request for Comment 3943, November 2004, http://www.ietf.org/rfc/rfc3943.txt

[RFC4033] Arends, R. et al., *DNS Security Introduction and Requirements*, Internet Engineering Task Force, Request for Comment 4033, March 2005, http://www.ietf.org/rfc/rfc4033.txt

[RFC4055] Shaad, J. et al., *Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Internet Engineering Task Force, Request for Comment 4055, June 2005, http://www.ietf.org/rfc/rfc4055.txt.

[RFC4279] Eronen, P. and Tschofenig, H. *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, Internet Engineering Task Force, Request for Comments 4279, December 2005, http://www.ietf.org/rfc/rfc4279.txt

[RFC4346] Dierks, T. and Rescoria, E., *The Transport Layer Security (TLS) Protocol Version 1.1*, Internet Engineering Task Force, Request for Comment 4346, April 2006, http://www.ietf.org/rfc/rfc4346.txt

[RFC4492] Blake-Wilson, S. et al., *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)*, Internet Engineering Task Force, Request for Comment 4492, May 2006, http://www.ietf.org/rfc/rfc4492.txt

[RFC5246] Dierks, T. and Rescoria, E., *The Transport Layer Security (TLS) Protocol Version 1.2*, Internet Engineering Task Force, Request for Comment 5246, August 2008, http://www.ietf.org/rfc/rfc5246.txt

[RFC5280] Cooper, D. et. al., *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Internet Engineering Task Force, Request for Comment 5280, May 2008, http://www.ietf.org/rfc/rfc5280.txt

[RFC5288] Salowey, J., Choudhury, A., McGrew, D., *AES Galois Counter Mode (GCM) Cipher Suites for TLS*, Internet Engineering Task Force, Request for Comment 5288, August 2008, http://www.ietf.org/rfc/rfc5288.txt

[RFC5289] Rescorla, E., *TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)*, Internet Engineering Task Force, Request for Comment 5289, August 2008, http://www.ietf.org/rfc/rfc5289.txt

[RFC5487] Badra, M., *Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode*, Internet Engineering Task Force, Request for Comment 5487, March 2009, http://www.ietf.org/rfc/rfc5487.txt

[RFC5489] Badra, M., *ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)*, Internet Engineering Task Force, Request for Comment 5489, March 2009, http://www.ietf.org/rfc/rfc5489.txt

[RFC5698] Kunz, T., Okunick, S. and Pordesch U., *Data Structure for the Security Suitability of Cryptographic Algorithms (DSSC)*, Internet Engineering Task Force, Request for Comment 5698, November 2009, http://www.ietf.org/rfc/rfc5698.txt

[RFC5746] Rescorla E. et. al., *Transport Layer Security (TLS) Renegotiation Indication Extension*, Internet Engineering Task Force, Request for Comment 5746, February 2010, http://www.ietf.org/rfc/rfc5746.txt

[RFC5758] Q. Dang et. al., *Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA*, Internet Engineering Task Force, Request for Comment 5758, January 2010, http://www.ietf.org/rfc/rfc5758.txt

[RFC5759] Solanis, J. and Zieglar L., *Suite B Certificate and Certificate Revocation List (CRL) Profile*, Internet Engineering Task Force, Request for Comment 5759, January 2010, http://www.ietf.org/rfc/rfc5759.txt

[RFC6066] Eastlake, D., *Transport Layer Security (TLS) Extensions: Extension Definition*, Internet Engineering Task Force, Request for Comment 6066, January 2011, http://www.ietf.org/rfc/rfc6066.txt

[RFC6101] Freier, A. e al., *The Secure Sockets Layer (SSL) Protocol Version 3.0*, Internet Engineering Task Force, Request for Comment 6101, August 2011, http://www.ietf.org/rfc/rfc6101.txt

[RFC6394] R. Barnes, *Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE)*, Internet Engineering Task Force, Request for Comment 6394, October 2011, http://www.ietf.org/rfc/rfc6394.txt

[RFC6460] Salter, M. and Housley, R., *Suite B Profile for Transport Layer Security (TLS)*, Internet Engineering Task Force, Request for Comment 6460, January 2012, http://www.ietf.org/rfc/rfc6460.txt

[RFC6655] McGrew, D. and Bailey, D., *AES-CCM Cipher Suites for TLS*, Internet Engineering Task Force, Request for Comment 6655, July 2012, http://www.ietf.org/rfc/rfc6655.txt

[RFC6818] Yee, P., *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, Internet Engineering Task Force, Request for Comment 6818, January 2013, http://www.ietf.org/rfc/rfc6818.txt

[RFC6960] S. Santesson et. al., *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, Internet Engineering Task Force, Request for Comment 6960, June 2013, http://www.ietf.org/rfc/rfc6960.txt

[RFC6961] Pettersen, Y., *The Transport Layer Security (TLS) Multiple Certificate Status Request Extension*, Internet Engineering Task Force, Request for Comment 6961, June 2013, http://www.ietf.org/rfc/rfc6961.txt

[RFC6962] Laurie, B., et al., *Certificate Transparency*, Internet Engineering Task Force, Request for Comment 6962, June 2013, http://www.ietf.org/rfc/rfc6962.txt

[SOVER] Sovereign Key Cryptography for Internet Domains, Electronic Frontier Foundation https://git.eff.org/?p=sovereign-keys.git;a=blob_plain;f=sovereign-key-design.txt;hb=master

[SP800-32] NIST Special Publication 800-32, *Introduction to Public Key Technology and the Federal PKI Infrastructure*, February 2001, http://csrc.nist.gov/publications/nistpubs/800-32/sp800-32.pdf

[SP800-53] NIST Special Publication 800-53, *Security and Privacy Controls for Federal Information Systems and Organizations*, April 2013, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

[SP800-57p1] NIST Special Publication 800-57 Part 1, *Recommendation for Key Management – Part 1: General (Revision 3)*, July 2012, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf

[SP800-56A] NIST Special Publication 800-56A, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, May 2013, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf

[SP800-56B] NIST Special Publication 800-56B, *Recommendation for Pair-Wise Key Establishment Schemes Using Factorization Cryptography*, August 2009, http://csrc.nist.gov/publications/nistpubs/800-56B/sp800-56B.pdf

[SP800-63] NIST Special Publication 800-63-1, *Electronic Authentication Guide*, December 2011, http://csrc.nist.gov/publications/nistpubs/800-63-1/SP-800-63-1.pdf

[SP800-67] NIST Special Publication 800-67 Revision 1, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*, January 2012, http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

[SP800-90A] NIST Special Publication 800-90A Revision 1, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, January 2012, http://csrc.nist.gov/publications/nistpubs/800-90A/SP800-90A.pdf

[SP800-107] NIST Special Publication 800-107 Revision 1, *Recommendation for Applications Using Approved Hash Algorithms*, August 2012, http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf

[SP800-135] NIST Special Publication 800-135 Revision 1, *Recommendation for Existing Application Specific Key Derivation Functions*, December 2011, http://csrc.nist.gov/publications/nistpubs/800-135-rev1/sp800-135-rev1.pdf

[Schneier96] Schneier, B., *Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd ed.*, (John Wiley & Sons, Inc. 1996).