وزارة المواصلات والاتصالات
MINISTRY OF TRANSPORT AND COMMUNICATIONS

TLS Guidelines

Version: 1.0

Author: Public Key Management and Digital identity Section

Document Classification: Public

Published Date: March 2020

Document History:

Date	Issue	Status	Author
20/01/2020	0.1	First draft	PKI Team
02/02/2020	0.2	Appendix	PKI Team
05/02/2020	0.3	First Review	PKI Section
23/02/2020	0.4	Formatting, editing, review revisions	PKI Team
10/03/2020	0.5	Second Review	QCERT Team
11/03/2020	0.9	Formatting, Document for approval	PKI Team
11/03/2020	1.0	Document for Publish	CAP-PMA Director

Content

I. Audience

The guideline in this document are specifically targeted towards government and private entities. While these guidelines are primarily designed for users and system administrators, who are is familiar with TLS protocols and public-key infrastructure concepts, including, for example, X.509 certificates, to adequately protect sensitive but data against serious threats on the Internet, they may also be used within closed network environments to segregate data.

II. Scope

This document is a technical guide that should be followed. Deviations from TLS configuration herein shall be configured according to these guidelines. This document focus on the common use cases where clients and servers must interoperate with a wide variety of implementations, and authentication is performed using public-key certificates. The scope is further limited to TLS when used in conjunction with TCP/IP protocol.

III. TLS Overview

The Transport Layer Security (TLS) protocol is used to secure communications in a wide variety of online transactions including but not limited to financial, healthcare, and social, All network services, whether or not they handle personally identifiable information, financial data, login information need to protect the confidentiality and integrity of the transmitted information. 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 based on a precursor protocol called Secure Sockets Layer (SSL), and is an improvement.

1. TLS Subprotocols

There are three subprotocols in the TLS protocol that are used to control the session connection:

- **The TLS handshake protocol**: Is used to negotiate the session parameters. The handshake protocol is used to optionally exchange X.509 public-key certificates to authenticate the server to the client and may be used to authenticate the client to the server as well. It initializes both the client and server to use cryptographic capabilities by negotiating a cipher suite of algorithms and functions, including key establishment, digital signature, confidentiality, and integrity algorithms.
- **The alert protocol:** Is used to notify the other party of an error condition. The alert protocol is used to alert status changes to the peer. The primary use of this protocol is to report the cause of failure. Status changes include such things as error condition like invalid message received or message cannot be decrypted, as well as things like the connection has closed. One thing that TLS has over SSL is that TLS has more alert messages than SSL does.
- **The change cipher spec protocol:** is used in TLS 1.0, 1.1, and 1.2 to change the cryptographic parameters of a session. The change cipher spec protocol is used to change the encryption being used by the client and server. It is normally used as part of the handshake process to switch to symmetric key encryption. The CCS protocol is a single message that tells the peer that the sender wants to change to a new set of keys, which are then created from information exchanged by the handshake protocol.

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 and the version of TLS used. The negotiation phase consists on: A client sends a ClientHello message specifying the highest TLS protocol version it supports, a random number, a list of suggested cipher suites and suggested compression methods. If the client is attempting to perform a resumed handshake, it may send a session ID. If the client can use Application-Layer Protocol Negotiation, it may include a list of supported application protocols, such as HTTP. The server responds with a ServerHello message, containing the chosen protocol version, a random number, cipher suite and compression method from the choices offered by the client. To confirm or allow resumed handshakes the server may send a session ID. The chosen protocol version should be the highest that both the client and server support. The server sends its Certificate message. The server sends its ServerKeyExchange message. Then, the server sends a ServerHelloDone message, indicating it is done with handshake negotiation. the client responds with a ClientKeyExchange message, which may contain a PreMasterSecret, public key, or nothing. This PreMasterSecret is encrypted using the public key of the server certificate. The client and server then use the random numbers and PreMasterSecret to compute a common secret, called the "master secret". All other key data for this connection is derived from this master secret (and the client- and server-generated random values), which is passed through a carefully designed pseudorandom function.

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 and another for encryption by the server. The sender of a message encrypts the message using the appropriate derived encryption key; the receiver uses the same (independently derived) 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.

4. Integrity

The keyed MAC algorithm, specified by the negotiated cipher suite, provides message integrity. The Keyed MAC algorithm is a code accompanying data in order to ensure the integrity of the latter, allowing verification that they have not undergone any modification, after transmission. The concept is relatively like hash functions. These are also algorithms that create a small authenticator block of fixed size. The big difference is that this authenticator block is no longer based only on the message, but also on a secret key.

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 security parameters and extensions. SSL/TLS client authentication is intended for the client rather than a server. In server certificates, the client (browser) verifies the identity of the server. If it finds the server and its certificate are legitimate entities, it goes ahead and establishes a connection. The entire process happens during SSL/TLS handshake. In many cases, authentication is performed explicitly by verifying digital signatures using public keys that are present in certificates or implicitly using 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 in the server's certificate. In client authentication, a server (website) makes a client generate a keypair for authentication purpose. The private key, the heart of an SSL certificate, is kept with the client instead of the server. It's stored in the browser. The server confirms the authenticity of the private key and then paves the way for secure communication. Client authentication is optional and only occurs at the server's request. Client authentication is based on the client's public-key certificate.

6. Anti-Replay

TLS provides inherent protection against replay attacks. 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.

IV. TLS Servers Requirements

This section provides a minimum set of requirements that a server must implement in order to meet these guidelines. 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. Protocol Version Support

Recent versions of TLS are more secure than older versions. The oldest three versions of TLS, SSL 1.0, SSL 2.0 and SSL 3.0 cannot be used securely. The most recent version of TLS, TLS 1.3 offers the best protection. Servers that support government applications shall be configured to use TLS 1.2 and should be configured to use TLS 1.3 as well. These servers should not be configured to use TLS 1.1 and shall not use TLS 1.0, SSL 3.0, or SSL 2.0. Government entities shall support TLS 1.3 by January 1, 2024. After this date, servers shall support TLS 1.3. In general, servers that support TLS 1.3 should be configured to use TLS 1.2 as well. However, TLS 1.2 may be disabled on servers that support TLS 1.3 if it has been determined that TLS 1.2 is not needed for interoperability.

Version	Observation	
	Above January 2024	**After January 2024**
TLS 1.3	Good and should be used	Recommended and shall be used
TLS 1.2	Recommended and shall be used	Disabled if it is not needed for interoperability
TLS 1.1 TLS 1.0	Phase out and shall not be used	
SSL 3.0 SSL 2.0 SSL 1.0	Shall not be used securely	

Table 1: TLS Versions

2. Server Keys and Certificates

The TLS server shall be configured according to these guidelines:

- The server shall be configured with certificates issued by a CA, rather than self-signed certificates.

- The server shall be configured with one or more public key certificates and the associated private keys.
- The server shall be configured with an RSA key encipherment certificate.
- The server 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.
- Server certificates shall be issued by a CA that publishes revocation information in either CRLs or OCSP responses.
- The source for the revocation information shall be included in the certificate in the appropriate extension to promote interoperability.
- All server certificates shall be X.509 version 3 certificates.
- Both the public key contained in the certificate and the signature shall have at least 112 bits of security. In addition, ephemeral keys, when used to establish the master secret, shall have at least 112 bits of security.
- The certificate shall be signed with an algorithm consistent with the public key.
- The server should be configured to support the server authentication extended key usage extension.
- The certificate profile of Table 2 should be used for the server certificate.
- The server shall perform revocation checking of the client certificate, when client authentication is used.
- Revocation information shall be obtained by the server from one or more of the locations described in Section 3.2.2.
- When the server cannot obtain current revocation information, the decision to accept or reject a certificate should be made according to the organization policy.

2.1. Server Certificate Profile

The TLS server certificate shall be an X.509 version 3 certificate. The following rules apply to the certificates sent by the server:

- The server's end-entity certificate's public key (and associated restrictions) MUST be compatible with the selected authentication algorithm from the client's "signature_algorithms" extension (currently RSA, ECDSA, or EdDSA).
- The certificate MUST allow the key to be used for signing with a signature scheme indicated in the client's "signature_algorithms"/"signature_algorithms_cert" extensions.
- The "server_name" and "certificate_authorities" extensions are used to guide certificate selection. As servers MAY require the presence of the "server_name" extension, clients SHOULD send this extension, when applicable.

Prior to TLS 1.2, the server Certificate message required that the signing algorithm for the certificate be the same as the algorithm for the certificate key. If the server supports TLS versions prior to TLS 1.2, the certificate should be signed with an algorithm consistent with the public key.

The server certificate profile is listed in the below tables should be used:

Field	Critical	Value	Description
Version	N/A	2	Version 3
Serial Number	N/A	Unique positive integer	Must be unique

Field	Critical	Value	Description
Issuer Signature Algorithm	N/A	*Values by CA key type:*	
		sha256WithRSAEncryption {1 2 840 113549 1 1 11}, or stronger	CA with RSA key
		id-RSASSA-PSS {1 2 840 113549 1 1 10 }	CA with RSA key
		ecdsa-with-SHA256 {1 2 840 10045 4 3 2}, or stronger	CA with elliptic curve key
		id-dsa-with-sha256 {2 16 840 1 101 3 4 3 2}, or stronger	CA with DSA key
Issuer Distinguished Name (DN)	N/A	Unique X.500 issuing CA DN	A single value **should** be encoded in each Relative Distinguished Name (RDN). All attributes that are of DirectoryString type **should** be encoded as a PrintableString.
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	A single value **should** be encoded in each RDN. All attributes that are of DirectoryString type **should** be encoded as a PrintableString. If present, the CN attribute **should** be of the form: CN={host IP address | host DNS name}
Subject Public Key Information	N/A	*Values by certificate type:*	
		rsaEncryption {1 2 840 113549 1 1 1}	RSA signature certificate 2048-bit RSA key modulus or other approved lengths Parameters: NULL

		ecPublicKey {1 2 840 10045 2 1}	ECDSA signature certificate or ECDH certificate Parameters: namedCurve OID for named curve specified in SP 800-186.[15] The curve **should** 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 (2048-bit large prime, i.e., p)
		dhpublicnumber {1 2 840 10046 2 1}	DH certificate Parameters: p, g, q (2048-bit large prime, i.e., p)
Issuer's Signature	N/A	Same value as in Issuer Signature Algorithm	

Extensions			
Field	**Critical**	**Value**	**Description**
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 Public-Key Cryptography Standards (PKCS) 10 request or calculated by the issuing CA
Key Usage	Yes	*Values by certificate type:*	
		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		Optional
Subject Alternative Name (SAN)	No	DNS host name, or IP address if there is no DNS name assigned.	Required. Multiple SANs are permitted, e.g., for load balanced environments.

		Other name forms may be included, if appropriate.	
Authority Information Access	No	id-ad-caIssuers	Required. Access method entry contains HTTP URL for certificates issued to issuing CA
		id-ad-ocsp	Required. 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
Signed Certificate Timestamps List	No	See comments	Optional. This extension contains a sequence of Signed Certificate Timestamps, which provide evidence that the certificate has been submitted to Certificate Transparency logs.
TLS feature	No	status_request(5)	Optional. This extension (sometimes referred to as the "must staple" extension) may be present to indicate to clients that the server supports OCSP stapling and will provide a stapled OCSP response when one is requested.

Table 2: Server Certificate Profile

2.2. Obtaining Revocation Status Information for the Client Certificate

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

1. Certificate Revocation List (CRL) or OCSP 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.
4. CRL from the CRL Distribution Points extension in the client certificate.

The server will either deny the connection or accept a potentially revoked or compromised certificate if the local store does not have the latest 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 decision to accept or reject a certificate in this situation should be made according to organization policy.

3. Cryptographic Support

Cryptographic support in TLS is provided using various cipher suites. A cipher suite specifies a collection of algorithms for key exchange (in TLS 1.2 and earlier only) and for providing confidentiality and integrity services to application data. The following rules apply:

- The server shall be configured for data confidentiality and integrity services.
- The server shall be configured to only support cipher suites that are composed entirely of Approved algorithms.
- The server shall only support cipher suites for which it has a valid certificate containing a signature providing at least 112 bits of security.

3.1. Cipher Suites

Cipher suites specify the cryptographic algorithms that will be used for a session. Cipher suites in TLS 1.0 through TLS 1.2 have the form:

TLS_KeyExchangeAlg_WITH_EncryptionAlg_MessageAuthenticationAlg

Cipher suites are defined differently in TLS 1.3. These cipher suites do not specify the key exchange algorithm and have the form:

TLS_AEAD_HASH

In TLS, the cryptographic negotiation proceeds by the client. 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. TLS 1.3 cipher suites cannot be negotiated for TLS 1.2 connections, and TLS 1.2 cipher suites cannot be negotiated with TLS 1.3. Therefore, there is no guarantee that the negotiation will settle on the strongest common suite. If no cipher suites are common to the client and server, the connection is aborted.

The server shall be configured to only use cipher suites that are composed entirely of approved algorithms[1][2]. A complete list of acceptable cipher suites for general use is provided in this section, grouped by certificate type and TLS protocol version.

3.1.1. Cipher Suites for TLS 1.2 and Earlier Versions

- ***Cipher Suites for ECDSA Certificates***

ECDSA cipher suites use elliptical curve cryptography (ECC). Because of its smaller size, it is helpful in environments where processing power, storage space, bandwidth, and power consumption are constrained. TLS version 1.2 includes authenticated encryption modes and support for the SHA-256 and SHA-384 hash algorithms, which are not supported in prior versions of TLS. TLS

[1] Electronic Signature Algorithms Standard
[2] Qatar National Cryptographic Standard

1.2 servers that are configured with ECDSA certificates may be configured to support the following cipher suites, which are only supported by TLS 1.2:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CCM
- TLS_ECDHE_ECDSA_WITH_AES_256_CCM
- TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
- TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

TLS servers may be configured to support the following cipher suites when ECDSA certificates are used with TLS versions 1.2, 1.1, or 1.0:

- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

- ***Cipher Suites for RSA Certificates***

TLS 1.2 servers that are configured with RSA certificates may be configured to support the following cipher suites:

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_128_CCM
- TLS_DHE_RSA_WITH_AES_256_CCM
- TLS_DHE_RSA_WITH_AES_128_CCM_8
- TLS_DHE_RSA_WITH_AES_256_CCM_8
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256

TLS servers may be configured to support the following cipher suites when RSA certificates are used with TLS versions 1.2, 1.1, or 1.0:

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA

- ***Cipher Suites for DSA Certificates***

TLS 1.2 servers that are configured with DSA certificates may be configured to support the following cipher suites:

- TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
- TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA256

TLS servers may be configured to support the following cipher suites when DSA certificates are used with TLS versions 1.2, 1.1, or 1.0:

- TLS_DHE_DSS_WITH_AES_128_CBC_SHA
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA

- ***Cipher Suites for DH Certificates***

TLS 1.2 servers that are configured with DSA-signed DH certificates may be configured to support the following cipher suites:

- TLS_DH_DSS_WITH_AES_128_GCM_SHA256
- TLS_DH_DSS_WITH_AES_256_GCM_SHA384
- TLS_DH_DSS_WITH_AES_128_CBC_SHA256
- TLS_DH_DSS_WITH_AES_256_CBC_SHA256

TLS servers may be configured to support the following cipher suites when DSA-signed DH certificates are used with TLS versions 1.2, 1.1, or 1.0:

- TLS_DH_DSS_WITH_AES_128_CBC_SHA
- TLS_DH_DSS_WITH_AES_256_CBC_SHA

TLS 1.2 servers that are configured with RSA-signed DH certificates may be configured to support the following cipher suites:

- TLS_DH_RSA_WITH_AES_128_GCM_SHA256
- TLS_DH_RSA_WITH_AES_256_GCM_SHA384
- TLS_DH_RSA_WITH_AES_128_CBC_SHA256
- TLS_DH_RSA_WITH_AES_256_CBC_SHA256

TLS servers may be configured to support the following cipher suites when RSA-signed DH certificates are used with TLS versions 1.2, 1.1, or 1.0:

- TLS_DH_RSA_WITH_AES_128_CBC_SHA
- TLS_DH_RSA_WITH_AES_256_CBC_SHA

- ***Cipher Suites for ECDH Certificates***

TLS 1.2 servers that are configured with ECDSA-signed ECDH certificates may be configured to support the following cipher suites:

- TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384

TLS servers may be configured to support the following cipher suites when ECDSA-signed ECDH certificates are used with TLS versions 1.2, 1.1, or 1.0:

- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA

TLS 1.2 servers that are configured with RSA-signed ECDH certificates may be configured to support the following cipher suites:

- TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384

TLS servers may be configured to support the following cipher suites when RSA-signed ECDH certificates are used with TLS versions 1.2, 1.1, or 1.0:

- TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA

3.1.2. Cipher Suites for TLS 1.3

TLS 1.3 servers may be configured to support the following cipher suites:

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_AES_128_CCM_SHA256
- TLS_AES_128_CCM_8_SHA256

These cipher suites may be used with either RSA or ECDSA server certificates; DSA and DH certificates are not supported by TLS 1.3.

4. TLS Extension Support

System administrators must carefully configure servers to support extensions listed as mandatory. Only extensions whose specifications have an impact on security are described. In general, servers should only be configured to support extensions that are required by the application or that enhance security. Extensions that are not needed should not be enabled.

4.1. Mandatory TLS Extensions

The server shall support the use of the following TLS extensions:

- **Renegotiation Indication**: which contains a cryptographic binding to the enclosing TLS connection (if any) for which the renegotiation is being performed. The "extension data" field of this extension contains a "RenegotiationInfo" structure. Server implementations shall perform initial and subsequent renegotiations in accordance with RFC 5746 and RFC 8446.
- **Server Name Indication:** Is an extension of TLS by which a client indicates which hostname it is attempting to connect to at the start of the handshaking process. This allows a server to present multiple certificates on the same IP address and TCP port number and hence allows multiple secure (HTTPS) websites (or any other service over TLS) to be served by the same IP address without requiring all those sites to use the same certificate. The server shall be able to process and respond to the server name indication extension received in a ClientHello message as described in RFC 6066.
- **Extended Master Secret:** TLS Extended Master Secret Extension fix a Hole in TLS. An active attacker can synchronize two TLS sessions: The known risk of the attack comes into play when TLS-based authentication is used during a renegotiation. In this scenario, if a malicious server is able to synchronize session parameters between the victim client and server, it becomes possible to have attacker-provided data attributed to an authenticated client. More generally speaking, it allows the malicious server to spoof a legitimate web site. The server shall support the use of this extension.
- **Signature Algorithms**: Clients which desire the server to authenticate itself via a certificate MUST send the "signature_algorithms" extension. If a server is authenticating via a certificate and the client has not sent a "signature_algorithms" extension, then the server must abort the handshake with a "missing_extension" alert. Servers shall support the processing of the signature algorithms extension received in a ClientHello message. rules are described in Sections RFC 5246 and RFC 8446.
- **Certificate Status Request**: Also referred to as "OCSP stapling". Client can use this extension to request the server's copy of the current status of its certificate. The benefits of this extension include a reduced number of roundtrips and network delays for the client to verify the status of the server's certificate and a reduced load on the certificate issuer's status response servers, thus solving a problem that can become significant when the issued certificate is presented by a frequently visited server.

Extension	TLS version
Renegotiation Indication	Applies to TLS versions: 1.0, 1.1, 1.2
Server Name Indication	Applies to TLS versions: 1.0, 1.1, 1.2, 1.3
Extended Master Secret	Applies to TLS versions: 1.0, 1.1, 1.2
Signature Algorithms	Applies to TLS versions: 1.2, 1.3
Certificate Status Request	Applies to TLS versions: 1.0, 1.1, 1.2, 1.3

Table 3: Mandatory TLS extensions

4.2. Conditional TLS Extensions

The TLS server supports the following TLS extensions when the conditions stated in this section are met:

- **Fallback Signaling Cipher Suite Value (SCSV)**: shall be supported if the server supports versions of TLS prior to TLS 1.2 and does not support TLS 1.3. SCSV is defined in RFC 7507 and provides for a work around interoperability problems with legacy servers, many TLS user-agent implementations do not rely on the TLS protocol version negotiation mechanism alone but will intentionally reconnect using a downgraded protocol if initial handshake attempts fail.
- **Supported Groups:** Supported Groups extension shall be supported if the server supports ephemeral ECDH cipher suites or if the server supports TLS 1.3. this extension allows the client to indicate the domain parameter groups that it supports to the server.
- **Key Share:** Key Share extension shall be supported if the server supports TLS 1.3. The "key_share" extension contains the endpoint's cryptographic parameters. Client may send an empty client shares vector in order to request group selection from the server, at the cost of an additional round trip. Each KeyShareEntry value MUST correspond to a group offered in the "supported_groups" extension and MUST appear in the same order. However, the values MAY be a non-contiguous subset of the "supported_groups" extension and MAY omit the most preferred groups.
- **EC Point Format:** EC Point Format is an extension that indicates the set of point formats that the client can parse. This extension shall be supported if the server supports EC cipher suites. A client that proposes ECC cipher suites in its ClientHello message appends these extensions (along with any others), enumerating the curves it supports and the point formats it can parse. Client should send both the Supported Elliptic Curves Extension and the Supported Point Formats Extension. In the other side, server that receives a ClientHello containing one or both extensions MUST use the client's enumerated capabilities to guide its selection of an appropriate cipher suite. One of the proposed ECC cipher suites must be negotiated only if the server can successfully complete the handshake while using the curves and point formats supported by the client.
- **Multiple Certificate Status:** This extension should be supported if status information for the server's certificate is available via OCSP and the extension is supported by the server implementation. It allows a server to send multiple OCSP responses in the TLS handshake. Servers that have this capability and have certificates issued by CAs that support OCSP should be configured to support this extension. It allows the client to request the status of all certificates of the server in the TLS handshake. The server returns the revocation status of all the certificates in the server certificate chain and the client does not need to query any OCSP responders.
- **Trusted CA Indication:** "trusted_ca_keys" extension shall be supported if the server communicates with memory-constrained clients. Constrained clients that, due to memory limitations, possess only a small number of CA root keys may wish to indicate to servers which root keys they possess, in order to avoid repeated handshake failures. In order to indicate which CA root keys, they possess. client may include an extension of type "trusted_ca_keys" in the extended client hello. client may include none, some, or all the CA root keys they possess in this extension. Servers that receive a client hello containing the

"trusted_ca_keys" extension may use the information contained in the extension to guide their selection of an appropriate certificate chain to return to the client. In this event, the server shall include an extension of type "trusted_ca_keys" in the extended server hello.

- **Encrypt-then-MAC**: The use of encrypt-then-MAC is negotiated via TLS extensions. When connecting, the client includes the encrypt_then_mac extension in its client_hello if it wishes to use encrypt-then-MAC rather than the default MAC-then-encrypt. If the server can meet this requirement, it responds with an encrypt_then_mac in its server_hello. The "extension_type" value for this extension shall be 22, and the "extension_data" field of this extension shall be empty. The client and server must not use encrypt-then-MAC unless both sides have successfully exchanged encrypt_then_mac extensions. Encrypt-then-MAC This extension shall be supported if the server is configured to negotiate CBC cipher suites. This extension prevents several known attacks on CBC cipher suites.
- **Truncated HMAC:** In TLS, the entire output of the hash function is used as the MAC tag. However, it may be desirable in constrained environments to save bandwidth by truncating the output of the hash function to 80 bits when forming MAC tags. In order to negotiate the use of 80-bit truncated HMAC, client may include an extension of type "truncated_hmac" in the extended client hello. Servers that receive an extended hello containing a "truncated_hmac" extension MAY agree to use a truncated HMAC by including an extension of type "truncated_hmac" in the extended server hello. when new cipher suites are added that do not use HMAC, and the session negotiates one of these cipher suites, this extension will have no effect. This extension may be supported if the server communicates with constrained-device clients, cipher suites that use CBC mode are supported, and the server implementation does not support variable-length padding.
- **Pre-Shared Key:** The "pre_shared_key" extension is used to negotiate the identity of the pre-shared key to be used with a given handshake in association with PSK key establishment. Prior to accepting PSK key establishment, the server must validate the corresponding binder value. If this value is not present or does not validate, the server must abort the handshake. Servers should not attempt to validate multiple binders; rather, they should select a single PSK and validate solely the binder that corresponds to that PSK. In order to accept PSK key establishment, the server sends a “pre_shared_key" extension indicating the selected identity.
- **Pre-Shared Key Exchange Modes extension**: In order to use pre-shared keys, client MUST also send a "psk_key_exchange_modes" extension. A client shall provide a "psk_key_exchange_modes" extension if it offers a "pre_shared_key" extension. If clients offer "pre_shared_key" without a "psk_key_exchange_modes" extension, servers shall abort the handshake. Servers shall not select a key exchange mode that is not listed by the client. TLS 1.3 shall support this extension.
- **Supported Versions:** This extension shall be supported if the server supports TLS 1.3. The "supported_versions" extension is used by the client to indicate which versions of TLS it supports and by the server to indicate which version it is using. The extension contains a list of supported versions in preference order, with the most preferred version first. Implementations shall send this extension in the ClientHello containing all versions of TLS which they are prepared to negotiate. When this extension is provided in the ClientHello, server shall not use the ClientHello legacy version value for version negotiation and shall use

only the "supported_versions" extension to determine client preferences. Server shall only select a version of TLS present in that extension and shall ignore any unknown versions that are present in that extension. If this extension is absent from the ClientHello, server which also support TLS 1.2 MUST negotiate TLS 1.2 or prior. Server may abort the handshake upon receiving a ClientHello with legacy version.

- **Cookie:** Two main applications of cookies: First it allows the server to force the client to demonstrate reachability at their apparent network address. That really is, foremost, useful for non-connection-oriented transports and second it allows Allowing the server to offload state to the client, thus allowing it to send a HelloRetryRequest without storing any state. The server can do this by storing the hash of the ClientHello. This extension shall be supported if the server supports TLS 1.3.
- **Certificate Signature Algorithms:** The Certificate Signature Algorithms extension applies to signatures in certificates. The "signature_algorithms_cert" extension was added to allow implementations which supported different sets of algorithms for certificates and in TLS itself to clearly signal their capabilities. TLS 1.2 implementations SHOULD also process this extension. This extension shall be supported if the server supports TLS 1.3 and should be supported for TLS 1.2.
- **Post-handshake Client Authentication:** Post-handshake Client Authentication "post_handshake_auth" extension is used to indicate that a client is willing to perform post-handshake authentication. Servers shall not send a post-handshake CertificateRequest to clients which do not offer this extension.
- **Signed Certificate Timestamps:** Server operators can deliver signed certificate timestamps by using a special TLS extension (signed_certificate_timestamp). In this case, the CA issues the certificate to the server operator, and the server operator submits the certificate to the log. The log sends the signed certificate timestamp to the server operator, and the server operator uses a TLS extension with type signed_certificate_timestamp to deliver the signed certificate timestamp to the client during the TLS handshake. This extension should be supported if the server's certificate was issued by a publicly trusted CA and the certificate does not include a Signed Certificate Timestamps List extension.

Extension	TLS version
Fallback Signaling Cipher Suite Value	Applies to TLS versions: 1.0, 1.1, 1.2
Supported Groups	Applies to TLS versions: 1.0, 1.1, 1.2, 1.3
Key Share	Applies to TLS version 1.3
Supported Point Formats	Applies to TLS versions: 1.0, 1.1, 1.2
Multiple Certificate Status	Applies to TLS versions: 1.0, 1.1, 1.2
Trusted CA Indication	Applies to TLS versions: 1.0, 1.1, 1.2
Encrypt-then-MAC	Applies to TLS versions: 1.0, 1.1, 1.2
Truncated HMAC	Applies to TLS versions: 1.0, 1.1, 1.2
Pre-Shared Key	Applies to TLS version 1.3
Pre-Shared Key Exchange Modes	Applies to TLS version 1.3
Supported Versions	Applies to TLS version 1.3
Cookie	Applies to TLS version 1.3
Certificate Signature Algorithms	Applies to TLS versions: 1.2, 1.3
Post-handshake Client Authentication	Applies to TLS version 1.3
Signed Certificate Timestamps	Applies to TLS versions: 1.0, 1.1, 1.2, 1.3

Table 4: Optional TLS Extensions

5. Client Authentication

- When server supports client authentication, it shall support certificate-based client authentication.
- The server shall check the public key size of the client, if necessary, to ensure that the public key of the client can offer at least 112 bits of security.
- 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.
- The server shall validate the client certificate in accordance with the certification path validation rules specified in RFC5280 when client authentication is performed.
- 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).
- If the server supports OCSP, then OCSP checking shall be in compliance with RFC6960 and should use only one of the options described in the mentioned document.
- 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 RFC5280.
- 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.
- The default set of trust anchors for the server shall be examined to determine if any of them are required for client authentication.
- The server shall check the client key length if client authentication is performed, and the server implementation provides a mechanism to do so.
- Organization shall use the key size guidelines to check the client key size.
- The server shall be configured to 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.
- The server hints list shall be distinct from the server's trust anchor store.
- 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.

V. TLS Clients Requirements

This section includes a basic set of requirements for adherence to these guidelines that a TLS client must satisfy.

1. Protocol Version Support

The client shall be configured to use TLS 1.2. The client shall not be configured to use SSL 2.0 or SSL 3.0. Organizations shall support TLS 1.3 by January 1, 2024. After this date, clients shall be configured to use TLS 1.3. In general, clients that support TLS 1.3 should be configured to use TLS 1.2 as well. However, TLS 1.2 may be disabled on clients that support TLS 1.3 if TLS 1.2 is not needed for interoperability.

2. Client Certificate Profile

The TLS client certificate shall be an X.509 version 3 certificate. Both the public key contained in the certificate and the signature shall provide at least 112 bits of security. When certificate-based client authentication is required, 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. If the client supports TLS versions prior to TLS 1.2, the certificate should be signed with an algorithm that is consistent with the public key:

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

The client certificate profile is listed in below table. This profile should be used for client certificates.

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 CA key type:	
		sha256WithRSAEncryption {1 2 840 113549 1 1 11}, or stronger	CA with RSA key
		id-RSASSA-PSS { 1 2 840 113549 1 1 10}	CA with RSA key
		ecdsa-with-SHA256 {1 2 840 10045 4 3 2}, or stronger	CA with elliptic curve key

		id-dsa-with-sha256 {2 16 840 1 101 3 4 3 2}, or stronger	CA with DSA key
Issuer Distinguished Name	N/A	Unique X.500 Issuing CA DN	A single value **should** be encoded in each RDN. All attributes that are of directoryString type **should** 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	A single value **should** be encoded in each RDN. All attributes that are of directoryString type **should** 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 signature certificate 2048-bit RSA key modulus, or other approved lengths as defined in [45]and [5] Parameters: NULL
		ecPublicKey {1 2 840 10045 2 1}	ECDSA signature certificate or ECDH certificate Parameters: namedCurve OID for names curve specified in SP 800-186. The curve **should** 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	Same value as in Issuer Signature Algorithm	

Extensions			
Field	**Critical**	**Value**	**Description**
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	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}	The anyExtendedKeyUsage OID **should** be present if the extended key usage extension is included, but there is no intention to limit the types of applications with which the certificate may be used (e.g., the certificate is a general-purpose authentication certificate).
			Prohibited: all others unless consistent with key usage extension
Certificate Policies	No	Per issuer's 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		Optional: HTTP value in distributionPoint field pointing to a full and complete CRL. Prohibited: reasons and cRLIssuer fields, and nameRelativetoCRLIssuer CHOICE

Table 5: TLS Client Certificate Profile

If a client has several certificates that satisfy the TLS server requirements, the TLS client may request that the user choose from a list of certificates. Client certificates are also filtered by TLS clients based on an ability to build a path to one of the trust anchors in the hints list sent by the server.

3. Obtaining Revocation Status Information for the Server Certificate

The client shall perform revocation checking of the server certificate. The client may receive Revocation information from one of the following locations:

- OCSP response or responses in the server's CertificateStatus message.
- Certificate Revocation List (CRL) or OCSP response in the client's local certificate store.
- OCSP response from a locally configured OCSP responder.
- OCSP response from the OCSP responder location identified in the OCSP field in the Authority Information Access extension in the server certificate.

- CRL from the CRL Distribution Point extension in the server certificate.

When the server does not provide the revocation status, the local certificate store does not have the current CRL or OCSP response, and the OCSP responder and the CRL distribution point are unavailable (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 certificate in this situation should be made according to organization policy.

4. Cipher Suites

The recommended cipher suites for a TLS client are the same as those for a TLS server. The client should not be configured to use cipher suites other than those listed in Section 3.1.

5. TLS Extension Support

In general, it is advised that clients only be configured to support extensions that are required for interoperability or enhance security. Extensions that are not needed should not be enabled.

5.1. Mandatory TLS Extensions

The client shall be configured to use the following extensions:

- Renegotiation Indication
- Server Name Indication
- Extended Master Secret
- Signature Algorithms
- Certificate Status Request

5.2. Conditional TLS Extensions

The Client supports the use of the following TLS extensions under the circumstances described in the following subsections:

- Fallback Signaling Cipher Suite Value (SCSV) shall be supported if the client supports versions of TLS prior to TLS 1.2 and does not support TLS 1.3.
- Supported Groups extension shall be supported if the client supports ephemeral ECDH cipher suites or if the client supports TLS 1.3.
- Key Share extension shall be supported if the client supports TLS 1.3.
- The EC Point Format TLS extension shall be supported if the client supports EC cipher suite(s).
- Multiple Certificate Status extension should be enabled if the extension is supported by the client implementation.
- Trusted CA Indication extension should be supported by clients that run on memory constrained devices where only a small number of CA root keys are stored.
- Encrypt-then-MAC extension shall be supported when CBC mode cipher suites are configured.
- Truncated HMAC extension may be supported by clients that run on constrained devices when variable-length padding is not supported and cipher suites that use CBC mode are supported.
- Pre-Shared Key extension may be supported by TLS 1.3 clients.

- Pre-Shared Key Exchange Modes extension shall be supported by TLS 1.3 clients that support the Pre-Shared Key extension.
- Supported Versions extension shall be supported by TLS 1.3 clients.
- Cookie extension shall be supported by TLS 1.3 clients.
- Certificate Signature Algorithms Extension shall be supported if the client supports TLS 1.3 and should be supported for TLS 1.2.
- Post-handshake Client Authentication extension may be supported if the client supports TLS 1.3.

6. 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. The client may use all or a subset of the following resources to build the certification path: the local certificate store, certificates received from the server during the handshake, LDAP.
- The client shall validate the server certificate in accordance with the certification path validation rules specified in RFC 5280. The client shall terminate the TLS connection if path validation fails.
- Clients shall not overpopulate their trust stores with various CA certificates that can be verified via cross-certification.
- The client shall check the public key length of the server if a mechanism for checking is given by client implementation. The client shall also check the server public key length if the server uses ephemeral keys for the creation of the master secret and the client implementation provides a mechanism for checking.

VI. Recommendations for System Administrators

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

1. Digital Certificates

- System administrators shall use Server Certificate Profile, Obtaining Revocation Status and Information for the Client Certificate and Client Certificate Profile to identify an appropriate source for certificates.

2. Version Support

- System administrators shall develop migration plans to support TLS 1.3 by January 1, 2024.
- System administrators shall install, maintain, and update certificates in accordance with the certificate recommendations described on PKI Certificates and Key Management Standard.

3. Client Authentication

- 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 only the minimum set of trust anchors needed.

4. Operational Considerations

- System administrators shall ensure That the server shall operate on a secure operating system.
- System administrators shall ensure that TLS servers include appropriate network security protections.
- 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.
- For sensitive applications, system administrators should use a FIPS 140 Level 2, CC EAL 4 or higher hardware cryptographic module to protect server private keys.
- The system administrator shall ensure that the server and associated platform are kept up to date in terms of security patches.

VII. TLS Best Practices

This section describes additional TLS best practices:

- **Ensure that the Certificate Is Valid, issued from a Trusted CA and Compliant:** The Server certificate is the weakest point of a TLS server configuration. A certificate that is not trusted renders TLS useless. Certificates obtained should be automatable, short-lived and published to Certificate Transparency logs. All certificates must include the domain in the Subject Alternative Name. All TLS certificates should be free of the following:
    - Domain name mismatch, including Subject Alternative Name (SAN) errors
    - Certificate not yet valid
    - Certificate expired
    - Certificate revoked
    - Certificates lasting longer than three years
    - Use of a self-signed certificate
    - Use of a certificate that is not trusted (unknown CA or some other validation error)
    - Insecure key
- **Certificate Transparency:** Certificate Transparency is a mechanism to allow domain owners to detect mis issuance of certificates, and its use is growing rapidly by browsers and commercial certificate authorities. Web server certificates should be submitted to public certificate transparency logs wherever feasible. System administrators are encouraged to choose commercial certificate authorities which automatically publish all certificates to certificate transparency logs.
- **Protect Private Keys:** System administrators shall treat private keys as an important asset, restricting access to the smallest possible group of employees (if needed and when applicable). These recommendations shall be followed:
    - System administrators shall generate private keys and Certificate Signing Requests (CSRs) on a trusted computer. Some CAs offer to generate keys and a CSRs for you, but that's inappropriate.
    - System administrators and employees accessing private keys shall never transmit by email an unencrypted private key, or the decryption passphrase to an encrypted

private key. In general, do not use email for anything related to private key management.
    - Password-protect keys to prevent compromise when they are stored in backup systems.
    - Systems administrators, after compromise, shall revoke old certificates and generate new keys to use with new certificates.
    - Systems administrators shall renew certificates before the expiring date and always with new private keys.
- **Enough Hostname Coverage:** System administrators should ensure that certificates cover all the names to use with a website. They should ensure that the certificate works with and without the www prefix. A secure web server should have a certificate that is valid for every DNS name configured to point to it. Wildcard certificates have their uses but should be avoided if using them means exposing the underlying keys to a larger group of people, and especially if crossing organizational boundaries.
- **Encryption:** The only way to reliably protect web site communication is to enforce encryption throughout without exception to include the entire website. System administrators should avoid mixed content (websites that mix TLS and non-TLS content).
- **Client-Initiated Renegotiation**: In TLS, renegotiation allows parties to stop exchanging data in order to renegotiate how the communication is secured. There are some cases in which renegotiation needs to be initiated by the server, but there is no known need for clients to do so. System administrators should ensure to disable Client-Initiated Renegotiation.
- **TLS_FALLBACK_SCSV Implementation:** TLS_FALLBACK_SCSV is a TLS Signaling Cipher Suite Value (SCSV) that can be used to guard against protocol downgrade attacks. TLS_FALLBACK_SCSV prevents protocol downgrade attacks on the TLS protocol.
- **Strict Transport Security (HSTS) Implementation**: Websites and services available over HTTPS must enable HTTP Strict Transport Security (HSTS) to instruct compliant browsers to assume HTTPS going forward. This reduces the number of insecure redirects and protects users against attacks that attempt to downgrade connections to plain HTTP.

References

[1] Qatar National Cryptographic Standard
[2] PKI Certificates and Key Management Standard
[3] Electronic Signature Algorithms Standard
[7] **NIST.SP.800-57pt1r4**: Recommendation for Key Management Part 1: General (2016).
[8] **NIST.SP.800-56Ar3**: Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography (2018).
[9] **NIST.SP.800-56Br2**: Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography (2019).
[10] **NIST.SP.800-90Ar1**: Recommendation for Random Number Generation Using Deterministic Random Bit Generators (2015).
[11] **NIST.SP.800-67r2**: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher (2017).
[12] **NIST.SP.800-131Ar2**: Transitioning the Use of Cryptographic Algorithms and Key Lengths (2019).
[13] **NIST.SP.800-52**: Guidelines for the Selection and Use of Transport Layer Security (TLS) Implementations (2005).
[14] **RFC4033**: DNS Security Introduction and Requirements (2005).
[15] **RFC5487**: Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode (2009).
[16] **RFC5489:** ECDHE_PSK Cipher Suites for Transport Layer Security (TLS) (2009).
[17] **RFC7627**: Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension (2015).
[18] **RFC2119**: Key words for use in RFCs to Indicate Requirement Levels (1997).
[19] **RFC5280**: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (2008).
[20] **RFC4346:** The Transport Layer Security (TLS) Protocol Version 1.1 (2006).
[21] **RFC5246:** The Transport Layer Security (TLS) Protocol Version 1.2 (2008).
[22] **RFC6066:** Transport Layer Security (TLS) Extensions: Extension Definitions (2011).
[23] **RFC4279:** Pre-Shared Key Ciphersuites for Transport Layer Security (TLS) (2005).
[24] **RFC7366:** Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) (2014).
[25] **RFC8422:** Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier (2018).
[26] **RFC6961:** The Transport Layer Security (TLS) Multiple Certificate Status Request Extension (2013).
[27] **RFC8446:** The Transport Layer Security (TLS) Protocol Version 1.3 (2018).
[28] **RFC5746**: Transport Layer Security (TLS) Renegotiation Indication Extension (2010).
[29] **RFC6069**: Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP (2013).
[30] **CA/Browser Forum** (2019) Baseline Requirements Certificate Policy for the Issuance and Management of Publicly-Trusted Certificates. Available at https://cabforum.org/baseline-requirements-documents/
[31] **CA/Browser Forum** (2019) Guidelines For The Issuance And Management Of Extended Validation Certificates. Available at https://cabforum.org/extended-validation
[32] **RFC7507**: TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks (2015).

Appendix A: Determining the Need for TLS 1.0 and 1.1[3]

Enabling TLS 1.0 or 1.1 when they are not needed may leave systems and users vulnerable to attacks (such as the BEAST attack and the Klima attack). However, disabling older versions of TLS when there is a need may deny access to users who are unable to install or upgrade to a client that is capable of TLS 1.3 or 1.2.

The system administrator must consider the benefits and risks of using TLS 1.0 or 1.1, in the context of applications supported by the server, and decide whether the benefits of using TLS 1.0 or 1.1 outweigh the risks. This decision should be driven by the service(s) running on the server and the versions supported by clients accessing the server. Services that do not access high-value information (such as personally identifiable information or financial data) may benefit from using TLS 1.0 by increasing accessibility with little increased risk. On the other hand, services that do access high-value data may increase the likelihood of a breach for relatively little gain in terms of accessibility. The decision to support TLS 1.0 or 1.1 must be technically assessed on a case-by-case basis. This is to ensure that supporting older TLS versions is necessary and that associated risks and business implications are understood and accepted.

These guidelines do not give specific recommendations on steps that can be taken to make this determination. There are tools available (such as the Data Analytics Program) that can provide information to system administrators that can be used to assess the impact of supporting or not supporting TLS versions prior to TLS 1.2. For example, DAP data on visitor OS and browser versions can help administrators determine what percentage of visitors to agency websites cannot negotiate recommended TLS versions by default.

Many products that implement TLS 1.1 also implement TLS 1.2. Because of this, it may be unnecessary for servers to support TLS 1.1. Administrators can determine whether TLS 1.1 is needed by assessing whether it must support connections with clients where TLS 1.1 is the highest version available.

[3] NIST.SP.800-52: Guidelines for the Selection and Use of Transport Layer Security (TLS) Implementations.

Appendix B: Other Considerations

- **Random number generators:** To ensure the security of cryptographic algorithms, a good source of entropy and a good generator of random numbers (pseudo random number generator PRNG) are required. The source of entropy supplies random data and is input to the PRNG. The PRNG turns this random data into uniformly distributed random numbers. In particular, the requirement for randomness is relevant for (but not limited to):
    - Generation of keys for certificates.
    - Generation of temporary keys and proof of secret key possession for forward secrecy.

  Generating enough entropy may be a bottleneck if a TLS server is under heavy load. The addition of a hardware module (hardware random number generator) to the server ensures the availability of enough entropy. Many modern processors integrate a hardware module to harvest randomness.
- **Certificate management:** Acquiring and managing certificates is not part of these guidelines[4]. However, good certificate management is an important condition for the safe deployment of TLS. That is why we list some points of interest:
    - Secret key generation: System administrators shall use a good random number generator to generate the secret key. Make sure you generate this key on a trusted system, for example a Hardware Security Module (HSM) or a computer that is physically disconnected from the internet. A key generated on a disconnected computer is then deployed on the server that will offer the certificate.
    - Certificate service provider: System administrators shall shoose a trustworthy CSP to issue and sign the certificate.
    - Domain names: System administrators shall ensure that the certificate contains a list of domain names (fully qualified domain names, FQDNs) that it applies to. System administrators shall make sure the certificate covers all domains for which it is used, including subdomains.
    - Extended validation Many certificate issuers also issue Extended Validation (EV) certificates. An EV certificate provides some assurance on the identity of the owner. Developers of client software have been removing the prominent visual distinctions between EV and normal certificates. EV certificates are usually more expensive than a normal certificate. A risk analysis can help to determine if an EV certificate is warranted.
    - Files on the server: The administrator of the server needs to include intermediate CA's between the root CA and the issued certificate on the server. The server offers these during TLS connections. The secret key of the own certificate needs to be adequately protected. A secret key can be stored in an HSM. An HSM is designed to offer physical protection against extraction of the secret key.
    - Administration: System administrators shall keep an administration of all certificates that are in use within the organization. If a certificate needs to be replaced, this will make it easier to determine where it is in use. System administrators shall include the expiration time for all certificates, to ensure timely replacement. Expired

[4] PKI Certificates and Key Management Standard

certificates should never be used. Some certificate issuers support mechanisms for automated renewal and deployment of