Stream:	Internet Engineering Task Force (IETF)
RFC:	9325
BCP:	195
Obsoletes:	7525
Updates:	5288, 6066
Category:	Best Current Practice
Published:	November 2022
ISSN:	2070-1721
Authors:	Y. Sheffer	P. Saint-Andre	T. Fossati
---	---	---	---
	Intuit	*Independent*	*ARM Limited*

RFC 9325
Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)

Abstract

Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) are used to protect data exchanged over a wide range of application protocols and can also form the basis for secure transport protocols. Over the years, the industry has witnessed several serious attacks on TLS and DTLS, including attacks on the most commonly used cipher suites and their modes of operation. This document provides the latest recommendations for ensuring the security of deployed services that use TLS and DTLS. These recommendations are applicable to the majority of use cases.

RFC 7525, an earlier version of the TLS recommendations, was published when the industry was transitioning to TLS 1.2. Years later, this transition is largely complete, and TLS 1.3 is widely available. This document updates the guidance given the new environment and obsoletes RFC 7525. In addition, this document updates RFCs 5288 and 6066 in view of recent attacks.

Status of This Memo

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9325.

Copyright Notice

Table of Contents

1. Introduction

Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) are used to protect data exchanged over a wide variety of application protocols, including HTTP [RFC9112] [RFC9113], IMAP [RFC9051], Post Office Protocol (POP) [STD53], SIP [RFC3261], SMTP [RFC5321], and the Extensible Messaging and Presence Protocol (XMPP) [RFC6120]. Such protocols use both the TLS or DTLS handshake protocol and the TLS or DTLS record layer. Although the TLS

handshake protocol can also be used with different record layers to define secure transport protocols (the most prominent example is QUIC [RFC9000]), such transport protocols are not directly in scope for this document; nevertheless, many of the recommendations here might apply insofar as such protocols use the TLS handshake protocol.

Over the years leading to 2015, the industry had witnessed serious attacks on TLS and DTLS, including attacks on the most commonly used cipher suites and their modes of operation. For instance, both the AES-CBC [RFC3602] and RC4 [RFC7465] encryption algorithms, which together were once the most widely deployed ciphers, were attacked in the context of TLS. Detailed information about the attacks known prior to 2015 is provided in a companion document [RFC7457] to the previous version of the TLS recommendations [RFC7525], which will help the reader understand the rationale behind the recommendations provided here. That document has not been updated in concert with this one; instead, newer attacks are described in this document, as are mitigations for those attacks.

The TLS community reacted to the attacks described in [RFC7457] in several ways:

- Detailed guidance was published on the use of TLS 1.2 [RFC5246] and DTLS 1.2 [RFC6347] along with earlier protocol versions. This guidance is included in the original [RFC7525] and mostly retained in this revised version; note that this guidance was mostly adopted by the industry since the publication of RFC 7525 in 2015.
- Versions of TLS earlier than 1.2 were deprecated [RFC8996].
- Version 1.3 of TLS [RFC8446] was released, followed by version 1.3 of DTLS [RFC9147]; these versions largely mitigate or resolve the described attacks.

Those who implement and deploy TLS and TLS-based protocols need guidance on how they can be used securely. This document provides guidance for deployed services as well as for software implementations, assuming the implementer expects their code to be deployed in the environments defined in Section 5. Concerning deployment, this document targets a wide audience, namely all deployers who wish to add authentication (be it one-way only or mutual), confidentiality, and data integrity protection to their communications.

The recommendations herein take into consideration the security of various mechanisms, their technical maturity and interoperability, and their prevalence in implementations at the time of writing. Unless it is explicitly called out that a recommendation applies to TLS alone or to DTLS alone, each recommendation applies to both TLS and DTLS.

This document attempts to minimize new guidance to TLS 1.2 implementations, and the overall approach is to encourage systems to move to TLS 1.3. However, this is not always practical. Newly discovered attacks, as well as ecosystem changes, necessitated some new requirements that apply to TLS 1.2 environments. Those are summarized in Appendix A.

Naturally, future attacks are likely, and this document cannot address them. Those who implement and deploy TLS/DTLS and protocols based on TLS/DTLS are strongly advised to pay attention to future developments. In particular, although it is known that the creation of quantum computers will have a significant impact on the security of cryptographic primitives

and the technologies that use them, currently post-quantum cryptography is a work in progress and it is too early to make recommendations; once the relevant specifications are standardized in the IETF or elsewhere, this document should be updated to reflect best practices at that time.

As noted, the TLS 1.3 specification resolves many of the vulnerabilities listed in this document. A system that deploys TLS 1.3 should have fewer vulnerabilities than TLS 1.2 or below. Therefore, this document replaces [RFC7525], with an explicit goal to encourage migration of most uses of TLS 1.2 to TLS 1.3.

These are minimum recommendations for the use of TLS in the vast majority of implementation and deployment scenarios, with the exception of unauthenticated TLS (see Section 5). Other specifications that reference this document can have stricter requirements related to one or more aspects of the protocol, based on their particular circumstances (e.g., for use with a specific application protocol); when that is the case, implementers are advised to adhere to those stricter requirements. Furthermore, this document provides a floor, not a ceiling: where feasible, administrators of services are encouraged to go beyond the minimum support available in implementations to provide the strongest security possible. For example, based on knowledge about the deployed base for an existing application protocol and a cost-benefit analysis regarding security strength vs. interoperability, a given service provider might decide to disable TLS 1.2 entirely and offer only TLS 1.3.

Community knowledge about the strength of various algorithms and feasible attacks can change quickly, and experience shows that a Best Current Practice (BCP) document about security is a point-in-time statement. Readers are advised to seek out any errata or updates that apply to this document.

This document updates [RFC5288] in view of the [Boeck2016] attack. See Section 7.2.1 for the details.

This document updates [RFC6066] in view of the [ALPACA] attack. See Section 3.7 for the details.

2. Terminology

A number of security-related terms in this document are used in the sense defined in [RFC4949], including "attack", "authentication", "certificate", "cipher", "compromise", "confidentiality", "credential", "data integrity", "encryption", "forward secrecy", "key", "key length", "self-signed certificate", "strength", and "strong".

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. General Recommendations

This section provides general recommendations on the secure use of TLS. Recommendations related to cipher suites are discussed in the following section.

3.1. Protocol Versions

3.1.1. SSL/TLS Protocol Versions

It is important both to stop using old, less secure versions of SSL/TLS and to start using modern, more secure versions; therefore, the following are the recommendations concerning TLS/SSL protocol versions:

- Implementations **MUST NOT** negotiate SSL version 2.

  Rationale: Today, SSLv2 is considered insecure [RFC6176].

- Implementations **MUST NOT** negotiate SSL version 3.

  Rationale: SSLv3 [RFC6101] was an improvement over SSLv2 and plugged some significant security holes but did not support strong cipher suites. SSLv3 does not support TLS extensions, some of which (e.g., renegotiation_info [RFC5746]) are security critical. In addition, with the emergence of the Padding Oracle On Downgraded Legacy Encryption (POODLE) attack [POODLE], SSLv3 is now widely recognized as fundamentally insecure. See [RFC7568] for further details.

- Implementations **MUST NOT** negotiate TLS version 1.0 [RFC2246].

  Rationale: TLS 1.0 (published in 1999) does not support many modern, strong cipher suites. In addition, TLS 1.0 lacks a per-record Initialization Vector (IV) for cipher suites based on cipher block chaining (CBC) and does not warn against common padding errors. This and other recommendations in this section are in line with [RFC8996].

- Implementations **MUST NOT** negotiate TLS version 1.1 [RFC4346].

  Rationale: TLS 1.1 (published in 2006) is a security improvement over TLS 1.0 but still does not support certain stronger cipher suites that were introduced with the standardization of TLS 1.2 in 2008, including the cipher suites recommended for TLS 1.2 by this document (see Section 4.2 below).

- Implementations **MUST** support TLS 1.2 [RFC5246].

  Rationale: TLS 1.2 is implemented and deployed more widely than TLS 1.3 at this time, and when the recommendations in this document are followed to mitigate known attacks, the use of TLS 1.2 is as safe as the use of TLS 1.3. In most application protocols that reuse TLS and DTLS, there is no immediate need to migrate solely to TLS 1.3. Indeed, because many application clients are dependent on TLS libraries or operating systems that do not yet support TLS 1.3, proactively deprecating TLS 1.2 would introduce significant interoperability issues, thus harming security more than helping it. Nevertheless, it is expected that a future version of this BCP will deprecate the use of TLS 1.2 when it is appropriate to do so.

- Implementations **SHOULD** support TLS 1.3 [RFC8446] and, if implemented, **MUST** prefer to negotiate TLS 1.3 over earlier versions of TLS.

  Rationale: TLS 1.3 is a major overhaul to the protocol and resolves many of the security issues with TLS 1.2. To the extent that an implementation supports TLS 1.2 (even if it defaults to TLS 1.3), it **MUST** follow the recommendations regarding TLS 1.2 specified in this document.

- New transport protocols that integrate the TLS/DTLS handshake protocol and/or record layer **MUST** use only TLS/DTLS 1.3 (for instance, QUIC [RFC9001] took this approach). New application protocols that employ TLS/DTLS for channel or session encryption **MUST** integrate with both TLS/DTLS versions 1.2 and 1.3; nevertheless, in rare cases where broad interoperability is not a concern, application protocol designers **MAY** choose to forego TLS 1.2.

  Rationale: Secure deployment of TLS 1.3 is significantly easier and less error prone than secure deployment of TLS 1.2. When designing a new secure transport protocol such as QUIC, there is no reason to support TLS 1.2. By contrast, new application protocols that reuse TLS need to support both TLS 1.3 and TLS 1.2 in order to take advantage of underlying library or operating system support for both versions.

This BCP applies to TLS 1.3, TLS 1.2, and earlier versions. It is not safe for readers to assume that the recommendations in this BCP apply to any future version of TLS.

3.1.2. DTLS Protocol Versions

DTLS, an adaptation of TLS for UDP datagrams, was introduced when TLS 1.1 was published. The following are the recommendations with respect to DTLS:

- Implementations **MUST NOT** negotiate DTLS version 1.0 [RFC4347].

  Version 1.0 of DTLS correlates to version 1.1 of TLS (see above).
- Implementations **MUST** support DTLS 1.2 [RFC6347].

  Version 1.2 of DTLS correlates to version 1.2 of TLS (see above). (There is no version 1.1 of DTLS.)
- Implementations **SHOULD** support DTLS 1.3 [RFC9147] and, if implemented, **MUST** prefer to negotiate DTLS version 1.3 over earlier versions of DTLS.

  Version 1.3 of DTLS correlates to version 1.3 of TLS (see above).

3.1.3. Fallback to Lower Versions

TLS/DTLS 1.2 clients **MUST NOT** fall back to earlier TLS versions, since those versions have been deprecated [RFC8996]. As a result, the downgrade-protection Signaling Cipher Suite Value (SCSV) mechanism [RFC7507] is no longer needed for clients. In addition, TLS 1.3 implements a new version-negotiation mechanism.

3.2. Strict TLS

The following recommendations are provided to help prevent "SSL Stripping" and STARTTLS command injection (attacks that are summarized in [RFC7457]):

- Many existing application protocols were designed before the use of TLS became common. These protocols typically support TLS in one of two ways: either via a separate port for TLS-only communication (e.g., port 443 for HTTPS) or via a method for dynamically upgrading a channel from unencrypted to TLS protected (e.g., STARTTLS, which is used in protocols such as IMAP and XMPP). Regardless of the mechanism for protecting the communication channel

(TLS-only port or dynamic upgrade), what matters is the end state of the channel. When a protocol defines both a dynamic upgrade method and a separate TLS-only method, then the separate TLS-only method **MUST** be supported by implementations and **MUST** be configured by administrators to be used in preference to the dynamic upgrade method. When a protocol supports only a dynamic upgrade method, implementations **MUST** provide a way for administrators to set a strict local policy that forbids use of plaintext in the absence of a negotiated TLS channel, and administrators **MUST** use this policy.

- HTTP client and server implementations intended for use in the World Wide Web (see Section 5) **MUST** support the HTTP Strict Transport Security (HSTS) header field [RFC6797] so that web servers can advertise that they are willing to accept TLS-only clients. Web servers **SHOULD** use HSTS to indicate that they are willing to accept TLS-only clients, unless they are deployed in such a way that using HSTS would in fact weaken overall security (e.g., it can be problematic to use HSTS with self-signed certificates, as described in Section 11.3 of [RFC6797]). Similar technologies exist for non-HTTP application protocols, such as Mail Transfer Agent Strict Transport Security (MTA-STS) for mail transfer agents [RFC8461] and methods based on DNS-Based Authentication of Named Entities (DANE) [RFC6698] for SMTP [RFC7672] and XMPP [RFC7712].

Rationale: Combining unprotected and TLS-protected communication opens the way to SSL Stripping and similar attacks, since an initial part of the communication is not integrity protected and therefore can be manipulated by an attacker whose goal is to keep the communication in the clear.

3.3. Compression

In order to help prevent compression-related attacks (summarized in Section 2.6 of [RFC7457]) when using TLS 1.2, implementations and deployments **SHOULD NOT** support TLS-level compression (Section 6.2.2 of [RFC5246]); the only exception is when the application protocol in question has been proven not to be open to such attacks. However, even in this case, extreme caution is warranted because of the potential for future attacks related to TLS compression. More specifically, the HTTP protocol is known to be vulnerable to compression-related attacks. (This recommendation applies to TLS 1.2 only, because compression has been removed from TLS 1.3.)

Rationale: TLS compression has been subject to security attacks such as the Compression Ratio Info-leak Made Easy (CRIME) attack.

Implementers should note that compression at higher protocol levels can allow an active attacker to extract cleartext information from the connection. The Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext (BREACH) attack is one such case. These issues can only be mitigated outside of TLS and are thus outside the scope of this document. See Section 2.6 of [RFC7457] for further details.

3.3.1. Certificate Compression

Certificate chains often take up most of the bytes transmitted during the handshake. In order to manage their size, some or all of the following methods can be employed (see also Section 4 of [RFC9191] for further suggestions):

- Limit the number of names or extensions.
- Use keys with small public key representations, like the Elliptic Curve Digital Signature Algorithm (ECDSA).
- Use certificate compression.

To achieve the latter, TLS 1.3 defines the `compress_certificate` extension in [RFC8879]. See also Section 5 of [RFC8879] for security and privacy considerations associated with its use. For the avoidance of doubt, CRIME-style attacks on TLS compression do not apply to certificate compression.

Due to the strong likelihood of middlebox interference, compression in the style of [RFC8879] has not been made available in TLS 1.2. In theory, the `cached_info` extension defined in [RFC7924] could be used, but it is not supported widely enough to be considered a practical alternative.

3.4. TLS Session Resumption

Session resumption drastically reduces the number of full TLS handshakes and thus is an essential performance feature for most deployments.

Stateless session resumption with session tickets is a popular strategy. For TLS 1.2, it is specified in [RFC5077]. For TLS 1.3, a more secure mechanism based on the use of a pre-shared key (PSK) is described in Section 4.6.1 of [RFC8446]. See [Springall16] for a quantitative study of the risks induced by TLS cryptographic "shortcuts", including session resumption.

When it is used, the resumption information **MUST** be authenticated and encrypted to prevent modification or eavesdropping by an attacker. Further recommendations apply to session tickets:

- A strong cipher **MUST** be used when encrypting the ticket (at least as strong as the main TLS cipher suite).
- Ticket-encryption keys **MUST** be changed regularly, e.g., once every week, so as not to negate the benefits of forward secrecy (see Section 7.3 for details on forward secrecy). Old ticket-encryption keys **MUST** be destroyed at the end of the validity period.
- For similar reasons, session ticket validity **MUST** be limited to a reasonable duration (e.g., half as long as ticket-encryption key validity).
- TLS 1.2 does not roll the session key forward within a single session. Thus, to prevent an attack where the server's ticket-encryption key is stolen and used to decrypt the entire content of a session (negating the concept of forward secrecy), a TLS 1.2 server **SHOULD NOT** resume sessions that are too old, e.g., sessions that have been open longer than two ticket-encryption key rotation periods.

Rationale: Session resumption is another kind of TLS handshake and therefore must be as secure as the initial handshake. This document (Section 4) recommends the use of cipher suites that provide forward secrecy, i.e., that prevent an attacker who gains momentary access to the TLS endpoint (either client or server) and its secrets from reading either past or future communication. The tickets must be managed so as not to negate this security property.

TLS 1.3 provides the powerful option of forward secrecy even within a long-lived connection that is periodically resumed. Section 2.2 of [RFC8446] recommends that clients **SHOULD** send a "key_share" when initiating session resumption. In order to gain forward secrecy, this document recommends that server implementations **SHOULD** select the "psk_dhe_ke" PSK key exchange mode and respond with a "key_share" to complete an Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) exchange on each session resumption. As a more performant alternative, server implementations **MAY** refrain from responding with a "key_share" until a certain amount of time (e.g., measured in hours) has passed since the last ECDHE exchange; this implies that the "key_share" operation would not occur for the presumed majority of session resumption requests (which would occur within a few hours) while still ensuring forward secrecy for longer-lived sessions.

TLS session resumption introduces potential privacy issues where the server is able to track the client, in some cases indefinitely. See [Sy2018] for more details.

3.5. Renegotiation in TLS 1.2

The recommendations in this section apply to TLS 1.2 only, because renegotiation has been removed from TLS 1.3.

Renegotiation in TLS 1.2 is a handshake that establishes new cryptographic parameters for an existing session. The mechanism existed in TLS 1.2 and in earlier protocol versions and was improved following several major attacks including a plaintext injection attack, CVE-2009-3555 [CVE].

TLS 1.2 clients and servers **MUST** implement the `renegotiation_info` extension, as defined in [RFC5746].

TLS 1.2 clients **MUST** send `renegotiation_info` in the Client Hello. If the server does not acknowledge the extension, the client **MUST** generate a fatal `handshake_failure` alert prior to terminating the connection.

Rationale: It is not safe for a client to connect to a TLS 1.2 server that does not support `renegotiation_info` regardless of whether either endpoint actually implements renegotiation. See also Section 4.1 of [RFC5746].

A related attack resulting from TLS session parameters not being properly authenticated is a Triple Handshake [Triple-Handshake]. To address this attack, TLS 1.2 implementations **MUST** support the `extended_master_secret` extension defined in [RFC7627].

3.6. Post-Handshake Authentication

Renegotiation in TLS 1.2 was (partially) replaced in TLS 1.3 by separate post-handshake authentication and key update mechanisms. In the context of protocols that multiplex requests over a single connection (such as HTTP/2 [RFC9113]), post-handshake authentication has the same problems as TLS 1.2 renegotiation. Multiplexed protocols **SHOULD** follow the advice provided for HTTP/2 in Section 9.2.3 of [RFC9113].

3.7. Server Name Indication (SNI)

TLS implementations **MUST** support the Server Name Indication (SNI) extension defined in Section 3 of [RFC6066] for those higher-level protocols that would benefit from it, including HTTPS. However, the actual use of SNI in particular circumstances is a matter of local policy. At the time of writing, a technology for encrypting the SNI (called Encrypted Client Hello) is being worked on in the TLS Working Group [TLS-ECH]. Once that method has been standardized and widely implemented, it will likely be appropriate to recommend its usage in a future version of this BCP.

Rationale: SNI supports deployment of multiple TLS-protected virtual servers on a single address, and therefore enables fine-grained security for these virtual servers, by allowing each one to have its own certificate. However, SNI also leaks the target domain for a given connection; this information leak will be closed by use of TLS Encrypted Client Hello once that method has been standardized.

In order to prevent the attacks described in [ALPACA], a server that does not recognize the presented server name **SHOULD NOT** continue the handshake and instead **SHOULD** fail with a fatal-level `unrecognized_name(112)` alert. Note that this recommendation updates Section 3 of [RFC6066], which stated:

> If the server understood the ClientHello extension but does not recognize the server name, the server **SHOULD** take one of two actions: either abort the handshake by sending a fatal-level `unrecognized_name(112)` alert or continue the handshake.

Clients **SHOULD** abort the handshake if the server acknowledges the SNI extension but presents a certificate with a different hostname than the one sent by the client.

3.8. Application-Layer Protocol Negotiation (ALPN)

TLS implementations (both client- and server-side) **MUST** support the Application-Layer Protocol Negotiation (ALPN) extension [RFC7301].

In order to prevent "cross-protocol" attacks resulting from failure to ensure that a message intended for use in one protocol cannot be mistaken for a message for use in another protocol, servers are advised to strictly enforce the behavior prescribed in Section 3.2 of [RFC7301]:

> In the event that the server supports no protocols that the client advertises, then the server **SHALL** respond with a fatal '`no_application_protocol`' alert.

Clients **SHOULD** abort the handshake if the server acknowledges the ALPN extension but does not select a protocol from the client list. Failure to do so can result in attacks such those described in [ALPACA].

Protocol developers are strongly encouraged to register an ALPN identifier for their protocols. This applies both to new protocols and to well-established protocols; however, because the latter might have a large deployed base, strict enforcement of ALPN usage may not be feasible when an ALPN identifier is registered for a well-established protocol.

3.9. Multi-Server Deployment

Deployments that involve multiple servers or services can increase the size of the attack surface for TLS. Two scenarios are of interest:

1. Deployments in which multiple services handle the same domain name via different protocols (e.g., HTTP and IMAP). In this case, an attacker might be able to direct a connecting endpoint to the service offering a different protocol and mount a cross-protocol attack. In a cross-protocol attack, the client and server believe they are using different protocols, which the attacker might exploit if messages sent in one protocol are interpreted as messages in the other protocol with undesirable effects (see [ALPACA] for more detailed information about this class of attacks). To mitigate this threat, service providers **SHOULD** deploy ALPN (see Section 3.8). In addition, to the extent possible, they **SHOULD** ensure that multiple services handling the same domain name provide equivalent levels of security that are consistent with the recommendations in this document; such measures **SHOULD** include the handling of configurations across multiple TLS servers and protections against compromise of credentials held by those servers.
2. Deployments in which multiple servers providing the same service have different TLS configurations. In this case, an attacker might be able to direct a connecting endpoint to a server with a TLS configuration that is more easily exploitable (see [DROWN] for more detailed information about this class of attacks). To mitigate this threat, service providers **SHOULD** ensure that all servers providing the same service provide equivalent levels of security that are consistent with the recommendations in this document.

3.10. Zero Round-Trip Time (0-RTT) Data in TLS 1.3

The 0-RTT early data feature is new in TLS 1.3. It provides reduced latency when TLS connections are resumed, at the potential cost of certain security properties. As a result, it requires special attention from implementers on both the server and the client side. Typically, this extends to the TLS library as well as protocol layers above it.

For HTTP over TLS, refer to [RFC8470] for guidance.

For QUIC on TLS, refer to Section 9.2 of [RFC9001].

For other protocols, generic guidance is given in Section 8 and Appendix E.5 of [RFC8446]. To paraphrase Appendix E.5, applications **MUST** avoid this feature unless an explicit specification exists for the application protocol in question to clarify when 0-RTT is appropriate and secure. This can take the form of an IETF RFC, a non-IETF standard, or documentation associated with a non-standard protocol.

4. Recommendations: Cipher Suites

TLS 1.2 provided considerable flexibility in the selection of cipher suites. Unfortunately, the security of some of these cipher suites has degraded over time to the point where some are known to be insecure (this is one reason why TLS 1.3 restricted such flexibility). Incorrectly configuring a server leads to no or reduced security. This section includes recommendations on the selection and negotiation of cipher suites.

4.1. General Guidelines

Cryptographic algorithms weaken over time as cryptanalysis improves: algorithms that were once considered strong become weak. Consequently, cipher suites using weak algorithms need to be phased out and replaced with more secure cipher suites. This helps to ensure that the desired security properties still hold. SSL/TLS has been in existence for well over 20 years and many of the cipher suites that have been recommended in various versions of SSL/TLS are now considered weak or at least not as strong as desired. Therefore, this section modernizes the recommendations concerning cipher suite selection.

- Implementations **MUST NOT** negotiate the cipher suites with NULL encryption.

  Rationale: The NULL cipher suites do not encrypt traffic and so provide no confidentiality services. Any entity in the network with access to the connection can view the plaintext of contents being exchanged by the client and server. Nevertheless, this document does not discourage software from implementing NULL cipher suites, since they can be useful for testing and debugging.
- Implementations **MUST NOT** negotiate RC4 cipher suites.

  Rationale: The RC4 stream cipher has a variety of cryptographic weaknesses, as documented in [RFC7465]. Note that DTLS specifically forbids the use of RC4 already.
- Implementations **MUST NOT** negotiate cipher suites offering less than 112 bits of security, including so-called "export-level" encryption (which provides 40 or 56 bits of security).

  Rationale: Based on [RFC3766], at least 112 bits of security is needed. 40-bit and 56-bit security (found in so-called "export ciphers") are considered insecure today.
- Implementations **SHOULD NOT** negotiate cipher suites that use algorithms offering less than 128 bits of security.

  Rationale: Cipher suites that offer 112 or more bits but less than 128 bits of security are not considered weak at this time; however, it is expected that their useful lifespan is short enough to justify supporting stronger cipher suites at this time. 128-bit ciphers are expected

to remain secure for at least several years and 256-bit ciphers until the next fundamental technology breakthrough. Note that, because of so-called "meet-in-the-middle" attacks [Multiple-Encryption], some legacy cipher suites (e.g., 168-bit Triple DES (3DES)) have an effective key length that is smaller than their nominal key length (112 bits in the case of 3DES). Such cipher suites should be evaluated according to their effective key length.

- Implementations **SHOULD NOT** negotiate cipher suites based on RSA key transport, a.k.a. "static RSA".

  Rationale: These cipher suites, which have assigned values starting with the string "TLS_RSA_WITH_*", have several drawbacks, especially the fact that they do not support forward secrecy.

- Implementations **SHOULD NOT** negotiate cipher suites based on non-ephemeral (static) finite-field Diffie-Hellman (DH) key agreement. Similarly, implementations **SHOULD NOT** negotiate non-ephemeral Elliptic Curve DH key agreement.

  Rationale: The former cipher suites, which have assigned values prefixed by "TLS_DH_*", have several drawbacks, especially the fact that they do not support forward secrecy. The latter ("TLS_ECDH_*") also lack forward secrecy and are subject to invalid curve attacks [Jager2015].

- Implementations **MUST** support and prefer to negotiate cipher suites offering forward secrecy. However, TLS 1.2 implementations **SHOULD NOT** negotiate cipher suites based on ephemeral finite-field Diffie-Hellman key agreement (i.e., "TLS_DHE_*" suites). This is justified by the known fragility of the construction (see [RACCOON]) and the limitation around negotiation, including using [RFC7919], which has seen very limited uptake.

  Rationale: Forward secrecy (sometimes called "perfect forward secrecy") prevents the recovery of information that was encrypted with older session keys, thus limiting how far back in time data can be decrypted when an attack is successful. See Sections 7.3 and 7.4 for a detailed discussion.

4.2. Cipher Suites for TLS 1.2

Given the foregoing considerations, implementation and deployment of the following cipher suites is **RECOMMENDED**:

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

As these are Authenticated Encryption with Associated Data (AEAD) algorithms [RFC5116], these cipher suites are supported only in TLS 1.2 and not in earlier protocol versions.

Typically, to prefer these suites, the order of suites needs to be explicitly configured in server software. It would be ideal if server software implementations were to prefer these suites by default.

Some devices have hardware support for AES Counter Mode with CBC-MAC (AES-CCM) but not AES Galois/Counter Mode (AES-GCM), so they are unable to follow the foregoing recommendations regarding cipher suites. There are even devices that do not support public key cryptography at all, but these are out of scope entirely.

A cipher suite that operates in CBC (cipher block chaining) mode (e.g., TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256) **SHOULD NOT** be used unless the `encrypt_then_mac` extension [RFC7366] is also successfully negotiated. This requirement applies to both client and server implementations.

When using ECDSA signatures for authentication of TLS peers, it is **RECOMMENDED** that implementations use the NIST curve P-256. In addition, to avoid predictable or repeated nonces (which could reveal the long-term signing key), it is **RECOMMENDED** that implementations implement "deterministic ECDSA" as specified in [RFC6979] and in line with the recommendations in [RFC8446].

Note that implementations of "deterministic ECDSA" may be vulnerable to certain side-channel and fault injection attacks precisely because of their determinism. While most fault injection attacks described in the literature assume physical access to the device (and therefore are more relevant in Internet of Things (IoT) deployments with poor or non-existent physical security), some can be carried out remotely [Poddebniak2017], e.g., as Rowhammer [Kim2014] variants. In deployments where side-channel attacks and fault injection attacks are a concern, implementation strategies combining both randomness and determinism (for example, as described in [CFRG-DET-SIGS]) can be used to avoid the risk of successful extraction of the signing key.

4.2.1. Implementation Details

Clients **SHOULD** include TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as the first proposal to any server. Servers **MUST** prefer this cipher suite over weaker cipher suites whenever it is proposed, even if it is not the first proposal. Clients are of course free to offer stronger cipher suites, e.g., using AES-256; when they do, the server **SHOULD** prefer the stronger cipher suite unless there are compelling reasons (e.g., seriously degraded performance) to choose otherwise.

The previous version of the TLS recommendations [RFC7525] implicitly allowed the old RFC 5246 mandatory-to-implement cipher suite, TLS_RSA_WITH_AES_128_CBC_SHA. At the time of writing, this cipher suite does not provide additional interoperability, except with very old clients. As with other cipher suites that do not provide forward secrecy, implementations **SHOULD NOT** support this cipher suite. Other application protocols specify other cipher suites as mandatory to implement (MTI).

[RFC8422] allows clients and servers to negotiate ECDH parameters (curves). Both clients and servers **SHOULD** include the "Supported Elliptic Curves Extension" [RFC8422]. Clients and servers **SHOULD** support the NIST P-256 (secp256r1) [RFC8422] and X25519 (x25519) [RFC7748] curves. Note that [RFC8422] deprecates all but the uncompressed point format. Therefore, if the client sends an `ec_point_formats` extension, the ECPointFormatList **MUST** contain a single element, "uncompressed".

4.3. Cipher Suites for TLS 1.3

This document does not specify any cipher suites for TLS 1.3. Readers are referred to Section 9.1 of [RFC8446] for cipher suite recommendations.

4.4. Limits on Key Usage

All ciphers have an upper limit on the amount of traffic that can be securely protected with any given key. In the case of AEAD cipher suites, two separate limits are maintained for each key:

1. Confidentiality limit (CL), i.e., the number of records that can be encrypted.
2. Integrity limit (IL), i.e., the number of records that are allowed to fail authentication.

The latter applies to DTLS (and also to QUIC) but not to TLS itself, since TLS connections are torn down on the first decryption failure.

When a sender is approaching CL, the implementation **SHOULD** initiate a new handshake (in TLS 1.3, this can be achieved by sending a KeyUpdate message on the established session) to rotate the session key. When a receiver has reached IL, the implementation **SHOULD** close the connection. Although these recommendations are a best practice, implementers need to be aware that it is not always easy to accomplish them in protocols that are built on top of TLS/DTLS without introducing coordination across layer boundaries. See Section 6 of [RFC9001] for an example of the cooperation that was necessary in QUIC between the crypto and transport layers to support key updates. Note that in general, application protocols might not be able to emulate that method given their more constrained interaction with TLS/DTLS. As a result of these complexities, these recommendations are not mandatory.

For all TLS 1.3 cipher suites, readers are referred to Section 5.5 of [RFC8446] for the values of CL and IL. For all DTLS 1.3 cipher suites, readers are referred to Section 4.5.3 of [RFC9147].

For all AES-GCM cipher suites recommended for TLS 1.2 and DTLS 1.2 in this document, CL can be derived by plugging the corresponding parameters into the inequalities in Section 6.1 of [AEAD-LIMITS] that apply to random, partially implicit nonces, i.e., the nonce construction used in TLS 1.2. Although the obtained figures are slightly higher than those for TLS 1.3, it is **RECOMMENDED** that the same limit of $2^{24.5}$ records is used for both versions.

For all AES-GCM cipher suites recommended for DTLS 1.2, IL (obtained from the same inequalities referenced above) is 2^{28}.

4.5. Public Key Length

When using the cipher suites recommended in this document, two public keys are normally used in the TLS handshake: one for the Diffie-Hellman key agreement and one for server authentication. Where a client certificate is used, a third public key is added.

With a key exchange based on modular exponential (MODP) Diffie-Hellman groups ("DHE" cipher suites), DH key lengths of at least 2048 bits are **REQUIRED**.

Rationale: For various reasons, in practice, DH keys are typically generated in lengths that are powers of two (e.g., 2^{10} = 1024 bits, 2^{11} = 2048 bits, 2^{12} = 4096 bits). Because a DH key of 1228 bits would be roughly equivalent to only an 80-bit symmetric key [RFC3766], it is better to use keys longer than that for the "DHE" family of cipher suites. A DH key of 1926 bits would be roughly equivalent to a 100-bit symmetric key [RFC3766]. A DH key of 2048 bits (equivalent to a 112-bit symmetric key) is the minimum allowed by the latest revision of [NIST.SP.800-56A] as of this writing (see in particular Appendix D of that document).

As noted in [RFC3766], correcting for the emergence of The Weizmann Institute Relation Locator (TWIRL) machine [TWIRL] would imply that 1024-bit DH keys yield about 61 bits of equivalent strength and that a 2048-bit DH key would yield about 92 bits of equivalent strength. The Logjam attack [Logjam] further demonstrates that 1024-bit Diffie-Hellman parameters should be avoided.

With regard to ECDH keys, implementers are referred to the IANA "TLS Supported Groups" registry (formerly known as the "EC Named Curve Registry") within the "Transport Layer Security (TLS) Parameters" registry [IANA_TLS] and in particular to the "recommended" groups. Curves of less than 224 bits **MUST NOT** be used. This recommendation is in line with the latest revision of [NIST.SP.800-56A].

When using RSA, servers **MUST** authenticate using certificates with at least a 2048-bit modulus for the public key. In addition, the use of the SHA-256 hash algorithm is **RECOMMENDED** and SHA-1 or MD5 **MUST NOT** be used [RFC9155] (for more details, see also [CAB-Baseline], for which the current version at the time of writing is 1.8.4). Clients **MUST** indicate to servers that they request SHA-256 by using the "Signature Algorithms" extension defined in TLS 1.2. For TLS 1.3, the same requirement is already specified by [RFC8446].

4.6. Truncated HMAC

Implementations **MUST NOT** use the Truncated HMAC Extension, defined in Section 7 of [RFC6066].

Rationale: The extension does not apply to the AEAD cipher suites recommended above. However, it does apply to most other TLS cipher suites. Its use has been shown to be insecure in [PatersonRS11].

5. Applicability Statement

The recommendations of this document primarily apply to the implementation and deployment of application protocols that are most commonly used with TLS and DTLS on the Internet today. Examples include, but are not limited to:

- Web software and services that wish to protect HTTP traffic with TLS.
- Email software and services that wish to protect IMAP, Post Office Protocol version 3 (POP3), or SMTP traffic with TLS.

- Instant-messaging software and services that wish to protect Extensible Messaging and Presence Protocol (XMPP) or Internet Relay Chat (IRC) traffic with TLS.
- Realtime media software and services that wish to protect Secure Realtime Transport Protocol (SRTP) traffic with DTLS.

This document does not modify the implementation and deployment recommendations (e.g., mandatory-to-implement cipher suites) prescribed by existing application protocols that employ TLS or DTLS. If the community that uses such an application protocol wishes to modernize its usage of TLS or DTLS to be consistent with the best practices recommended here, it needs to explicitly update the existing application protocol definition (one example is [RFC7590], which updates [RFC6120]).

Designers of new application protocols developed through the Internet Standards Process [RFC2026] are expected at minimum to conform to the best practices recommended here, unless they provide documentation of compelling reasons that would prevent such conformance (e.g., widespread deployment on constrained devices that lack support for the necessary algorithms).

Although many of the recommendations provided here might also apply to QUIC insofar that it uses the TLS 1.3 handshake protocol, QUIC and other such secure transport protocols are out of scope of this document. For QUIC specifically, readers are referred to Section 9.2 of [RFC9001].

This document does not address the use of TLS in constrained-node networks [RFC7228]. For recommendations regarding the profiling of TLS and DTLS for small devices with severe constraints on power, memory, and processing resources, the reader is referred to [RFC7925] and [IOT-PROFILE].

5.1. Security Services

This document provides recommendations for an audience that wishes to secure their communication with TLS to achieve the following:

Confidentiality: all application-layer communication is encrypted with the goal that no party should be able to decrypt it except the intended receiver.

Data integrity: any changes made to the communication in transit are detectable by the receiver.

Authentication: an endpoint of the TLS communication is authenticated as the intended entity to communicate with.

With regard to authentication, TLS enables authentication of one or both endpoints in the communication. In the context of opportunistic security [RFC7435], TLS is sometimes used without authentication. As discussed in Section 5.2, considerations for opportunistic security are not in scope for this document.

If deployers deviate from the recommendations given in this document, they need to be aware that they might lose access to one of the foregoing security services.

This document applies only to environments where confidentiality is required. It requires algorithms and configuration options that enforce secrecy of the data in transit.

This document also assumes that data integrity protection is always one of the goals of a deployment. In cases where integrity is not required, it does not make sense to employ TLS in the first place. There are attacks against confidentiality-only protection that utilize the lack of integrity to also break confidentiality (see, for instance, [DegabrieleP07] in the context of IPsec).

This document addresses itself to application protocols that are most commonly used on the Internet with TLS and DTLS. Typically, all communication between TLS clients and TLS servers requires all three of the above security services. This is particularly true where TLS clients are user agents like web browsers or email clients.

This document does not address the rarer deployment scenarios where one of the above three properties is not desired, such as the use case described in Section 5.2. As another scenario where confidentiality is not needed, consider a monitored network where the authorities in charge of the respective traffic domain require full access to unencrypted (plaintext) traffic and where users collaborate and send their traffic in the clear.

5.2. Opportunistic Security

There are several important scenarios in which the use of TLS is optional, i.e., the client decides dynamically ("opportunistically") whether to use TLS with a particular server or to connect in the clear. This practice, often called "opportunistic security", is described at length in [RFC7435] and is often motivated by a desire for backward compatibility with legacy deployments.

In these scenarios, some of the recommendations in this document might be too strict, since adhering to them could cause fallback to cleartext, a worse outcome than using TLS with an outdated protocol version or cipher suite.

6. IANA Considerations

This document has no IANA actions.

7. Security Considerations

This entire document discusses the security practices directly affecting applications using the TLS protocol. This section contains broader security considerations related to technologies used in conjunction with or by TLS. The reader is referred to the Security Considerations sections of TLS 1.3 [RFC8446], DTLS 1.3 [RFC9147], TLS 1.2 [RFC5246], and DTLS 1.2 [RFC6347] for further context.

7.1. Host Name Validation

Application authors should take note that some TLS implementations do not validate host names. If the TLS implementation they are using does not validate host names, authors might need to write their own validation code or consider using a different TLS implementation.

It is noted that the requirements regarding host name validation (and, in general, binding between the TLS layer and the protocol that runs above it) vary between different protocols. For HTTPS, these requirements are defined by Sections 4.3.3, 4.3.4, and 4.3.5 of [RFC9110].

Host name validation is security-critical for all common TLS use cases. Without it, TLS ensures that the certificate is valid and guarantees possession of the private key but does not ensure that the connection terminates at the desired endpoint. Readers are referred to [RFC6125] for further details regarding generic host name validation in the TLS context. In addition, that RFC contains a long list of application protocols, some of which implement a policy very different from HTTPS.

If the host name is discovered indirectly and insecurely (e.g., by a cleartext DNS query for an SRV or Mail Exchange (MX) record), it **SHOULD NOT** be used as a reference identifier [RFC6125] even when it matches the presented certificate. This proviso does not apply if the host name is discovered securely (for further discussion, see [RFC7673] and [RFC7672]).

Host name validation typically applies only to the leaf "end entity" certificate. Naturally, in order to ensure proper authentication in the context of the PKI, application clients need to verify the entire certification path in accordance with [RFC5280].

7.2. AES-GCM

Section 4.2 recommends the use of the AES-GCM authenticated encryption algorithm. Please refer to Section 6 of [RFC5288] for security considerations that apply specifically to AES-GCM when used with TLS.

7.2.1. Nonce Reuse in TLS 1.2

The existence of deployed TLS stacks that mistakenly reuse the AES-GCM nonce is documented in [Boeck2016], showing there is an actual risk of AES-GCM getting implemented insecurely and thus making TLS sessions that use an AES-GCM cipher suite vulnerable to attacks such as [Joux2006]. (See [CVE] records: CVE-2016-0270, CVE-2016-10213, CVE-2016-10212, and CVE-2017-5933.)

While this problem has been fixed in TLS 1.3, which enforces a deterministic method to generate nonces from record sequence numbers and shared secrets for all its AEAD cipher suites (including AES-GCM), TLS 1.2 implementations could still choose their own (potentially insecure) nonce generation methods.

It is therefore **RECOMMENDED** that TLS 1.2 implementations use the 64-bit sequence number to populate the `nonce_explicit` part of the GCM nonce, as described in the first two paragraphs of Section 5.3 of [RFC8446]. This stronger recommendation updates Section 3 of [RFC5288], which specifies that the use of 64-bit sequence numbers to populate the `nonce_explicit` field is optional.

We note that at the time of writing, there are no cipher suites defined for nonce-reuse-resistant algorithms such as AES-GCM-SIV [RFC8452].

7.3. Forward Secrecy

Forward secrecy (also called "perfect forward secrecy" or "PFS" and defined in [RFC4949]) is a defense against an attacker who records encrypted conversations where the session keys are only encrypted with the communicating parties' long-term keys.

Should the attacker be able to obtain these long-term keys at some point later in time, the session keys and thus the entire conversation could be decrypted.

In the context of TLS and DTLS, such compromise of long-term keys is not entirely implausible. It can happen, for example, due to:

- A client or server being attacked by some other attack vector, and the private key retrieved.
- A long-term key retrieved from a device that has been sold or otherwise decommissioned without prior wiping.
- A long-term key used on a device as a default key [Heninger2012].
- A key generated by a trusted third party like a CA and later retrieved from it by either extortion or compromise [Soghoian2011].
- A cryptographic breakthrough or the use of asymmetric keys with insufficient length [Kleinjung2010].
- Social engineering attacks against system administrators.
- Collection of private keys from inadequately protected backups.

Forward secrecy ensures in such cases that it is not feasible for an attacker to determine the session keys even if the attacker has obtained the long-term keys some time after the conversation. It also protects against an attacker who is in possession of the long-term keys but remains passive during the conversation.

Forward secrecy is generally achieved by using the Diffie-Hellman scheme to derive session keys. The Diffie-Hellman scheme has both parties maintain private secrets and send parameters over the network as modular powers over certain cyclic groups. The properties of the so-called Discrete Logarithm Problem (DLP) allow the parties to derive the session keys without an eavesdropper being able to do so. There is currently no known attack against DLP if sufficiently large parameters are chosen. A variant of the Diffie-Hellman scheme uses elliptic curves instead of the originally proposed modular arithmetic. Given the current state of the art, Elliptic Curve Diffie-Hellman appears to be more efficient, permits shorter key lengths, and allows less freedom for implementation errors than finite-field Diffie-Hellman.

Unfortunately, many TLS/DTLS cipher suites were defined that do not feature forward secrecy, e.g., TLS_RSA_WITH_AES_256_CBC_SHA256. This document therefore advocates strict use of forward-secrecy-only ciphers.

7.4. Diffie-Hellman Exponent Reuse

For performance reasons, it is not uncommon for TLS implementations to reuse Diffie-Hellman and Elliptic Curve Diffie-Hellman exponents across multiple connections. Such reuse can result in major security issues:

- If exponents are reused for too long (in some cases, even as little as a few hours), an attacker who gains access to the host can decrypt previous connections. In other words, exponent reuse negates the effects of forward secrecy.
- TLS implementations that reuse exponents should test the DH public key they receive for group membership, in order to avoid some known attacks. These tests are not standardized in TLS at the time of writing, although general guidance in this area is provided by [NIST.SP.800-56A] and available in many protocol implementations.
- Under certain conditions, the use of static finite-field DH keys, or of ephemeral finite-field DH keys that are reused across multiple connections, can lead to timing attacks (such as those described in [RACCOON]) on the shared secrets used in Diffie-Hellman key exchange.
- An "invalid curve" attack can be mounted against Elliptic Curve DH if the victim does not verify that the received point lies on the correct curve. If the victim is reusing the DH secrets, the attacker can repeat the probe varying the points to recover the full secret (see [Antipa2003] and [Jager2015]).

To address these concerns:

- TLS implementations **SHOULD NOT** use static finite-field DH keys and **SHOULD NOT** reuse ephemeral finite-field DH keys across multiple connections.
- Server implementations that want to reuse Elliptic Curve DH keys **SHOULD** either use a "safe curve" [SAFECURVES] (e.g., X25519) or perform the checks described in [NIST.SP.800-56A] on the received points.

7.5. Certificate Revocation

The following considerations and recommendations represent the current state of the art regarding certificate revocation, even though no complete and efficient solution exists for the problem of checking the revocation status of common public key certificates [RFC5280]:

- Certificate revocation is an important tool when recovering from attacks on the TLS implementation as well as cases of misissued certificates. TLS implementations **MUST** implement a strategy to distrust revoked certificates.
- Although Certificate Revocation Lists (CRLs) are the most widely supported mechanism for distributing revocation information, they have known scaling challenges that limit their usefulness, despite workarounds such as partitioned CRLs and delta CRLs. The more modern [CRLite] and the follow-on Let's Revoke [LetsRevoke] build on the availability of Certificate Transparency [RFC9162] logs and aggressive compression to allow practical use of the CRL infrastructure, but at the time of writing, neither solution is deployed for client-side revocation processing at scale.

- Proprietary mechanisms that embed revocation lists in the web browser's configuration database cannot scale beyond the few most heavily used web servers.
- The Online Certification Status Protocol (OCSP) [RFC6960] in its basic form presents both scaling and privacy issues. In addition, clients typically "soft-fail", meaning that they do not abort the TLS connection if the OCSP server does not respond. (However, this might be a workaround to avoid denial-of-service attacks if an OCSP responder is taken offline.) For a recent survey of the status of OCSP deployment in the web PKI, see [Chung18].
- The TLS Certificate Status Request extension (Section 8 of [RFC6066]), commonly called "OCSP stapling", resolves the operational issues with OCSP. However, it is still ineffective in the presence of an active on-path attacker because the attacker can simply ignore the client's request for a stapled OCSP response.
- [RFC7633] defines a certificate extension that indicates that clients must expect stapled OCSP responses for the certificate and must abort the handshake ("hard-fail") if such a response is not available.
- OCSP stapling as used in TLS 1.2 does not extend to intermediate certificates within a certificate chain. The Multiple Certificate Status extension [RFC6961] addresses this shortcoming, but it has seen little deployment and had been deprecated by [RFC8446]. As a result, although this extension was recommended for TLS 1.2 in [RFC7525], it is no longer recommended by this document.
- TLS 1.3 (Section 4.4.2.1 of [RFC8446]) allows the association of OCSP information with intermediate certificates by using an extension to the CertificateEntry structure. However, using this facility remains impractical because many certification authorities (CAs) either do not publish OCSP for CA certificates or publish OCSP reports with a lifetime that is too long to be useful.
- Both CRLs and OCSP depend on relatively reliable connectivity to the Internet, which might not be available to certain kinds of nodes. A common example is newly provisioned devices that need to establish a secure connection in order to boot up for the first time.

For the common use cases of public key certificates in TLS, servers **SHOULD** support the following as a best practice given the current state of the art and as a foundation for a possible future solution: OCSP [RFC6960] and OCSP stapling using the `status_request` extension defined in [RFC6066]. Note that the exact mechanism for embedding the `status_request` extension differs between TLS 1.2 and 1.3. As a matter of local policy, server operators **MAY** request that CAs issue must-staple [RFC7633] certificates for the server and/or for client authentication, but we recommend reviewing the operational conditions before deciding on this approach.

The considerations in this section do not apply to scenarios where the DNS-Based Authentication of Named Entities (DANE) TLSA resource record [RFC6698] is used to signal to a client which certificate a server considers valid and good to use for TLS connections.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, DOI 10.17487/RFC3766, April 2004, <https://www.rfc-editor.org/info/rfc3766>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/info/rfc5288>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <https://www.rfc-editor.org/info/rfc5746>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6176] Turner, S. and T. Polk, "Prohibiting Secure Sockets Layer (SSL) Version 2.0", RFC 6176, DOI 10.17487/RFC6176, March 2011, <https://www.rfc-editor.org/info/rfc6176>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC7366] Gutmann, P., "Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7366, DOI 10.17487/RFC7366, September 2014, <https://www.rfc-editor.org/info/rfc7366>.

[RFC7465] Popov, A., "Prohibiting RC4 Cipher Suites", RFC 7465, DOI 10.17487/RFC7465, February 2015, <https://www.rfc-editor.org/info/rfc7465>.

[RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <https://www.rfc-editor.org/info/rfc7627>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/info/rfc8422>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8996] Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <https://www.rfc-editor.org/info/rfc8996>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[RFC9155] Velvindron, L., Moriarty, K., and A. Ghedini, "Deprecating MD5 and SHA-1 Signature Hashes in TLS 1.2 and DTLS 1.2", RFC 9155, DOI 10.17487/RFC9155, December 2021, <https://www.rfc-editor.org/info/rfc9155>.

8.2. Informative References

[AEAD-LIMITS] Günther, F., Thomson, M., and C. A. Wood, "Usage Limits on AEAD Algorithms", Work in Progress, Internet-Draft, draft-irtf-cfrg-aead-limits-05, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-aead-limits-05>.

[ALPACA] Brinkmann, M., Dresen, C., Merget, R., Poddebniak, D., Müller, J., Somorovsky, J., Schwenk, J., and S. Schinzel, "ALPACA: Application Layer Protocol Confusion - Analyzing and Mitigating Cracks in TLS Authentication", 30th USENIX Security Symposium (USENIX Security 21), August 2021, <https://www.usenix.org/conference/usenixsecurity21/presentation/brinkmann>.

[Antipa2003] Antipa, A., Brown, D. R. L., Menezes, A., Struik, R., and S. Vanstone, "Validation of Elliptic Curve Public Keys", Public Key Cryptography - PKC 2003, December 2003, <https://doi.org/10.1007/3-540-36288-6_16>.

[Boeck2016] Böck, H., Zauner, A., Devlin, S., Somorovsky, J., and P. Jovanovic, "Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS", May 2016, <https://eprint.iacr.org/2016/475.pdf>.

[CAB-Baseline] CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates", Version 1.8.4, April 2022, <https://cabforum.org/documents/>.

[CFRG-DET-SIGS] Preuß Mattsson, J., Thormarker, E., and S. Ruohomaa, "Deterministic ECDSA and EdDSA Signatures with Additional Randomness", Work in Progress, Internet-Draft, draft-irtf-cfrg-det-sigs-with-noise-00, 8 August 2022, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-det-sigs-with-noise-00>.

[Chung18] Chung, T., Lok, J., Chandrasekaran, B., Choffnes, D., Levin, D., Maggs, B., Mislove, A., Rula, J., Sullivan, N., and C. Wilson, "Is the Web Ready for OCSP Must-Staple?", Proceedings of the Internet Measurement Conference 2018, DOI 10.1145/3278532.3278543, October 2018, <https://doi.org/10.1145/3278532.3278543>.

[CRLite] Larisch, J., Choffnes, D., Levin, D., Maggs, B., Mislove, A., and C. Wilson, "CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers", 2017 IEEE Symposium on Security and Privacy (SP), DOI 10.1109/sp.2017.17, May 2017, <https://doi.org/10.1109/sp.2017.17>.

[CVE] MITRE, "Common Vulnerabilities and Exposures", <https://cve.mitre.org>.

[DegabrieleP07] Degabriele, J. and K. Paterson, "Attacking the IPsec Standards in Encryption-only Configurations", 2007 IEEE Symposium on Security and Privacy (SP '07), DOI 10.1109/sp.2007.8, May 2007, <https://doi.org/10.1109/sp.2007.8>.

[DROWN] Aviram, N., Schinzel, S., Somorovsky, J., Heninger, N., Dankel, M., Steube, J., Valenta, L., Adrian, D., Halderman, J., Dukhovni, V., Käsper, E., Cohney, S., Engels, S., Paar, C., and Y. Shavitt, "DROWN: Breaking TLS using SSLv2", 25th USENIX Security Symposium (USENIX Security 16), August 2016, <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/aviram>.

[Heninger2012] Heninger, N., Durumeric, Z., Wustrow, E., and J. A. Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices", 21st Usenix Security Symposium, August 2012.

[IANA_TLS] IANA, "Transport Layer Security (TLS) Parameters", <https://www.iana.org/assignments/tls-parameters>.

[IOT-PROFILE] Tschofenig, H. and T. Fossati, "TLS/DTLS 1.3 Profiles for the Internet of Things", Work in Progress, Internet-Draft, draft-ietf-uta-tls13-iot-profile-05, 6 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-uta-tls13-iot-profile-05>.

[Jager2015] Jager, T., Schwenk, J., and J. Somorovsky, "Practical Invalid Curve Attacks on TLS-ECDH", Computer Security -- ESORICS 2015, pp. 407-425, DOI 10.1007/978-3-319-24174-6_21, 2015, <https://doi.org/10.1007/978-3-319-24174-6_21>.

[Joux2006] Joux, A., "Authentication Failures in NIST version of GCM", 2006, <https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/comments/800-38-series-drafts/gcm/joux_comments.pdf>.

[Kim2014] Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J. H., Lee, D., Wilkerson, C., Lai, K., and O. Mutlu, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors", DOI 10.1109/ISCA.2014.6853210, July 2014, <https://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf>.

[Kleinjung2010] Kleinjung, T., Aoki, K., Franke, J., Lenstra, A., Thomé, E., Bos, J., Gaudry, P., Kruppa, A., Montgomery, P., Osvik, D., te Riele, H., Timofeev, A., and P. Zimmermann, "Factorization of a 768-Bit RSA Modulus", Advances in Cryptology - CRYPTO 2010, pp. 333-350, DOI 10.1007/978-3-642-14623-7_18, 2010, <https://doi.org/10.1007/978-3-642-14623-7_18>.

[LetsRevoke] Smith, T., Dickinson, L., and K. Seamons, "Let's Revoke: Scalable Global Certificate Revocation", Proceedings 2020 Network and Distributed System Security Symposium, DOI 10.14722/ndss.2020.24084, February 2020, <https://doi.org/10.14722/ndss.2020.24084>.

[Logjam] Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P., Green, M., Halderman, J., Heninger, N., Springall, D., Thomé, E., Valenta, L., VanderSloot, B., Wustrow, E., Zanella-Béguelin, S., and P. Zimmermann, "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice", Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 5-17, DOI 10.1145/2810103.2813707, October 2015, <https://doi.org/10.1145/2810103.2813707>.

[Multiple-Encryption] Merkle, R. and M. Hellman, "On the security of multiple encryption", Communications of the ACM, Vol. 24, Issue 7, pp. 465-467, DOI 10.1145/358699.358718, July 1981, <https://doi.org/10.1145/358699.358718>.

[NIST.SP.800-56A] National Institute of Standards and Technology, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography", Revision 3, NIST Special Publication 800-56A, DOI 10.6028/NIST.SP.800-56Ar3, April 2018, <https://doi.org/10.6028/NIST.SP.800-56Ar3>.

[PatersonRS11] Paterson, K., Ristenpart, T., and T. Shrimpton, "Tag Size Does Matter: Attacks and Proofs for the TLS Record Protocol", Proceedings of the 17th International conference on The Theory and Application of Cryptology and Information Security, pp. 372-389, DOI 10.1007/978-3-642-25385-0_20, December 2011, <https://doi.org/10.1007/978-3-642-25385-0_20>.

[Poddebniak2017] Poddebniak, D., Somorovsky, J., Schinzel, S., Lochter, M., and P. Rösler, "Attacking Deterministic Signature Schemes using Fault Attacks", Conference: 2018 IEEE European Symposium on Security and Privacy, DOI 10.1109/EuroSP.2018.00031, April 2018, <https://eprint.iacr.org/2017/1014.pdf>.

[POODLE] US-CERT, "SSL 3.0 Protocol Vulnerability and POODLE Attack", October 2014, <https://www.us-cert.gov/ncas/alerts/TA14-290A>.

[RACCOON] Merget, R., Brinkmann, M., Aviram, N., Somorovsky, J., Mittmann, J., and J. Schwenk, "Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E)", 30th USENIX Security Symposium (USENIX Security 21), 2021, <https://www.usenix.org/conference/usenixsecurity21/presentation/merget>.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, DOI 10.17487/RFC2246, January 1999, <https://www.rfc-editor.org/info/rfc2246>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, DOI 10.17487/RFC3602, September 2003, <https://www.rfc-editor.org/info/rfc3602>.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <https://www.rfc-editor.org/info/rfc4346>.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, DOI 10.17487/RFC4347, April 2006, <https://www.rfc-editor.org/info/rfc4347>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <https://www.rfc-editor.org/info/rfc5077>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/info/rfc5116>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC6101] Freier, A., Karlton, P., and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", RFC 6101, DOI 10.17487/RFC6101, August 2011, <https://www.rfc-editor.org/info/rfc6101>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <https://www.rfc-editor.org/info/rfc6797>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

[RFC6961] Pettersen, Y., "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension", RFC 6961, DOI 10.17487/RFC6961, June 2013, <https://www.rfc-editor.org/info/rfc6961>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <https://www.rfc-editor.org/info/rfc7435>.

[RFC7457] Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, DOI 10.17487/RFC7457, February 2015, <https://www.rfc-editor.org/info/rfc7457>.

[RFC7507] Moeller, B. and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks", RFC 7507, DOI 10.17487/RFC7507, April 2015, <https://www.rfc-editor.org/info/rfc7507>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC7568] Barnes, R., Thomson, M., Pironti, A., and A. Langley, "Deprecating Secure Sockets Layer Version 3.0", RFC 7568, DOI 10.17487/RFC7568, June 2015, <https://www.rfc-editor.org/info/rfc7568>.

[RFC7590] Saint-Andre, P. and T. Alkemade, "Use of Transport Layer Security (TLS) in the Extensible Messaging and Presence Protocol (XMPP)", RFC 7590, DOI 10.17487/RFC7590, June 2015, <https://www.rfc-editor.org/info/rfc7590>.

[RFC7633] Hallam-Baker, P., "X.509v3 Transport Layer Security (TLS) Feature Extension", RFC 7633, DOI 10.17487/RFC7633, October 2015, <https://www.rfc-editor.org/info/rfc7633>.

[RFC7672] Dukhovni, V. and W. Hardaker, "SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)", RFC 7672, DOI 10.17487/RFC7672, October 2015, <https://www.rfc-editor.org/info/rfc7672>.

[RFC7673] Finch, T., Miller, M., and P. Saint-Andre, "Using DNS-Based Authentication of Named Entities (DANE) TLSA Records with SRV Records", RFC 7673, DOI 10.17487/RFC7673, October 2015, <https://www.rfc-editor.org/info/rfc7673>.

[RFC7712] Saint-Andre, P., Miller, M., and P. Hancke, "Domain Name Associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP)", RFC 7712, DOI 10.17487/RFC7712, November 2015, <https://www.rfc-editor.org/info/rfc7712>.

[RFC7919] Gillmor, D., "Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)", RFC 7919, DOI 10.17487/RFC7919, August 2016, <https://www.rfc-editor.org/info/rfc7919>.

[RFC7924] Santesson, S. and H. Tschofenig, "Transport Layer Security (TLS) Cached Information Extension", RFC 7924, DOI 10.17487/RFC7924, July 2016, <https://www.rfc-editor.org/info/rfc7924>.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/info/rfc7925>.

[RFC8452] Gueron, S., Langley, A., and Y. Lindell, "AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption", RFC 8452, DOI 10.17487/RFC8452, April 2019, <https://www.rfc-editor.org/info/rfc8452>.

[RFC8461] Margolis, D., Risher, M., Ramakrishnan, B., Brotman, A., and J. Jones, "SMTP MTA Strict Transport Security (MTA-STS)", RFC 8461, DOI 10.17487/RFC8461, September 2018, <https://www.rfc-editor.org/info/rfc8461>.

[RFC8470] Thomson, M., Nottingham, M., and W. Tarreau, "Using Early Data in HTTP", RFC 8470, DOI 10.17487/RFC8470, September 2018, <https://www.rfc-editor.org/info/rfc8470>.

[RFC8879] Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", RFC 8879, DOI 10.17487/RFC8879, December 2020, <https://www.rfc-editor.org/info/rfc8879>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

[RFC9051] Melnikov, A., Ed. and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, <https://www.rfc-editor.org/info/rfc9051>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/info/rfc9112>.

[RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/info/rfc9113>.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <https://www.rfc-editor.org/info/rfc9162>.

[RFC9191] Sethi, M., Preuß Mattsson, J., and S. Turner, "Handling Large Certificates and Long Certificate Chains in TLS-Based EAP Methods", RFC 9191, DOI 10.17487/RFC9191, February 2022, <https://www.rfc-editor.org/info/rfc9191>.

[SAFECURVES] Bernstein, D. J. and T. Lange, "SafeCurves: choosing safe curves for elliptic-curve cryptography", December 2014, <https://safecurves.cr.yp.to>.

[Soghoian2011] Soghoian, C. and S. Stamm, "Certified Lies: Detecting and Defeating Government Interception Attacks Against SSL", SSRN Electronic Journal, DOI 10.2139/ssrn.1591033, April 2010, <https://doi.org/10.2139/ssrn.1591033>.

[Springall16] Springall, D., Durumeric, Z., and J. Halderman, "Measuring the Security Harm of TLS Crypto Shortcuts", Proceedings of the 2016 Internet Measurement Conference, pp. 33-47, DOI 10.1145/2987443.2987480, November 2016, <https://doi.org/10.1145/2987443.2987480>.

[STD53] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

<https://www.rfc-editor.org/info/std53>

[Sy2018] Sy, E., Burkert, C., Federrath, H., and M. Fischer, "Tracking Users across the Web via TLS Session Resumption", Proceedings of the 34th Annual Computer Security Applications Conference, pp. 289-299, DOI 10.1145/3274694.3274708, December 2018, <https://doi.org/10.1145/3274694.3274708>.

[TLS-ECH] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-15, 3 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-15>.

[Triple-Handshake] Bhargavan, K., Lavaud, A., Fournet, C., Pironti, A., and P. Strub, "Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS", 2014 IEEE Symposium on Security and Privacy, DOI 10.1109/sp.2014.14, May 2014, <https://doi.org/10.1109/sp.2014.14>.

[TWIRL] Shamir, A. and E. Tromer, "Factoring Large Numbers with the TWIRL Device", 2014 IEEE Symposium on Security and Privacy, DOI 10.1007/978-3-540-45146-4_1, 2004, <https://cs.tau.ac.il/~tromer/papers/twirl.pdf>.

Appendix A. Differences from RFC 7525

This revision of the Best Current Practices contains numerous changes, and this section is focused on the normative changes.

- High-level differences:
  - Described the expectations from new TLS-incorporating transport protocols and from new application protocols layered on TLS.
  - Clarified items (e.g., renegotiation) that only apply to TLS 1.2.
  - Changed the status of TLS 1.0 and 1.1 from "**SHOULD NOT**" to "**MUST NOT**".
  - Added TLS 1.3 at a "**SHOULD**" level.
  - Made similar changes to DTLS.
  - Included specific guidance for multiplexed protocols.
  - **MUST**-level implementation requirement for ALPN and more specific **SHOULD**-level guidance for ALPN and SNI.
  - Clarified discussion of strict TLS policies, including **MUST**-level recommendations.
  - Limits on key usage.
  - New attacks since [RFC7457]: ALPACA, Raccoon, Logjam, and "Nonce-Disrespecting Adversaries".
  - RFC 6961 (OCSP status_request_v2) has been deprecated.
  - **MUST**-level requirement for server-side RSA certificates to have a 2048-bit modulus at a minimum, replacing a "**SHOULD**".
- Differences specific to TLS 1.2:
  - **SHOULD**-level guidance on AES-GCM nonce generation.
  - **SHOULD NOT** use (static or ephemeral) finite-field DH key agreement.

- **SHOULD NOT** reuse ephemeral finite-field DH keys across multiple connections.
- **SHOULD NOT** use static Elliptic Curve DH key exchange.
- 2048-bit DH is now a "**MUST**" and ECDH minimal curve size is 224 (vs. 192 previously).
- Support for `extended_master_secret` is now a "**MUST**" (previously it was a soft recommendation, as the RFC had not been published at the time). Also removed other, more complicated, related mitigations.
- **MUST**-level restriction on session ticket validity, replacing a "**SHOULD**".
- **SHOULD**-level restriction on the TLS session duration, depending on the rotation period of an [RFC5077] ticket key.
- Dropped TLS_DHE_RSA_WITH_AES from the recommended ciphers.
- Added TLS_ECDHE_ECDSA_WITH_AES to the recommended ciphers.
- **SHOULD NOT** use the old MTI cipher suite, TLS_RSA_WITH_AES_128_CBC_SHA.
- Recommended curve X25519 alongside NIST P-256.

• Differences specific to TLS 1.3:
- New TLS 1.3 capabilities: 0-RTT.
- Removed capabilities: renegotiation and compression.
- Added mention of TLS Encrypted Client Hello, but no recommendation for use until it is finalized.
- **SHOULD**-level requirement for forward secrecy in TLS 1.3 session resumption.
- Generic **MUST**-level guidance to avoid 0-RTT unless it is documented for the particular protocol.

Acknowledgments

Thanks to Alexey Melnikov, Alvaro Retana, Andrei Popov, Ben Kaduk, Christian Huitema, Corey Bonnell, Cullen Jennings, Daniel Kahn Gillmor, David Benjamin, Eric Rescorla, Éric Vyncke, Francesca Palombini, Hannes Tschofenig, Hubert Kario, Ilari Liusvaara, John Preuß Mattsson, John R. Levine, Julien Élie, Lars Eggert, Leif Johansson, Magnus Westerlund, Martin Duke, Martin Thomson, Mohit Sahni, Nick Sullivan, Nimrod Aviram, Paul Wouters, Peter Gutmann, Rich Salz, Robert Sayre, Robert Wilton, Roman Danyliw, Ryan Sleevi, Sean Turner, Stephen Farrell, Tim Evans, Valery Smyslov, Viktor Dukhovni, and Warren Kumari for helpful comments and discussions that have shaped this document.

The authors gratefully acknowledge the contribution of Ralph Holz, who was a coauthor of RFC 7525, the previous version of the TLS recommendations.

See RFC 7525 for additional acknowledgments specific to the previous version of the TLS recommendations.

Authors' Addresses

Yaron Sheffer
Intuit
Email: yaronf.ietf@gmail.com

Peter Saint-Andre
Independent
Email: stpeter@stpeter.im

Thomas Fossati
ARM Limited
Email: thomas.fossati@arm.com