

Network Working Group
Internet-Draft
Obsoletes: 6125 (if approved)
Intended status: Standards Track
Expires: 3 November 2022

P. Saint-Andre
J. Hodges
R. Salz
Akamai Technologies
2 May 2022

Service Names in TLS
draft-ietf-uta-rfc6125bis-05

Abstract

Many application technologies enable secure communication between two entities by means of Transport Layer Security (TLS) with Internet Public Key Infrastructure Using X.509 (PKIX) certificates. This document specifies procedures for representing and verifying the identity of application services in such interactions.

This document obsoletes RFC 6125.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Using TLS in Applications Working Group mailing list (uta@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/uta/>.

Source for this draft and an issue tracker can be found at <https://github.com/richsalz/draft-ietf-uta-rfc6125bis>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 November 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Motivation	3
1.2. Applicability	3
1.3. Overview of Recommendations	4
1.4. Scope	4
1.4.1. In Scope	4
1.4.2. Out of Scope	5
1.5. Terminology	6
2. Naming of Application Services	8
3. Designing Application Protocols	10
4. Representing Server Identity	10
4.1. Rules	10
4.2. Examples	11
5. Requesting Server Certificates	12
6. Verifying Service Identity	12
6.1. Constructing a List of Reference Identifiers	13
6.1.1. Rules	13
6.1.2. Examples	14
6.2. Preparing to Seek a Match	15
6.3. Matching the DNS Domain Name Portion	16
6.4. Matching the Application Service Type Portion	17
6.5. Outcome	18
7. Security Considerations	18
7.1. Wildcard Certificates	18
7.2. Internationalized Domain Names	19
7.3. Multiple Presented Identifiers	19
7.4. Multiple Reference Identifiers	19
8. IANA Considerations	20
9. References	20
9.1. Normative References	20

9.2. Informative References	21
Appendix A. Changes from RFC 6125	24
Acknowledgements	24
Authors' Addresses	25

1. Introduction

1.1. Motivation

The visible face of the Internet largely consists of services that employ a client-server architecture in which a client communicates with an application service. When a client communicates with an application service using [TLS], [DTLS], or a protocol built on those, it has some notion of the server's identity (e.g., "the website at example.com") while attempting to establish secure communication. Likewise, during TLS negotiation, the server presents its notion of the service's identity in the form of a public-key certificate that was issued by a certificate authority (CA) in the context of the Internet Public Key Infrastructure using X.509 [PKIX]. Informally, we can think of these identities as the client's "reference identity" and the server's "presented identity"; more formal definitions are given later. A client needs to verify that the server's presented identity matches its reference identity so it can deterministically and automatically authenticate the communication.

This document defines procedures for how clients do this verification. It therefore also defines requirements on other parties, such as the certificate authorities that issue certificates, the service administrators requesting them, and the protocol designers defining how things are named.

This document obsoletes RFC 6125. Changes from RFC 6125 are described under Appendix A.

* Additional text on multiple identifiers, and their security considerations, has been added.

1.2. Applicability

This document does not supersede the rules for certificate issuance or validation specified by [PKIX]. That document also governs any certificate-related topic on which this document is silent. This includes certificate syntax, extensions such as name constraints or extended key usage, and handling of certification paths.

This document addresses only name forms in the leaf "end entity" server certificate. It does not address the name forms in the chain of certificates used to validate a certificate, let alone creating or checking the validity of such a chain. In order to ensure proper authentication, applications need to verify the entire certification path as per [PKIX].

1.3. Overview of Recommendations

The previous version of this specification, [VERIFY], surveyed the then-current practice from many IETF standards and tried to generalize best practices (see Appendix A [VERIFY] for details). This document takes the lessons learned since then and codifies them. The rules are brief:

- * Only check DNS domain names via the subjectAlternativeName extension designed for that purpose: dNSName.
- * Allow use of even more specific subjectAlternativeName extensions where appropriate such as uniformResourceIdentifier and the otherName form SRVName.
- * Wildcard support is now the default. Constrain wildcard certificates so that the wildcard can only be the complete left-most component of a domain name.
- * Do not include or check strings that look like domain names in the subject's Common Name.

1.4. Scope

1.4.1. In Scope

This document applies only to service identities that meet these three characteristics: associated with fully-qualified domain names (FQDNs), used with TLS and DTLS, and are PKIX-based.

TLS uses the words client and server, where the client is the entity that initiates the connection. In many cases, this is consistent with common practice, such as a browser connecting to a Web origin. For the sake of clarity, and to follow the usage in [TLS] and related specifications, we will continue to use the terms client and server in this document. However, these are TLS-layer roles, and the application protocol could support the TLS server making requests to the TLS client after the TLS handshake; there is no requirement that the roles at the application layer match the TLS layer.

At the time of this writing, other protocols such as [QUIC] and Network Time Security ([NTS]) use DTLS or TLS to do the initial establishment of cryptographic key material. The rules specified here apply to such services, as well.

1.4.2. Out of Scope

The following topics are out of scope for this specification:

- * Security protocols other than [TLS] or [DTLS] except as described above.
- * Keys or certificates employed outside the context of PKIX-based systems.
- * Client or end-user identities. Certificates representing client identities other than as described above, such as rfc822Name, are beyond the scope of this document.
- * Identifiers other than FQDNs. Identifiers such as IP address are not discussed. In addition, the focus of this document is on application service identities, not specific resources located at such services. Therefore this document discusses Uniform Resource Identifiers [URI] only as a way to communicate a DNS domain name (via the URI "host" component or its equivalent), not other aspects of a service such as a specific resource (via the URI "path" component) or parameters (via the URI "query" component).
- * Certification authority policies. This includes items such as the following:
 - How to certify or validate FQDNs and application service types (see [ACME] for some definition of this).
 - Issuance of certificates with identifiers such as IP addresses instead of or in addition to FQDNs.
 - Types or "classes" of certificates to issue and whether to apply different policies for them.
 - How to certify or validate other kinds of information that might be included in a certificate (e.g., organization name).

- * Resolution of DNS domain names. Although the process whereby a client resolves the DNS domain name of an application service can involve several steps, for our purposes we care only about the fact that the client needs to verify the identity of the entity with which it communicates as a result of the resolution process. Thus the resolution process itself is out of scope for this specification.
- * User interface issues. In general, such issues are properly the responsibility of client software developers and standards development organizations dedicated to particular application technologies (see, for example, [WSC-UI]).

1.5. Terminology

Because many concepts related to "identity" are often too vague to be actionable in application protocols, we define a set of more concrete terms for use in this specification.

application service: A service on the Internet that enables clients to connect for the purpose of retrieving or uploading information, communicating with other entities, or connecting to a broader network of services.

application service provider: An entity that hosts or deploys an application service.

application service type: A formal identifier for the application protocol used to provide a particular kind of application service at a domain. This often appears as a URI scheme [URI], DNS SRV Service [DNS-SRV], or an ALPN [ALPN] identifier.

delegated domain: A domain name or host name that is explicitly configured for communicating with the source domain, either by the human user controlling the client or by a trusted administrator. For example, a server at mail.example.net could be a delegated domain for connecting to an IMAP server hosting an email address of user@example.net.

derived domain: A domain name or host name that a client has derived from the source domain in an automated fashion (e.g., by means of a [DNS-SRV] lookup).

identifier: A particular instance of an identifier type that is either presented by a server in a certificate or referenced by a client for matching purposes.

identifier type: A formally-defined category of identifier that can

be included in a certificate and therefore that can also be used for matching purposes. For conciseness and convenience, we define the following identifier types of interest:

- * DNS-ID: a subjectAltName entry of type dNSName as defined in [PKIX].
- * SRV-ID: a subjectAltName entry of type otherName whose name form is SRVName, as defined in [SRVNAME].
- * URI-ID: a subjectAltName entry of type uniformResourceIdentifier as defined in [PKIX]. This entry MUST include both a "scheme" and a "host" component (or its equivalent) that matches the "reg-name" rule (where the quoted terms represent the associated [ABNF] productions from [URI]). If the entry does not have both, it is not a valid URI-ID and MUST be ignored.

PKIX: The short name for the Internet Public Key Infrastructure using X.509 defined in [PKIX]. That document provides a profile of the X.509v3 certificate specifications and X.509v2 certificate revocation list (CRL) specifications for use in the Internet.

presented identifier: An identifier presented by a server to a client within a PKIX certificate when the client attempts to establish secure communication with the server. The certificate can include one or more presented identifiers of different types, and if the server hosts more than one domain then the certificate might present distinct identifiers for each domain.

reference identifier: An identifier used by the client when examining presented identifiers. It is constructed from the source domain, and optionally an application service type.

Relative Distinguished Name (RDN): An ASN.1-based construction which itself is a building-block component of Distinguished Names. See [LDAP-DN], Section 2.

source domain: The FQDN that a client expects an application service to present in the certificate. This is typically input by a human user, configured into a client, or provided by reference such as a URL. The combination of a source domain and, optionally, an application service type enables a client to construct one or more reference identifiers.

subjectAltName entry: An identifier placed in a subjectAltName extension.

subjectAltName extension: A standard PKIX extension enabling identifiers of various types to be bound to the certificate subject.

subjectName: The name of a PKIX certificate's subject, encoded in a certificate's subject field (see [PKIX], Section 4.1.2.6).

Security-related terms used in this document, but not defined here or in [PKIX] should be understood in the the sense defined in [SECTERMS]. Such terms include "attack", "authentication", "identity", "trust", "validate", and "verify".

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.

2. Naming of Application Services

This document assumes that the name of an application service is based on a DNS domain name (e.g., example.com) -- supplemented in some circumstances by an application service type (e.g., "the IMAP server at example.net"). The DNS name conforms to one of the following forms:

1. A "traditional domain name", i.e., a FQDN (see [DNS-CONCEPTS]) all of whose labels are "LDH labels" as described in [IDNA-DEFS]. Informally, such labels are constrained to [US-ASCII] letters, digits, and the hyphen, with the hyphen prohibited in the first character position. Additional qualifications apply (refer to the above-referenced specifications for details), but they are not relevant here.
2. An "internationalized domain name", i.e., a DNS domain name that includes at least one label containing appropriately encoded Unicode code points outside the traditional US-ASCII range. That is, it contains at least one U-label or A-label, but otherwise may contain any mixture of NR-LDH labels, A-labels, or U-labels, as described in [IDNA-DEFS] and the associated documents.

From the perspective of the application client or user, some names are `_direct_` because they are provided directly by a human user. This includes runtime input, prior configuration, or explicit acceptance of a client communication attempt. Other names are `_indirect_` because they are automatically resolved by the application based on user input, such as a target name resolved from a source name using DNS SRV or [NAPTR] records. The distinction matters most for certificate consumption, specifically verification as discussed in this document.

From the perspective of the application service, some names are `_unrestricted_` because they can be used in any type of service, such as a single certificate being used for both the HTTP and IMAP services at the host `example.com`. Other names are `_restricted_` because they can only be used for one type of service, such as a special-purpose certificate that can only be used for an IMAP service. This distinction matters most for certificate issuance.

We can categorize the three identifier types as follows:

- * A DNS-ID is direct and unrestricted.
- * An SRV-ID is typically indirect but can be direct, and is restricted.
- * A URI-ID is direct and restricted.

It is important to keep these distinctions in mind, because best practices for the deployment and use of the identifiers differ. Note that cross-protocol attacks such as [ALPACA] are possible when two different protocol services use the same certificate. This can be addressed by using restricted identifiers, or deploying services so that they do not share certificates. Protocol specifications **MUST** specify which identifiers are mandatory-to-implement and **SHOULD** provide operational guidance when necessary.

The Common Name RDN **MUST NOT** be used to identify a service. Reasons for this include:

- * It is not strongly typed and therefore suffers from ambiguities in interpretation.
- * It can appear more than once in the `subjectName`.

For similar reasons, other RDN's within the `subjectName` **MUST NOT** be used to identify a service.

3. Designing Application Protocols

This section defines how protocol designers should reference this document, which would typically be a normative reference in their specification. Its specification MAY choose to allow only one of the identifier types defined here.

If the technology does not use DNS SRV records to resolve the DNS domain names of application services then its specification MUST state that SRV-ID as defined in this document is not supported. Note that many existing application technologies use DNS SRV records to resolve the DNS domain names of application services, but do not rely on representations of those records in PKIX certificates by means of SRV-IDs as defined in [SRVNAME].

If the technology does not use URIs to identify application services, then its specification MUST state that URI-ID as defined in this document is not supported. Note that many existing application technologies use URIs to identify application services, but do not rely on representation of those URIs in PKIX certificates by means of URI-IDs.

A technology MAY disallow the use of the wildcard character in DNS names. If it does so, then the specification MUST state that wildcard certificates as defined in this document are not supported.

4. Representing Server Identity

This section provides instructions for issuers of certificates.

4.1. Rules

When a certificate authority issues a certificate based on the FQDN at which the application service provider will provide the relevant application, the following rules apply to the representation of application service identities. Note that some of these rules are cumulative and can interact in important ways that are illustrated later in this document.

1. The certificate SHOULD include a "DNS-ID" as a baseline for interoperability.
2. If the service using the certificate deploys a technology for which the relevant specification stipulates that certificates ought to include identifiers of type SRV-ID (e.g., [XMPP]), then the certificate SHOULD include an SRV-ID.

3. If the service using the certificate deploys a technology for which the relevant specification stipulates that certificates ought to include identifiers of type URI-ID (e.g., [SIP] as specified by [SIP-CERTS]), then the certificate SHOULD include a URI-ID. The scheme MUST be that of the protocol associated with the application service type and the "host" component (or its equivalent) MUST be the FQDN of the service. The application protocol specification MUST specify which URI schemes are acceptable in URI-IDs contained in PKIX certificates used for the application protocol (e.g., sip but not sips or tel for SIP as described in [SIP-SIPS]).
4. The certificate MAY contain more than one DNS-ID, SRV-ID, or URI-ID as further explained under Section 7.3.
5. The certificate MAY include other application-specific identifiers for compatibility with a deployed base. Such identifiers are out of scope for this specification.

4.2. Examples

Consider a simple website at `www.example.com`, which is not discoverable via DNS SRV lookups. Because HTTP does not specify the use of URIs in server certificates, a certificate for this service might include only a DNS-ID of `www.example.com`.

Consider an IMAP-accessible email server at the host `mail.example.net` servicing email addresses of the form `user@example.net` and discoverable via DNS SRV lookups on the application service name of `example.net`. A certificate for this service might include SRV-IDs of `_imap.example.net` and `_imaps.example.net` (see [EMAIL-SRV]) along with DNS-IDs of `example.net` and `mail.example.net`.

Consider a SIP-accessible voice-over-IP (VoIP) server at the host `voice.example.edu` servicing SIP addresses of the form `user@voice.example.edu` and identified by a URI of `<sip:voice.example.edu>`. A certificate for this service would include a URI-ID of `sip:voice.example.edu` (see [SIP-CERTS]) along with a DNS-ID of `voice.example.edu`.

Consider an XMPP-compatible instant messaging (IM) server at the host `im.example.org` servicing IM addresses of the form `user@im.example.org` and discoverable via DNS SRV lookups on the `im.example.org` domain. A certificate for this service might include SRV-IDs of `_xmpp-client.im.example.org` and `_xmpp-server.im.example.org` (see [XMPP]), a DNS-ID of `im.example.org`. For backward compatibility, it may also have an XMPP-specific `XmppAddr` of `im.example.org` (see [XMPP]).

5. Requesting Server Certificates

This section provides instructions for service providers regarding the information to include in certificate signing requests (CSRs). In general, service providers SHOULD request certificates that include all of the identifier types that are required or recommended for the application service type that will be secured using the certificate to be issued.

If the certificate will be used for only a single type of application service, the service provider SHOULD request a certificate that includes a DNS-ID and, if appropriate for the application service type, an SRV-ID or URI-ID that limits the deployment scope of the certificate to only the defined application service type.

If the certificate might be used for any type of application service, then the service provider SHOULD request a certificate that includes only a DNS-ID. Again, because of multi-protocol attacks this practice is discouraged; this can be mitigated by deploying only one service on a host.

If a service provider offers multiple application service types and wishes to limit the applicability of certificates using SRV-IDs or URI-IDs, they SHOULD request multiple certificates, rather than a single certificate containing multiple SRV-IDs or URI-IDs each identifying a different application service type. This rule does not apply to application service type "bundles" that identify distinct access methods to the same underlying application such as an email application with access methods denoted by the application service types of imap, imaps, pop3, pop3s, and submission as described in [EMAIL-SRV].

6. Verifying Service Identity

At a high level, the client verifies the application service's identity by performing the following actions:

1. The client constructs a list of acceptable reference identifiers based on the source domain and, optionally, the type of service to which the client is connecting.
2. The server provides its identifiers in the form of a PKIX certificate.

3. The client checks each of its reference identifiers against the presented identifiers for the purpose of finding a match. When checking a reference identifier against a presented identifier, the client matches the source domain of the identifiers and, optionally, their application service type.

Naturally, in addition to checking identifiers, a client should perform further checks, such as expiration and revocation, to ensure that the server is authorized to provide the requested service. Because such checking is not a matter of verifying the application service identity presented in a certificate, methods for doing so are out of scope for this document.

6.1. Constructing a List of Reference Identifiers

6.1.1. Rules

The client **MUST** construct a list of acceptable reference identifiers, and **MUST** do so independently of the identifiers presented by the service.

The inputs used by the client to construct its list of reference identifiers might be a URI that a user has typed into an interface (e.g., an HTTPS URL for a website), configured account information (e.g., the domain name of a host for retrieving email, which might be different from the DNS domain name portion of a username), a hyperlink in a web page that triggers a browser to retrieve a media object or script, or some other combination of information that can yield a source domain and an application service type.

The client might need to extract the source domain and application service type from the input(s) it has received. The extracted data **MUST** include only information that can be securely parsed out of the inputs, such as parsing the FQDN out of the "host" component or deriving the application service type from the scheme of a URI. Other possibilities include pulling the data from a delegated domain that is explicitly established via client or system configuration or resolving the data via [DNSSEC]. These considerations apply only to extraction of the source domain from the inputs. Naturally, if the inputs themselves are invalid or corrupt (e.g., a user has clicked a link provided by a malicious entity in a phishing attack), then the client might end up communicating with an unexpected application service.

For example, given an input URI of <sip:alice@example.net>, a client would derive the application service type sip from the scheme and parse the domain name example.net from the host component.

Each reference identifier in the list MUST be based on the source domain and MUST NOT be based on a derived domain such as a domain name discovered through DNS resolution of the source domain. This rule is important because only a match between the user inputs and a presented identifier enables the client to be sure that the certificate can legitimately be used to secure the client's communication with the server. This removes DNS and DNS resolution from the attack surface.

Using the combination of FQDN(s) and application service type, the client MUST construct its list of reference identifiers in accordance with the following rules:

- * The list SHOULD include a DNS-ID. A reference identifier of type DNS-ID can be directly constructed from a FQDN that is (a) contained in or securely derived from the inputs, or (b) explicitly associated with the source domain by means of user configuration.
- * If a server for the application service type is typically discovered by means of DNS SRV records, then the list SHOULD include an SRV-ID.
- * If a server for the application service type is typically associated with a URI for security purposes (i.e., a formal protocol document specifies the use of URIs in server certificates), then the list SHOULD include a URI-ID.

Which identifier types a client includes in its list of reference identifiers, and their priority, is a matter of local policy. For example, a client that is built to connect only to a particular kind of service might be configured to accept as valid only certificates that include an SRV-ID for that application service type. By contrast, a more lenient client, even if built to connect only to a particular kind of service, might include both SRV-IDs and DNS-IDs in its list of reference identifiers.

6.1.2. Examples

A web browser that is connecting via HTTPS to the website at `www.example.com` would have a single reference identifier: a DNS-ID of `www.example.com`.

A mail user agent that is connecting via IMAPS to the email service at `example.net` (resolved as `mail.example.net`) might have three reference identifiers: an SRV-ID of `_imaps.example.net` (see [EMAIL-SRV]), and DNS-IDs of `example.net` and `mail.example.net`. An email user agent that does not support [EMAIL-SRV] would probably be

explicitly configured to connect to mail.example.net, whereas an SRV-aware user agent would derive example.net from an email address of the form user@example.net but might also accept mail.example.net as the DNS domain name portion of reference identifiers for the service.

A voice-over-IP (VoIP) user agent that is connecting via SIP to the voice service at voice.example.edu might have only one reference identifier: a URI-ID of sip:voice.example.edu (see [SIP-CERTS]).

An instant messaging (IM) client that is connecting via XMPP to the IM service at im.example.org might have three reference identifiers: an SRV-ID of _xmpp-client.im.example.org (see [XMPP]), a DNS-ID of im.example.org, and an XMPP-specific XmppAddr of im.example.org (see [XMPP]).

6.2. Preparing to Seek a Match

Once the client has constructed its list of reference identifiers and has received the server's presented identifiers, the client checks its reference identifiers against the presented identifiers for the purpose of finding a match. The search fails if the client exhausts its list of reference identifiers without finding a match. The search succeeds if any presented identifier matches one of the reference identifiers, at which point the client SHOULD stop the search.

Before applying the comparison rules provided in the following sections, the client might need to split the reference identifier into its DNS domain name portion and its application service type portion, as follows:

- * A DNS-ID reference identifier MUST be used directly as the DNS domain name and there is no application service type.
- * For an SRV-ID reference identifier, the DNS domain name portion is the Name and the application service type portion is the Service. For example, an SRV-ID of _imaps.example.net has a DNS domain name portion of example.net and an application service type portion of imaps, which maps to the IMAP application protocol as explained in [EMAIL-SRV].
- * For a reference identifier of type URI-ID, the DNS domain name portion is the "reg-name" part of the "host" component and the application service type portion is the scheme, as defined above. Matching only the "reg-name" rule from [URI] limits verification to DNS domain names, thereby differentiating a URI-ID from a uniformResourceIdentifier entry that contains an IP address or a mere host name, or that does not contain a "host" component at

all. Furthermore, note that extraction of the "reg-name" might necessitate normalization of the URI (as explained in [URI]). For example, a URI-ID of sip:voice.example.edu would be split into a DNS domain name portion of voice.example.edu and an application service type of sip (associated with an application protocol of SIP as explained in [SIP-CERTS]).

A client MUST match the DNS name, and if an application service type is present it MUST also match the service type as well. These are described below.

6.3. Matching the DNS Domain Name Portion

This section describes how the client must determine if the presented DNS name matches the reference DNS name. The rules differ depending on whether the domain to be checked is a traditional domain name or an internationalized domain name, as defined in Section 2. For clients that support names containing the wildcard character "*", this section also specifies a supplemental rule for such "wildcard certificates". This section uses the description of labels and domain names in [DNS-CONCEPTS].

If the DNS domain name portion of a reference identifier is a traditional domain name, then matching of the reference identifier against the presented identifier MUST be performed by comparing the set of domain name labels using a case-insensitive ASCII comparison, as clarified by [DNS-CASE]. For example, WWW.Example.Com would be lower-cased to www.example.com for comparison purposes. Each label MUST match in order for the names to be considered to match, except as supplemented by the rule about checking of wildcard labels given below.

If the DNS domain name portion of a reference identifier is an internationalized domain name, then the client MUST convert any U-labels [IDNA-DEFS] in the domain name to A-labels before checking the domain name. In accordance with [IDNA-PROTO], A-labels MUST be compared as case-insensitive ASCII. Each label MUST match in order for the domain names to be considered to match, except as supplemented by the rule about checking of wildcard labels given below.

If the technology specification supports wildcards, then the client MUST match the reference identifier against a presented identifier whose DNS domain name portion contains the wildcard character "*" in a label provided these requirements are met:

1. There is only one wildcard character.

2. The wildcard character appears only as the complete content of the left-most label.

If the requirements are not met, the presented identifier is invalid and MUST be ignored.

A wildcard in a presented identifier can only match exactly one label in a reference identifier. Note that this is not the same as DNS wildcard matching, where the "*" label always matches at least one whole label and sometimes more. See [DNS-CONCEPTS], Section 4.3.3 and [DNS-WILDCARDS].

For information regarding the security characteristics of wildcard certificates, see Section 7.1.

6.4. Matching the Application Service Type Portion

The rules for matching the application service type depend on whether the identifier is an SRV-ID or a URI-ID.

These identifiers provide an application service type portion to be checked, but that portion is combined only with the DNS domain name portion of the SRV-ID or URI-ID itself. For example, if a client's list of reference identifiers includes an SRV-ID of _xmpp-client.im.example.org and a DNS-ID of apps.example.net, the client MUST check both the combination of an application service type of xmpp-client and a DNS domain name of im.example.org and a DNS domain name of apps.example.net. However, the client MUST NOT check the combination of an application service type of xmpp-client and a DNS domain name of apps.example.net because it does not have an SRV-ID of _xmpp-client.apps.example.net in its list of reference identifiers.

If the identifier is an SRV-ID, then the application service name MUST be matched in a case-insensitive manner, in accordance with [DNS-SRV]. Note that the _ character is prepended to the service identifier in DNS SRV records and in SRV-IDs (per [SRVNAME]), and thus does not need to be included in any comparison.

If the identifier is a URI-ID, then the scheme name portion MUST be matched in a case-insensitive manner, in accordance with [URI]. Note that the : character is a separator between the scheme name and the rest of the URI, and thus does not need to be included in any comparison.

6.5. Outcome

If the client has found a presented identifier that matches a reference identifier, then the service identity check has succeeded. In this case, the client **MUST** use the matched reference identifier as the validated identity of the application service.

If the client does not find a presented identifier matching any of the reference identifiers, then the client **MUST** proceed as described as follows.

If the client is an automated application, then it **SHOULD** terminate the communication attempt with a bad certificate error and log the error appropriately. The application **MAY** provide a configuration setting to disable this behavior, but it **MUST** enable it by default.

If the client is one that is directly controlled by a human user, then it **SHOULD** inform the user of the identity mismatch and automatically terminate the communication attempt with a bad certificate error in order to prevent users from inadvertently bypassing security protections in hostile situations. Such clients **MAY** give advanced users the option of proceeding with acceptance despite the identity mismatch. Although this behavior can be appropriate in certain specialized circumstances, it needs to be handled with extreme caution, for example by first encouraging even an advanced user to terminate the communication attempt and, if they choose to proceed anyway, by forcing the user to view the entire certification path before proceeding.

The application **MAY** also present the user with the ability to accept the presented certificate as valid for subsequent connections. Such ad-hoc "pinning" **SHOULD NOT** restrict future connections to just the pinned certificate. Local policy that statically enforces a given certificate for a given peer **SHOULD** be made available only as prior configuration, rather than a just-in-time override for a failed connection.

7. Security Considerations

7.1. Wildcard Certificates

Wildcard certificates automatically vouch for any single-label host names within their domain, but not multiple levels of domains. This can be convenient for administrators but also poses the risk of vouching for rogue or buggy hosts. See for example [Defeating-SSL] (beginning at slide 91) and [HTTPSbytes] (slides 38-40).

Protection against a wildcard that identifies a public suffix [Public-Suffix], such as *.co.uk or *.com, is beyond the scope of this document.

7.2. Internationalized Domain Names

Allowing internationalized domain names can lead to visually similar characters, also referred to as "confusables", being included within certificates. For discussion, see for example [IDNA-DEFS], Section 4.4 and [UTS-39].

7.3. Multiple Presented Identifiers

A given application service might be addressed by multiple DNS domain names for a variety of reasons, and a given deployment might service multiple domains or protocols. TLS Extensions such as TLS Server Name Indication (SNI), discussed in [TLS], Section 4.4.2.2, and Application Layer Protocol Negotiation (ALPN), discussed in [ALPN], provide a way for the application to indicate the desired identifier and protocol to the server, which it can then use to select the most appropriate certificate.

This specification allows multiple DNS-IDs, SRV-IDs, or URI-IDs in a certificate. As a result, an application service can use the same certificate for multiple hostnames, such as when a client does not support the TLS SNI extension, or for multiple protocols, such as SMTP and HTTP, on a single hostname. The use of a single certificate and its keypair in such environments can make it easier to launch cross-protocol attacks, particularly when used in inconsistent TLS configurations; see, for example, [ALPACA] and [DROWN]. Server operators SHOULD take steps to mitigate the risk of cross-protocol attacks, such as ensuring all TLS endpoints using a given certificate support exactly the same TLS version(s) and ciphersuite(s), and SHOULD use the TLS ALPN extension to ensure the correct protocol is used.

7.4. Multiple Reference Identifiers

This specification describes how a client may construct multiple acceptable reference identifiers, and may match any of those reference identifiers with the set of presented identifiers. [PKIX], Section 4.2.1.10 describes a mechanism to allow CA certificates to be constrained in the set of presented identifiers that they may include within server certificates. However, these constraints only apply to the explicitly enumerated name forms. For example, a CA that is only name constrained for DNS-IDs is not constrained for SRV-IDs and URI-IDs, unless those name forms are also explicitly included within the name constraints extension.

A client that constructs multiple reference identifiers of different types, such as both DNS-IDs and SRV-IDs, as described in Section 6.1.1, SHOULD take care to ensure that CAs issuing such certificates are appropriately constrained. This MAY take the form of local policy through agreement with the issuing CA, or MAY be enforced by the client requiring that if one form of presented identifier is constrained, such as a `dnsName` name constraint for DNS-IDs, then all other forms of acceptable reference identities are also constrained, such as requiring a `uniformResourceIndicator` name constraint for URI-IDs.

8. IANA Considerations

This document has no actions for IANA.

9. References

9.1. Normative References

[DNS-CONCEPTS]

Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/rfc/rfc1034>>.

[DNS-SRV]

Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <<https://www.rfc-editor.org/rfc/rfc2782>>.

[DNS-WILDCARDS]

Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, DOI 10.17487/RFC4592, July 2006, <<https://www.rfc-editor.org/rfc/rfc4592>>.

[IDNA-DEFS]

Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/rfc/rfc5890>>.

[IDNA-PROTO]

Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <<https://www.rfc-editor.org/rfc/rfc5891>>.

- [LDAP-DN] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, DOI 10.17487/RFC4514, June 2006, <<https://www.rfc-editor.org/rfc/rfc4514>>.
- [PKIX] 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/rfc/rfc5280>>.
- [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/rfc/rfc2119>>.
- [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/rfc/rfc8174>>.
- [SRVNAME] Santesson, S., "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name", RFC 4985, DOI 10.17487/RFC4985, August 2007, <<https://www.rfc-editor.org/rfc/rfc4985>>.
- [URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/rfc/rfc3986>>.

9.2. Informative References

- [ABNF] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/rfc/rfc5234>>.
- [ACME] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <<https://www.rfc-editor.org/rfc/rfc8555>>.
- [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", September 2021, <<https://alpaca-attack.com/ALPACA.pdf>>.

- [ALPN] 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/rfc/rfc7301>>.
- [Defeating-SSL] Marlinspike, M., "New Tricks for Defeating SSL in Practice", BlackHat DC, February 2009, <<http://www.blackhat.com/presentations/bh-dc-09/Marlinspike/BlackHat-DC-09-Marlinspike-Defeating-SSL.pdf>>.
- [DNS-CASE] Eastlake 3rd, D., "Domain Name System (DNS) Case Insensitivity Clarification", RFC 4343, DOI 10.17487/RFC4343, January 2006, <<https://www.rfc-editor.org/rfc/rfc4343>>.
- [DNSSEC] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/rfc/rfc4033>>.
- [DROWN] "The DROWN Attack", n.d., <<https://drownattack.com>>.
- [DTLS] 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/rfc/rfc6347>>.
- [EMAIL-SRV] Daboo, C., "Use of SRV Records for Locating Email Submission/Access Services", RFC 6186, DOI 10.17487/RFC6186, March 2011, <<https://www.rfc-editor.org/rfc/rfc6186>>.
- [HTTPSbytes] Sokol, J. and R. Hansen, "HTTPS Can Byte Me", BlackHat Abu Dhabi, November 2010, <<https://media.blackhat.com/bh-ad-10/Hansen/Blackhat-AD-2010-Hansen-Sokol-HTTPS-Can-Byte-Me-slides.pdf>>.
- [NAPTR] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database", RFC 3403, DOI 10.17487/RFC3403, October 2002, <<https://www.rfc-editor.org/rfc/rfc3403>>.

- [NTS] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <<https://www.rfc-editor.org/rfc/rfc8915>>.
- [Public-Suffix] "Public Suffix List", 2020, <<https://publicsuffix.org>>.
- [QUIC] 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/rfc/rfc9001>>.
- [SECTERMS] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <<https://www.rfc-editor.org/rfc/rfc4949>>.
- [SIP] 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/rfc/rfc3261>>.
- [SIP-CERTS] Gurbani, V., Lawrence, S., and A. Jeffrey, "Domain Certificates in the Session Initiation Protocol (SIP)", RFC 5922, DOI 10.17487/RFC5922, June 2010, <<https://www.rfc-editor.org/rfc/rfc5922>>.
- [SIP-SIPS] Audet, F., "The Use of the SIPS URI Scheme in the Session Initiation Protocol (SIP)", RFC 5630, DOI 10.17487/RFC5630, October 2009, <<https://www.rfc-editor.org/rfc/rfc5630>>.
- [TLS] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/rfc/rfc8446>>.
- [US-ASCII] American National Standards Institute, "Coded Character Set - 7-bit American Standard Code for Information Interchange", ANSI X3.4, 1986.
- [UTS-39] Davis, M. and M. Suignard, "Unicode Security Mechanisms", n.d., <<https://unicode.org/reports/tr39>>.

- [VERIFY] 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/rfc/rfc6125>>.
- [WSC-UI] Saldhana, A. and T. Roessler, "Web Security Context: User Interface Guidelines", August 2010, <<https://www.w3.org/TR/2010/REC-wsc-ui-20100812/>>.
- [XMPP] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <<https://www.rfc-editor.org/rfc/rfc6120>>.

Appendix A. Changes from RFC 6125

This document revises and obsoletes [VERIFY] based on the decade of experience and changes since it was published. The major changes, in no particular order, include:

- * The only legal place for a certificate wildcard name is as the complete left-most component in a domain name.
- * The server identity can only be expressed in the subjectAltNames extension; it is no longer valid to use the commonName RDN, known as CN-ID in [VERIFY].
- * Detailed discussion of pinning (configuring use of a certificate that doesn't match the criteria in this document) has been removed and replaced with two paragraphs in Section 6.5.
- * The sections detailing different target audiences and which sections to read (first) have been removed.
- * References to the X.500 directory, the survey of prior art, and the sample text in Appendix A have been removed.
- * All references have been updated to the current latest version.
- * The TLS SNI extension is no longer new, it is commonplace.

Acknowledgements

We gratefully acknowledge everyone who contributed to the previous version of this document, [VERIFY]. Thanks also to Carsten Bormann for converting the previous document to Markdown so that we could more easily use Martin Thomson's i-d-template software.

In addition to discussion on the mailing list, the following people contributed significant changes: Viktor Dukhovni, Jim Fenton, Olle Johansson, and Ryan Sleevi.

Authors' Addresses

Peter Saint-Andre
United States of America
Email: stpeter@stpeter.im

Jeff Hodges
United States of America
Email: networkeddude@gmail.com

Rich Salz
Akamai Technologies
United States of America
Email: rsalz@akamai.com

UTA Working Group
Internet-Draft
Obsoletes: 7525 (if approved)
Updates: 5288, 6066 (if approved)
Intended status: Best Current Practice
Expires: 25 September 2022

Y. Sheffer
Intuit
P. Saint-Andre
independent
T. Fossati
arm
24 March 2022

Recommendations for Secure Use of Transport Layer Security (TLS) and
Datagram Transport Layer Security (DTLS)
draft-ietf-uta-rfc7525bis-06

Abstract

Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) are widely used to protect data exchanged over application protocols such as HTTP, SMTP, IMAP, POP, SIP, and XMPP. 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 recommendations for improving the security of deployed services that use TLS and DTLS. The recommendations are applicable to the majority of use cases.

An earlier version of this document was published as RFC 7525 when the industry was in the midst of its transition 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. In addition, the document updates RFC 5288 and RFC 6066 in view of recent attacks.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Terminology	5
3. General Recommendations	5
3.1. Protocol Versions	5
3.1.1. SSL/TLS Protocol Versions	5
3.1.2. DTLS Protocol Versions	6
3.1.3. Fallback to Lower Versions	7
3.2. Strict TLS	7
3.3. Compression	8
3.4. TLS Session Resumption	8
3.5. Renegotiation in TLS 1.2	9
3.6. Post-Handshake Authentication	10
3.7. Server Name Indication	10
3.8. Application-Layer Protocol Negotiation	11
3.9. Zero Round Trip Time (0-RTT) Data in TLS 1.3	11
4. Recommendations: Cipher Suites	12
4.1. General Guidelines	12
4.2. Cipher Suites for TLS 1.2	13
4.2.1. Implementation Details	14
4.3. Cipher Suites for TLS 1.3	15
4.4. Limits on Key Usage	15
4.5. Public Key Length	16
4.6. Truncated HMAC	17
5. Applicability Statement	17
5.1. Security Services	18
5.2. Opportunistic Security	19
6. IANA Considerations	19
7. Security Considerations	19
7.1. Host Name Validation	19
7.2. AES-GCM	20
7.2.1. Nonce Reuse in TLS 1.2	20
7.3. Forward Secrecy	21

7.4. Diffie-Hellman Exponent Reuse	22
7.5. Certificate Revocation	23
8. Acknowledgments	25
9. References	25
9.1. Normative References	25
9.2. Informative References	27
Appendix A. Differences from RFC 7525	34
Appendix B. Document History	36
B.1. draft-ietf-uta-rfc7525bis-06	36
B.2. draft-ietf-uta-rfc7525bis-05	36
B.3. draft-ietf-uta-rfc7525bis-04	36
B.4. draft-ietf-uta-rfc7525bis-03	36
B.5. draft-ietf-uta-rfc7525bis-02	37
B.6. draft-ietf-uta-rfc7525bis-01	37
B.7. draft-ietf-uta-rfc7525bis-00	37
B.8. draft-sheffer-uta-rfc7525bis-00	38
B.9. draft-sheffer-uta-bcpl95bis-00	38
Authors' Addresses	38

1. Introduction

Transport Layer Security (TLS) and Datagram Transport Security Layer (DTLS) are widely used to protect data exchanged over application protocols such as HTTP, SMTP, IMAP, POP, SIP, and XMPP. Over the years leading to 2015, the industry has 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, have been attacked in the context of TLS. A companion document [RFC7457] provides detailed information about these attacks and 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 these attacks 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 and version 1.3 of DTLS was finalized [I-D.ietf-tls-dtls13]; these versions largely mitigate or resolve the described attacks.

Those who implement and deploy TLS and DTLS, in particular versions 1.2 or earlier of these protocols, need guidance on how TLS can be used securely. This document provides guidance for deployed services as well as for software implementations, assuming the implementer expects his or her code to be deployed in 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.

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. This document is being republished with this in mind, and with an explicit goal to migrate most uses of TLS 1.2 into 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 particular 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, so stronger options are always allowed (e.g., depending on differing evaluations of the importance of cryptographic strength vs. computational load).

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.

2. Terminology

A number of security-related terms in this document are used in the sense defined in [RFC4949].

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 POODLE attack [POODLE], SSLv3 is now widely recognized as fundamentally insecure. See [DEP-SSLv3] 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 CBC-based cipher suites 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.

- * Implementations MUST support TLS 1.2 [RFC5246] and MUST prefer to negotiate TLS version 1.2 over earlier versions of TLS.

Rationale: Several stronger cipher suites are available only with TLS 1.2 (published in 2008). In fact, the cipher suites recommended by this document for TLS 1.2 (Section 4.2 below) are only available in this version.

- * 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. We note that as long as TLS 1.2 is still allowed by a particular implementation, even if it defaults to TLS 1.3, implementers MUST still follow all the recommendations in this document.

- * Implementations of "greenfield" protocols or deployments, where there is no need to support legacy endpoints, SHOULD support TLS 1.3, with no negotiation of earlier versions. Similarly, we RECOMMEND that new protocol designs that embed the TLS mechanisms (such as QUIC has done [RFC9001]) include TLS 1.3.

Rationale: secure deployment of TLS 1.3 is significantly easier and less error prone than the secure deployment of TLS 1.2.

This BCP applies to TLS 1.2, 1.3 and to 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] and MUST prefer to negotiate DTLS version 1.2 over earlier versions of DTLS.
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 [I-D.ietf-tls-dtls13] 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]. We note that as a result of that, the 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 (an attack that is summarized in Section 2.1 of [RFC7457]):

- * In cases where an application protocol allows implementations or deployments a choice between strict TLS configuration and dynamic upgrade from unencrypted to TLS-protected traffic (such as STARTTLS), clients and servers SHOULD prefer strict TLS configuration.
- * Application protocols typically provide a way for the server to offer TLS during an initial protocol exchange, and sometimes also provide a way for the server to advertise support for TLS (e.g., through a flag indicating that TLS is required); unfortunately, these indications are sent before the communication channel is encrypted. A client SHOULD attempt to negotiate TLS even if these indications are not communicated by the server.
- * HTTP client and server implementations MUST support the HTTP Strict Transport Security (HSTS) header [RFC6797], in order to allow Web servers to 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]).

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 disable TLS-level compression (Section 6.2.2 of [RFC5246]), unless the application protocol in question has been shown not to be open to such attacks. Note: 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 CRIME attack.

Implementers should note that compression at higher protocol levels can allow an active attacker to extract cleartext information from the connection. The 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.

3.4. TLS Session Resumption

Session resumption drastically reduces the number of 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 PSK-based mechanism is described in Section 4.6.1 of [RFC8446]. See this post (<https://blog.filippo.io/we-need-to-talk-about-session-tickets/>) by Filippo Valsorda for a comparison of TLS 1.2 and 1.3 session resumption, and [Springall16] for a quantitative study of 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 suite MUST be used when encrypting the ticket (as least as strong as the main TLS cipher suite).
- * Ticket 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 keys MUST be destroyed shortly after a new key version is made available.
- * For similar reasons, session ticket validity SHOULD be limited to a reasonable duration (e.g., half as long as ticket key validity).
- * TLS 1.2 does not roll the session key forward within a single session. Thus, to prevent an attack where a stolen ticket key is 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 key rotation periods. Note that this implies that some server implementations might need to abort sessions after a certain duration.

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 respond with a "key_share", to complete an ECDHE exchange on each session resumption.

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.

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 properly authenticated is Triple Handshake [`triple-handshake`]. To address this attack, TLS 1.2 implementations SHOULD support the `extended_master_secret` extension defined in [RFC7627].

3.6. Post-Handshake Authentication

Renegotiation in TLS 1.2 was 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), post-handshake authentication has the same problems as TLS 1.2 renegotiation. Multiplexed protocols SHOULD follow the advice provided for HTTP/2 in [RFC8740].

3.7. Server Name Indication

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. Implementers are strongly encouraged to support TLS Encrypted Client Hello (formerly called Encrypted SNI) once [I-D.ietf-tls-esni] has been standardized.

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 plugged by use of TLS Encrypted Client Hello.

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]: "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." It is also RECOMMENDED that clients 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

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 should 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." It is also RECOMMENDED that clients 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 to new protocols, as well as well-established protocols such as SMTP.

3.9. 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 improved latency when TLS connections are resumed, at the potential cost of security. As a result, it requires special attention from implementers on both the server and the client side. Typically this extends to both the TLS library as well as protocol layers above it.

For use in HTTP-over-TLS, readers are referred to [RFC8470] for guidance.

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

For other protocols, generic guidance is given in Sec. 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 even documentation associated with a non-standard protocol.

4. Recommendations: Cipher Suites

TLS and its implementations provide 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. 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. Such algorithms need to be phased out over time 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 almost 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 provide 40 or 56 bits of security).

Rationale: Based on [RFC3766], at least 112 bits of security is needed. 40-bit and 56-bit security are considered insecure today. TLS 1.1 and 1.2 never negotiate 40-bit or 56-bit export ciphers.

- * Implementations SHOULD NOT negotiate cipher suites that use algorithms offering less than 128 bits of security.

Rationale: Cipher suites that offer between 112-bits and 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 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 key agreement.

Rationale: These cipher suites, which have assigned values prefixed by "TLS_DH_*", have several drawbacks, especially the fact that they do not support forward secrecy.

- * 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 the amount of time during which attacks can be successful. See Section 7.3 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

These cipher suites are supported only in TLS 1.2 and not in earlier protocol versions, because they are authenticated encryption (AEAD) algorithms [RFC5116].

Typically, in order to prefer these suites, the order of suites needs to be explicitly configured in server software. (See [BETTERCRYPTO] for helpful deployment guidelines, but note that its recommendations differ from the current document in some details.) It would be ideal if server software implementations were to prefer these suites by default.

Some devices have hardware support for AES-CCM but not 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 they are out of scope entirely.

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 (that would allow revealing 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].

4.2.1. Implementation Details

Clients SHOULD include TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as the first proposal to any server, unless they have prior knowledge that the server cannot respond to a TLS 1.2 client_hello message.

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 this document 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 extremely 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 Sec. 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 only applies to DTLS since TLS connections are torn down on the first decryption failure.

When a sender is approaching CL, the implementation SHOULD initiate a new handshake (or in TLS 1.3, a Key Update) to rotate the session key.

When a receiver has reached IL, the implementation SHOULD close the connection.

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 [I-D.ietf-tls-dtls13].

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 [I-D.irtf-cfrg-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).

As noted in [RFC3766], correcting for the emergence of a TWIRL machine would imply that 1024-bit DH keys yield about 65 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 "Supported Groups Registry" (former "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 SHOULD 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], and see [CAB-Baseline] for more details). Clients MUST indicate to servers that they request SHA-256, by using the "Signature Algorithms" extension defined in TLS 1.2.

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, 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).

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 recommends 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 software.

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 below. 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.

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 [I-D.ietf-httpbis-semantics].

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 example protocols, some of which implement a policy very different from HTTPS.

If the host name is discovered indirectly and in an insecure manner (e.g., by an insecure DNS query for an MX or SRV 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 [DANE-SRV] and [DANE-SMTP]).

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] (see also [RFC6125]).

7.2. AES-GCM

Section 4.2 above recommends the use of the AES-GCM authenticated encryption algorithm. Please refer to Section 11 of [RFC5246] for general security considerations when using TLS 1.2, and 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 in an insecure way 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, 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 of 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]. Note that this stronger recommendation updates Section 3 of [RFC5288]: "The nonce_explicit MAY be the 64-bit sequence number."

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 either by extortion or compromise [Soghoian2011].
- * A cryptographic break-through, or the use of asymmetric keys with insufficient length [Kleijnung2010].
- * 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, many TLS implementations 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 a small number of the most heavily used Web servers.
- * The On-Line 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 an up-to-date 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 a MITM 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, we no longer recommend this extension for TLS 1.2.
- * 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 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 to review the operational conditions before deciding on this approach.

The considerations in this section do not apply to scenarios where the 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. Acknowledgments

Thanks to Alexey Melnikov, Andrei Popov, Christian Huitema, Daniel Kahn Gillmor, David Benjamin, Eric Rescorla, Hannes Tschofenig, Hubert Kario, Ilari Liusvaara, John Mattsson, John R Levine, Julien Élie, Leif Johansson, Martin Thomson, Mohit Sahni, Nick Sullivan, Nimrod Aviram, Rich Salz, Ryan Sleevi, Sean Turner, Valery Smyslov, Viktor Dukhovni 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 this document.

See RFC 7525 for additional acknowledgments for the previous revision of this document.

9. References

9.1. Normative References

- [I-D.ietf-httpbis-semantic]
Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantic-19, 12 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-httpbis-semantic-19.txt>>.
- [I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021, <<https://www.ietf.org/archive/id/draft-ietf-tls-dtls13-43.txt>>.
- [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>>.
- [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>>.
- [RFC8740] Benjamin, D., "Using TLS 1.3 with HTTP/2", RFC 8740, DOI 10.17487/RFC8740, February 2020, <<https://www.rfc-editor.org/info/rfc8740>>.
- [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>>.
- [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>>.

9.2. Informative References

- [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) , 2021, <<https://www.usenix.org/conference/usenixsecurity21/presentation/brinkmann>>.

- [Antipa2003] Antipa, A., Brown, D.R.L., Menezes, A., Struik, R., and S.A. Vanstone, "Validation of Elliptic Curve Public Keys", Public Key Cryptography - PKC 2003 , 2003.
- [BETTERCRYPTO] bettercrypto.org, "Applied Crypto Hardening", April 2015, <<https://bettercrypto.org/>>.
- [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 1.1.6", 2013, <<https://www.cabforum.org/documents.html>>.
- [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/>>.
- [DANE-SMTP] 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>>.
- [DANE-SRV] 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>>.

[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>>.

[DEP-SSLv3]

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>>.

[Heninger2012]

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

[I-D.ietf-tls-esni]

Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-14, 13 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-tls-esni-14.txt>>.

[I-D.irtf-cfrg-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-04, 7 March 2022, <<https://www.ietf.org/archive/id/draft-irtf-cfrg-aead-limits-04.txt>>.

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

[Jager2015]

Jager, T., Schwenk, J., and J. Somorovsky, "Practical Invalid Curve Attacks on TLS-ECDH", European Symposium on Research in Computer Security (ESORICS) 2015 , 2015.

[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>.

[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, 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., Zaneella-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*, 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, pp. 465-467, DOI 10.1145/358699.358718, July 1981, <<https://doi.org/10.1145/358699.358718>>.

[NIST.SP.800-56A]

Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography", *National Institute of Standards and Technology report*, 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", *Lecture Notes in Computer Science* pp. 372-389, DOI 10.1007/978-3-642-25385-0_20, 2011, <https://doi.org/10.1007/978-3-642-25385-0_20>.

- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [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, 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, DOI 10.1145/2987443.2987480, November 2016, <<https://doi.org/10.1145/2987443.2987480>>.

[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, DOI 10.1145/3274694.3274708, December 2018, <<https://doi.org/10.1145/3274694.3274708>>.

[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>>.

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:

- Clarified items (e.g. renegotiation) that only apply to TLS 1.2.
- Changed status of TLS 1.0 and 1.1 from SHOULD NOT to MUST NOT.
- Added TLS 1.3 at a SHOULD level.
- Similar changes to DTLS, pending publication of DTLS 1.3.
- Specific guidance for multiplexed protocols.
- MUST-level implementation requirement for ALPN, and more specific SHOULD-level guidance for ALPN and SNI.

- Limits on key usage.
 - New attacks since [RFC7457]: ALPACA, Raccoon, Logjam, "Nonce-Disrespecting Adversaries".
 - RFC 6961 (OCSP status_request_v2) has been deprecated.
- * 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.
 - 2048-bit DH now a MUST, ECDH minimal curve size is 224, vs. 192 previously.
 - Support for extended_master_secret is a SHOULD. Also removed other, more complicated, related mitigations.
 - SHOULD-level restriction on the TLS session duration, depending on the rotation period of an [RFC5077] ticket key.
 - Drop TLS_DHE_RSA_WITH_AES from the recommended ciphers
 - Add TLS_ECDHE_ECDSA_WITH_AES to the recommended ciphers
 - SHOULD NOT use the old MTI cipher suite, TLS_RSA_WITH_AES_128_CBC_SHA.
 - Recommend curve X25519 alongside NIST P-256
- * Differences specific to TLS 1.3:
- New TLS 1.3 capabilities: 0-RTT.
 - Removed capabilities: renegotiation, compression.
 - Added mention of TLS Encrypted Client Hello, but no recommendation to use until it is finalized.
 - SHOULD-level requirement for forward secrecy in TLS 1.3 session resumption.

- Generic SHOULD-level guidance to avoid 0-RTT unless it is documented for the particular protocol.

Appendix B. Document History

// Note to RFC Editor: please remove before publication.

B.1. draft-ietf-uta-rfc7525bis-06

- * Addressed several I-D nits raised by the document shepherd.

B.2. draft-ietf-uta-rfc7525bis-05

- * Addressed WG Last Call comments, specifically:
 - More clarity and guidance on session resumption.
 - Clarity on TLS 1.2 renegotiation.
 - Wording on the 0-RTT feature aligned with RFC 8446.
 - SHOULD NOT guidance on static and ephemeral finite field DH cipher suites.
 - Revamped the recommended TLS 1.2 cipher suites, removing DHE and adding ECDSA. The latter due to the wide adoption of ECDSA certificates and in line with RFC 8446.
 - Recommendation to use deterministic ECDSA.
 - Finally deprecated the old TLS 1.2 MTI cipher suite.
 - Deeper discussion of ECDH public key reuse issues, and as a result, recommended support of X25519.
 - Reworded the section on certificate revocation and OCSF following a long mailing list thread.

B.3. draft-ietf-uta-rfc7525bis-04

- * No version fallback from TLS 1.2 to earlier versions, therefore no SCSV.

B.4. draft-ietf-uta-rfc7525bis-03

- * Cipher integrity and confidentiality limits.

- * Require extended_master_secret.

B.5. draft-ietf-uta-rfc7525bis-02

- * Adjusted text about ALPN support in application protocols
- * Incorporated text from draft-ietf-tls-md5-sha1-deprecate

B.6. draft-ietf-uta-rfc7525bis-01

- * Many more changes, including:
 - SHOULD-level requirement for forward secrecy in TLS 1.3 session resumption.
 - Removed TLS 1.2 capabilities: renegotiation, compression.
 - Specific guidance for multiplexed protocols.
 - MUST-level implementation requirement for ALPN, and more specific SHOULD-level guidance for ALPN and SNI.
 - Generic SHOULD-level guidance to avoid 0-RTT unless it is documented for the particular protocol.
 - SHOULD-level guidance on AES-GCM nonce generation in TLS 1.2.
 - SHOULD NOT use static DH keys or reuse ephemeral DH keys across multiple connections.
 - 2048-bit DH now a MUST, ECDH minimal curve size is 224, up from 192.

B.7. draft-ietf-uta-rfc7525bis-00

- * Renamed: WG document.
- * Started populating list of changes from RFC 7525.
- * General rewording of abstract and intro for revised version.
- * Protocol versions, fallback.
- * Reference to ECHO.

B.8. draft-sheffer-uta-rfc7525bis-00

- * Renamed, since the BCP number does not change.
- * Added an empty "Differences from RFC 7525" section.

B.9. draft-sheffer-uta-bcp195bis-00

- * Initial release, the RFC 7525 text as-is, with some minor editorial changes to the references.

Authors' Addresses

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

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

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

UTA
Internet-Draft
Updates: 7925 (if approved)
Intended status: Standards Track
Expires: 8 September 2022

H. Tschofenig
T. Fossati
Arm Limited
7 March 2022

TLS/DTLS 1.3 Profiles for the Internet of Things
draft-ietf-uta-tls13-iot-profile-04

Abstract

This document is a companion to RFC 7925 and defines TLS/DTLS 1.3 profiles for Internet of Things devices. It also updates RFC 7925 with regards to the X.509 certificate profile.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at
<https://github.com/thomas-fossati/draft-tls13-iot>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document.

Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Conventions and Terminology	3
2. Credential Types	3
3. Error Handling	4
4. Session Resumption	4
5. Compression	4
6. Perfect Forward Secrecy	5
7. Keep-Alive	5
8. Timeouts	5
9. Random Number Generation	5
10. Server Name Indication	5
11. Maximum Fragment Length Negotiation	5
12. Crypto Agility	6
13. Key Length Recommendations	6
14. 0-RTT Data	6
15. Certificate Profile	7
15.1. All Certificates	7
15.1.1. Version	7
15.1.2. Serial Number	7
15.1.3. Signature	7
15.1.4. Issuer	7
15.1.5. Validity	7
15.1.6. subjectPublicKeyInfo	8
15.2. Root CA Certificate	8
15.3. Intermediate CA Certificate	8
15.4. End Entity Certificate	8
15.4.1. Client Certificate Subject	9
16. Certificate Revocation Checks	9
17. Certificate Overhead	9
18. Ciphersuites	10
19. Open Issues	11
20. Security Considerations	11
21. Acknowledgements	11
22. IANA Considerations	11
23. References	12
23.1. Normative References	12
23.2. Informative References	13
Authors' Addresses	15

1. Introduction

This document defines a profile of DTLS 1.3 [DTLS13] and TLS 1.3 [RFC8446] that offers communication security services for IoT applications and is reasonably implementable on many constrained devices. Profile thereby means that available configuration options and protocol extensions are utilized to best support the IoT environment.

For IoT profiles using TLS/DTLS 1.2 please consult [RFC7925]. This document re-uses the communication pattern defined in [RFC7925] and makes IoT-domain specific recommendations for version 1.3 (where necessary).

TLS 1.3 has been re-designed and several previously defined extensions are not applicable to the new version of TLS/DTLS anymore. This clean-up also simplifies this document. Furthermore, many outdated ciphersuites have been omitted from the TLS/DTLS 1.3 specification.

1.1. Conventions and Terminology

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.

2. Credential Types

In accordance with the recommendations in [RFC7925], a compliant implementation MUST implement TLS_AES_128_GCM_SHA256. It SHOULD implement TLS_CHACHA20_POLY1305_SHA256.

Pre-shared key based authentication is integrated into the main TLS/DTLS 1.3 specification and has been harmonized with session resumption.

A compliant implementation supporting authentication based on certificates and raw public keys MUST support digital signatures with `ecdsa_secp256r1_sha256`. A compliant implementation MUST support the key exchange with `secp256r1` (NIST P-256) and SHOULD support key exchange with `X25519`.

A plain PSK-based TLS/DTLS client or server MUST implement the following extensions:

- * Supported Versions,

- * Cookie,
- * Server Name Indication (SNI),
- * Pre-Shared Key,
- * PSK Key Exchange Modes, and
- * Application-Layer Protocol Negotiation (ALPN).

The SNI extension is discussed in this document and the justification for implementing and using the ALPN extension can be found in [RFC7525bis].

For TLS/DTLS clients and servers implementing raw public keys and/or certificates the guidance for mandatory-to-implement extensions described in Section 9.2 of [RFC8446] MUST be followed.

3. Error Handling

TLS 1.3 simplified the Alert protocol but the underlying challenge in an embedded context remains unchanged, namely what should an IoT device do when it encounters an error situation. The classical approach used in a desktop environment where the user is prompted is often not applicable with unattended devices. Hence, it is more important for a developer to find out from which error cases a device can recover from.

4. Session Resumption

TLS 1.3 has built-in support for session resumption by utilizing PSK-based credentials established in an earlier exchange.

5. Compression

TLS 1.3 does not have support for compression of application data traffic, as offered by previous versions of TLS. Applications are therefore responsible for transmitting payloads that are either compressed or use a more efficient encoding otherwise.

With regards to the handshake itself, various strategies have been applied to reduce the size of the exchanged payloads. TLS and DTLS 1.3 use less overhead, depending on the type of key confirmations, when compared to previous versions of the protocol. Additionally, the work on Compact TLS (cTLS) [I-D.ietf-tls-ctls] has taken compression of the handshake a step further by utilizing out-of-band knowledge between the communication parties to reduce the amount of data to be transmitted at each individual handshake, among applying other techniques.

6. Perfect Forward Secrecy

TLS 1.3 allows the use of PFS with all ciphersuites since the support for it is negotiated independently.

7. Keep-Alive

The discussion in Section 10 of [RFC7925] is applicable.

8. Timeouts

The recommendation in Section 11 of [RFC7925] is applicable. In particular this document RECOMMENDED to use an initial timer value of 9 seconds with exponential back off up to no less than 60 seconds.

9. Random Number Generation

The discussion in Section 12 of [RFC7925] is applicable with one exception: the ClientHello and the ServerHello messages in TLS 1.3 do not contain `gmt_unix_time` component anymore.

10. Server Name Indication

This specification mandates the implementation of the Server Name Indication (SNI) extension. Where privacy requirements require it, the Encrypted Client Hello extension [I-D.ietf-tls-esni] prevents an on-path attacker to determine the domain name the client is trying to connect to.

Note: To avoid leaking DNS lookups from network inspection altogether further protocols are needed, including DoH [RFC8484] and DPRIVE [RFC7858] [RFC8094]. Since the Encrypted Client Hello extension requires use of Hybrid Public Key Encryption (HPKE) [I-D.irtf-cfrg-hpke] and additional protocols require further protocol exchanges and cryptographic operations, there is a certain amount of overhead associated with this privacy property.

11. Maximum Fragment Length Negotiation

The Maximum Fragment Length Negotiation (MFL) extension has been superseded by the Record Size Limit (RSL) extension [RFC8449]. Implementations in compliance with this specification MUST implement the RSL extension and SHOULD use it to indicate their RAM limitations.

12. Crypto Agility

The recommendations in Section 19 of [RFC7925] are applicable.

13. Key Length Recommendations

The recommendations in Section 20 of [RFC7925] are applicable.

14. 0-RTT Data

When clients and servers share a PSK, TLS/DTLS 1.3 allows clients to send data on the first flight ("early data"). This features reduces communication setup latency but requires application layer protocols to define its use with the 0-RTT data functionality.

For HTTP this functionality is described in [RFC8470]. This document specifies the application profile for CoAP, which follows the design of [RFC8470].

For a given request, the level of tolerance to replay risk is specific to the resource it operates upon (and therefore only known to the origin server). In general, if processing a request does not have state-changing side effects, the consequences of replay are not significant. The server can choose whether it will process early data before the TLS handshake completes.

It is RECOMMENDED that origin servers allow resources to explicitly configure whether early data is appropriate in requests.

This document specifies the Early-Data option, which indicates that the request has been conveyed in early data and that a client understands the 4.25 (Too Early) status code. The semantic follows [RFC8470].

No.	C	U	N	R	Name	Format	Length	Default	E
TBD	x				Early-Data	empty	0	(none)	x

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable,
E=Encrypt and Integrity Protect (when using OSCORE)

Figure 1: Early-Data Option

// Note that 4.25 is just the suggested CoAP response code, which has
// not been registered yet.

15. Certificate Profile

This section contains updates and clarifications to the certificate profile defined in [RFC7925]. The content of Table 1 of [RFC7925] has been split by certificate "type" in order to clarify exactly what requirements and recommendations apply to which entity in the PKI hierarchy.

15.1. All Certificates

15.1.1. Version

Certificates MUST be of type X.509 v3.

15.1.2. Serial Number

CAs SHALL generate non-sequential Certificate serial numbers greater than zero (0) containing at least 64 bits of output from a CSPRNG (cryptographically secure pseudo-random number generator).

15.1.3. Signature

The signature MUST be ecdsa-with-SHA256 or stronger [RFC5758].

15.1.4. Issuer

Contains the DN of the issuing CA.

15.1.5. Validity

No maximum validity period is mandated. Validity values are expressed in notBefore and notAfter fields, as described in Section 4.1.2.5 of [RFC5280]. In particular, values MUST be expressed in Greenwich Mean Time (Zulu) and MUST include seconds even where the number of seconds is zero.

Note that the validity period is defined as the period of time from notBefore through notAfter, inclusive. This means that a hypothetical certificate with a notBefore date of 9 June 2021 at 03:42:01 and a notAfter date of 7 September 2021 at 03:42:01 becomes valid at the beginning of the :01 second, and only becomes invalid at the :02 second, a period that is 90 days plus 1 second. So for a 90-day, notAfter must actually be 03:42:00.

In many cases it is necessary to indicate that a certificate does not expire. This is likely to be the case for manufacturer-provisioned certificates. RFC 5280 provides a simple solution to convey the fact that a certificate has no well-defined expiration date by setting the `notAfter` to the `GeneralizedTime` value of `99991231235959Z`.

Some devices might not have a reliable source of time and for those devices it is also advisable to use certificates with no expiration date and to let a device management solution manage the lifetime of all the certificates used by the device. While this approach does not utilize certificates to its widest extent, it is a solution that extends the capabilities offered by a raw public key approach.

15.1.6. `subjectPublicKeyInfo`

The `SubjectPublicKeyInfo` structure indicates the algorithm and any associated parameters for the ECC public key. This profile uses the `id-ecPublicKey` algorithm identifier for ECDSA signature keys, as defined and specified in [RFC5480].

15.2. Root CA Certificate

- * `basicConstraints` MUST be present and MUST be marked critical. The `ca` field MUST be set true. The `pathLenConstraint` field SHOULD NOT be present.
- * `keyUsage` MUST be present and MUST be marked critical. Bit position for `keyCertSign` MUST be set.
- * `extendedKeyUsage` MUST NOT be present.

15.3. Intermediate CA Certificate

- * `basicConstraints` MUST be present and MUST be marked critical. The `ca` field MUST be set true. The `pathLenConstraint` field MAY be present.
- * `keyUsage` MUST be present and MUST be marked critical. Bit position for `keyCertSign` MUST be set.
- * `extendedKeyUsage` MUST NOT be present.

15.4. End Entity Certificate

- * `extendedKeyUsage` MUST be present and contain at least one of `id-kp-serverAuth` or `id-kp-clientAuth`.
- * `keyUsage` MAY be present and contain one of `digitalSignature` or `keyAgreement`.
- * Domain names MUST NOT be encoded in the `subject commonName`, instead they MUST be encoded in a `subjectAltName` of type `DNS-ID`. Domain names MUST NOT contain wildcard (*) characters. `subjectAltName` MUST NOT contain multiple names.

15.4.1. Client Certificate Subject

The requirement in Section 4.4.2 of [RFC7925] to only use EUI-64 for client certificates is lifted.

If the EUI-64 format is used to identify the subject of a client certificate, it MUST be encoded in a subjectAltName of type DNS-ID as a string of the form HH-HH-HH-HH-HH-HH-HH where 'H' is one of the symbols '0'-'9' or 'A'-'F'.

16. Certificate Revocation Checks

The considerations in Section 4.4.3 of [RFC7925] hold.

Since the publication of RFC 7925 the need for firmware update mechanisms has been reinforced and the work on standardizing a secure and interoperable firmware update mechanism has made substantial progress, see [I-D.ietf-suit-architecture]. RFC 7925 recommends to use a software / firmware update mechanism to provision devices with new trust anchors.

The use of device management protocols for IoT devices, which often include an onboarding or bootstrapping mechanism, has also seen considerable uptake in deployed devices and these protocols, some of which are standardized, allow provision of certificates on a regular basis. This enables a deployment model where IoT device utilize end-entity certificates with shorter lifetime making certificate revocation protocols, like OCSP and CRLs, less relevant.

Hence, instead of performing certificate revocation checks on the IoT device itself this specification recommends to delegate this task to the IoT device operator and to take the necessary action to allow IoT devices to remain operational.

17. Certificate Overhead

In a public key-based key exchange, certificates and public keys are a major contributor to the size of the overall handshake. For example, in a regular TLS 1.3 handshake with minimal ECC certificates and no intermediate CA utilizing the secp256r1 curve with mutual authentication, around 40% of the entire handshake payload is consumed by the two exchanged certificates.

Hence, it is not surprising that there is a strong desire to reduce the size of certificates and certificate chains. This has lead to various standardization efforts. Here is a brief summary of what options an implementer has to reduce the bandwidth requirements of a public key-based key exchange:

- * Use elliptic curve cryptography (ECC) instead of RSA-based certificate due to the smaller certificate size.
- * Avoid deep and complex CA hierarchies to reduce the number of intermediate CA certificates that need to be transmitted.
- * Pay attention to the amount of information conveyed inside certificates.
- * Use session resumption to reduce the number of times a full handshake is needed. Use Connection IDs [DTLS-CID], when possible, to enable long-lasting connections.
- * Use the TLS cached info [RFC7924] extension to avoid sending certificates with every full handshake.
- * Use client certificate URLs [RFC6066] instead of full certificates for clients.
- * Use certificate compression as defined in [I-D.ietf-tls-certificate-compression].
- * Use alternative certificate formats, where possible, such as raw public keys [RFC7250] or CBOR-encoded certificates [I-D.ietf-cose-cbor-encoded-cert].

The use of certificate handles, as introduced in cTLS [I-D.ietf-tls-ctls], is a form of caching or compressing certificates as well.

Whether to utilize any of the above extensions or a combination of them depends on the anticipated deployment environment, the availability of code, and the constraints imposed by already deployed infrastructure (e.g., CA infrastructure, tool support).

18. Ciphersuites

Section 4.5.3 of [DTLS13] flags AES-CCM with 8-octet authentication tags (CCM_8) as unsuitable for general use with DTLS. In fact, due to its low integrity limits (i.e., a high sensitivity to forgeries), endpoints that negotiate ciphersuites based on such AEAD are susceptible to a trivial DoS. (See also Section 5.3 and 5.4 of [I-D.irtf-cfrg-aead-limits] for further discussion on this topic, as well as references to the analysis supporting these conclusions.)

Specifically, [DTLS13] warns that:

"TLS_AES_128_CCM_8_SHA256 MUST NOT be used in DTLS without additional safeguards against forgery. Implementations MUST set usage limits for AEAD_AES_128_CCM_8 based on an understanding of any additional forgery protections that are used."

Since all the ciphersuites mandated by [RFC7925] and [CoAP] are based on CCM_8, there is no stand-by ciphersuite to use for applications that want to avoid the security and availability risks associated with CCM_8 while retaining interoperability with the rest of the ecosystem.

In order to ameliorate the situation, this document RECOMMENDS that implementations support the following two ciphersuites:

- * TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- * TLS_ECDHE_ECDSA_WITH_AES_128_CCM

and offer them as their first choice. These ciphersuites provide confidentiality and integrity limits that are considered acceptable in the most general settings. For the details on the exact bounds of both ciphersuites see Section 4.5.3 of [DTLS13]. Note that the GCM-based ciphersuite offers superior interoperability with cloud services at the cost of a slight increase in the wire and peak RAM footprints.

When the GCM-based ciphersuite is used with TLS 1.2, the recommendations in Section 6.2.1 of [RFC7525bis] related to deterministic nonce generation apply. In addition, the integrity limits on key usage detailed in Section 4.4 of [RFC7525bis] also apply.

19. Open Issues

A list of open issues can be found at <https://github.com/thomas-fossati/draft-tls13-iot/issues>

20. Security Considerations

This entire document is about security.

21. Acknowledgements

We would like to thank Ben Kaduk and John Mattsson.

22. IANA Considerations

IANA is asked to add the Option defined in Figure 2 to the CoAP Option Numbers registry.

Number	Name	Reference
TBD	Early-Data	RFCThis

Figure 2: Early-Data Option

IANA is asked to add the Response Code defined in Figure 3 to the CoAP Response Code registry.

Code	Description	Reference
4.25	Too Early	RFCThis

Figure 3: Too Early Response Code

23. References

23.1. Normative References

- [DTLS13] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-tls-dtls13-43>>.
- [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/rfc/rfc2119>>.
- [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/rfc/rfc5280>>.
- [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <<https://www.rfc-editor.org/rfc/rfc5480>>.

- [RFC5758] Dang, Q., Santesson, S., Moriarty, K., Brown, D., and T. Polk, "Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA", RFC 5758, DOI 10.17487/RFC5758, January 2010, <<https://www.rfc-editor.org/rfc/rfc5758>>.
- [RFC7525bis] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", Work in Progress, Internet-Draft, draft-ietf-uta-rfc7525bis-05, 3 February 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-uta-rfc7525bis-05>>.
- [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/rfc/rfc7925>>.
- [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/rfc/rfc8174>>.
- [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/rfc/rfc8446>>.
- [RFC8449] Thomson, M., "Record Size Limit Extension for TLS", RFC 8449, DOI 10.17487/RFC8449, August 2018, <<https://www.rfc-editor.org/rfc/rfc8449>>.
- [RFC8470] Thomson, M., Nottingham, M., and W. Tareau, "Using Early Data in HTTP", RFC 8470, DOI 10.17487/RFC8470, September 2018, <<https://www.rfc-editor.org/rfc/rfc8470>>.

23.2. Informative References

- [CoAP] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/rfc/rfc7252>>.

- [DTLS-CID] Rescorla, E., Tschofenig, H., Fossati, T., and A. Kraus, "Connection Identifiers for DTLS 1.2", Work in Progress, Internet-Draft, draft-ietf-tls-dtls-connection-id-13, 22 June 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-tls-dtls-connection-id-13>>.
- [I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuheid, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-03, 10 January 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-03>>.
- [I-D.ietf-suit-architecture]
Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", Work in Progress, Internet-Draft, draft-ietf-suit-architecture-16, 27 January 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-suit-architecture-16>>.
- [I-D.ietf-tls-certificate-compression]
Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", Work in Progress, Internet-Draft, draft-ietf-tls-certificate-compression-10, 6 January 2020, <<https://datatracker.ietf.org/doc/html/draft-ietf-tls-certificate-compression-10>>.
- [I-D.ietf-tls-ctls]
Rescorla, E., Barnes, R., and H. Tschofenig, "Compact TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-ctls-04, 25 October 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-tls-ctls-04>>.
- [I-D.ietf-tls-esni]
Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-14, 13 February 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-14>>.

- [I-D.irtf-cfrg-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-03, 12 July 2021, <<https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-aead-limits-03>>.
- [I-D.irtf-cfrg-hpke]
Barnes, R. L., Bhargavan, K., Lipp, B., and C. A. Wood, "Hybrid Public Key Encryption", Work in Progress, Internet-Draft, draft-irtf-cfrg-hpke-12, 2 September 2021, <<https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-hpke-12>>.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/rfc/rfc6066>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <<https://www.rfc-editor.org/rfc/rfc7250>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/rfc/rfc7858>>.
- [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/rfc/rfc7924>>.
- [RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <<https://www.rfc-editor.org/rfc/rfc8094>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/rfc/rfc8484>>.

Authors' Addresses

Hannes Tschofenig
Arm Limited

Email: Hannes.Tschofenig@gmx.net

Thomas Fossati
Arm Limited
Email: Thomas.Fossati@arm.com