UTA
Internet-Draft
Updates: 7925 (if approved)
Intended status: Standards Track
Expires: 26 November 2026

H. Tschofenig
UniBw M.
T. Fossati
Linaro
M. Richardson
Sandelman Software Works
D. Migault
Ericsson

25 May 2026

TLS/DTLS 1.3 Profiles for the Internet of Things
draft-ietf-uta-tls13-iot-profile-21

Abstract

RFC 7925 offers guidance to developers on using TLS/DTLS 1.2 for Internet of Things (IoT) devices with resource constraints. This document is a companion to RFC 7925, defining TLS/DTLS 1.3 profiles for IoT devices. Additionally, it updates RFC 7925 with respect to the X.509 certificate profile and ciphersuite requirements.

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

Table of Contents

1. Introduction

> | Note to RFC Editor: Once RFC 9846 (RFC 8446bis) is published, all references to RFC 8446 must be updated to refer to RFC 9846. All section references must also be updated accordingly.

In the rapidly evolving Internet of Things (IoT) ecosystem, communication security is a critical requirement. The Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols have been foundational for ensuring encryption, integrity, and authenticity in communications. However, the constraints of a certain class of IoT devices render conventional, off-the-shelf TLS/DTLS implementations suboptimal for many IoT use cases. This document addresses these limitations by specifying TLS 1.3 and DTLS 1.3 profiles that are optimized for resource-constrained IoT devices.

Note that IoT devices vary widely in terms of capabilities. While some are highly resource-constrained, others offer performance comparable to regular desktop computers but operate without end-user interfaces. For a detailed description of the different classes of IoT devices, please refer to [RFC7228] and [I-D.ietf-iotops-7228bis].

It is crucial for developers to thoroughly assess the limitations of their IoT devices and communication technologies to implement the most suitable optimizations. The profiles in this document aim to balance strong security with the hardware and software limitations of IoT devices.

TLS 1.3 has been re-designed and several previously defined extensions are not applicable to the new version of TLS/DTLS anymore. The following features changed with the transition from TLS 1.2 to 1.3:

* TLS 1.3 introduced the concept of post-handshake authentication messages, which partially replaced the need for the re-negotiation feature [RFC5746] available in earlier TLS versions. However, the rekeying mechanism defined in Section 4.6.3 of [RFC8446] does not provide post-compromise security (see Appendix E.1.5 of [RFC8446]). Furthermore, post-handshake authentication defined in Section 4.6.2 of [RFC8446] only offers client authentication (client-to-server). The "Exported Authenticator" specification, see [RFC9261], added support for mutual post-handshake authentication, but this requires the Certificate, CertificateVerify and the Finished messages to be conveyed by the application layer protocol, as it is exercised for HTTP/2 and HTTP/3 in [I-D.ietf-httpbis-secondary-server-certs]. Therefore, the application layer protocol must be enhanced whenever this feature is required.
* Rekeying of the application traffic secret does not lead to an update of the exporter secret (see Section 7.5 of [RFC8446]) since the derived export secret is based on the exporter_master_secret and not on the application traffic secret.
* Flight #4, which was used by EAP-TLS 1.2 [RFC5216], does not exist in TLS 1.3. As a consequence, EAP-TLS 1.3 [RFC9190] introduced a placeholder message.
* [RFC4279] introduced PSK-based authentication to TLS, a feature re-designed in TLS 1.3. The "PSK identity hint" defined in [RFC4279], which is used by the server to help the client in selecting which PSK identity to use, is, however, not available anymore in TLS 1.3.
* Finally, ciphersuites were deprecated and the RSA-based key transport is not supported in TLS 1.3. As a consequence, only a Diffie-Hellman-based key exchange is available for non-PSK-based (i.e., certificate-based) authentication. (For PSK-based authentication the use of Diffie-Hellman is optional.)

The profiles in this specification are designed to be adaptable to the broad spectrum of IoT applications, from low-power consumer devices to large-scale industrial deployments. It provides guidelines for implementing TLS/DTLS 1.3 in diverse networking

contexts, including reliable, connection-oriented transport via TCP for TLS, and lightweight, connectionless communication via UDP for DTLS. In particular, DTLS is emphasized for scenarios where low-latency communication is paramount, such as multi-hop mesh networks and low-power wide-area networks, where the amount of data exchanged needs to be minimized.

This document offers comprehensive guidance for deploying secure communication in resource-constrained IoT environments. It outlines best practices for configuring TLS/DTLS 1.3 to meet the unique needs of IoT devices, ensuring robust security without overwhelming their limited processing, memory, and power resources. The document aims to facilitate the development of secure and efficient IoT deployments and promote the broad adoption of secure communication standards.

This document updates [RFC7925] with respect to the X.509 certificate profile (Section 17) and ciphersuite requirements (Section 20).

2. 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.

This document reuses the terms "SHOULD+", "SHOULD-" and "MUST-" from [RFC8221].

3. Credential Types

TLS/DTLS allow different credential types to be used. These include X.509 certificates and raw public keys, pre-shared keys (PSKs), and passwords. The extensions used in TLS/DTLS differ depending on the credential types supported. Self-signed X.509 certificates are still X.509, not raw public keys; raw public keys are conveyed via the raw_public_key extension.

This profile considers three authentication modes for IoT devices: (1) certificate-based, (2) raw public key-based and (3) external PSK-based. PSK with (EC)DHE is optional and not assumed by default.

TLS/DTLS 1.3 supports PSK-based authentication, wherein PSKs can be established via session tickets from prior connections or via some external, out-of-band mechanism. To distinguish the two modes, the former is called resumption PSK and the latter external PSK. For performance reasons the support for resumption PSKs is often found in implementations that use X.509 certificates for authentication.

Implementations that only support external PSKs are common in constrained devices; implementations using certificates often also support resumption PSKs for performance.

A "plain" PSK-based TLS/DTLS client or server, which only implements support for external PSKs as its long-term credential, 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).

Note that these extensions may also appear in ECDHE or resumption handshakes; the requirement here is that external PSK-only endpoints MUST support them.

For external pre-shared keys, [RFC9258] recommends that applications SHOULD provision separate PSKs for (D)TLS 1.3 and prior versions.

Where possible, the importer interface defined in [RFC9258] MUST be used for external PSKs. This ensures that external PSKs used in (D)TLS 1.3 are bound to a specific key derivation function (KDF) and hash function.

SNI is discussed in Section 12; the justification for implementing and using the ALPN extension can be found in [RFC9325].

An 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.

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. In addition, compliant implementations MUST implement the Record Size Limit (RSL) extension; see Section 13.

Entities deploying IoT devices may select credential types based on security characteristics, operational requirements, cost, and other factors. Consequently, this specification does not prescribe a single credential type but provides guidance on considerations relevant to the use of particular types.

4. 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.

5. Session Resumption

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

6. Compression

TLS 1.3 does not define 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.

7. Forward Secrecy

RFC 8446 has removed Static RSA and Static Diffie-Hellman cipher suites, therefore all public-key-based key exchange mechanisms available in TLS 1.3 provide forward secrecy.

Pre-shared keys (PSKs) can be used with (EC)DHE key exchange to provide forward secrecy or can be used alone, at the cost of losing forward secrecy for the application data. For PSK use, endpoints SHOULD use (EC)DHE to achieve forward secrecy; PSK-only SHOULD be avoided unless the application can tolerate the loss of forward secrecy.

8. Authentication and Integrity-only Cipher Suites

For a few, very specific Industrial IoT use cases [RFC9150] defines two cipher suites that provide data authenticity, but not data confidentiality. For details and use constraints, defer to [RFC9150] (especially Section 9 of [RFC9150]). Implementations may not support these suites; deployments should not assume availability. This document does not add new guidance beyond [RFC9150]. Profiling the

use of authentication- and integrity-only cipher suites is out of scope for this specification.

9. Keep-Alive

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

10. Timers and ACKs

Compared to DTLS 1.2 timeout-based whole flight retransmission, DTLS 1.3 ACKs sensibly decrease the risk of congestion collapse which was the basis for the very conservative recommendations given in Section 11 of [RFC7925].

The recommendations in Section 7.3 of [RFC9147] regarding ACKs apply. In particular,

> When DTLS 1.3 is used in deployments with lossy networks, such as low-power, long-range radio networks as well as low-power mesh networks, the use of ACKs is recommended.

ACKs provide explicit feedback on which handshake messages have been received. This enables endpoints to detect a lack of progress more quickly and to trigger selective or early retransmission, leading to more efficient use of bandwidth and memory.

Due to the vast range of network technologies used in IoT deployments, from wired LAN to GSM-SMS, it’s not possible to provide a universal recommendation for an initial timeout. Therefore, it is RECOMMENDED that DTLS 1.3 implementations allow developers to explicitly set the initial timer value. Developers SHOULD set the initial timeout to be twice the expected round-trip time (RTT), but no less than 1000ms, which is a conservative default aligned with the guidance in Section 11 of [RFC7925]. For specific application/ network combinations, a sub-second initial timeout MAY be set. In cases where no RTT estimates are available, a 1000ms initial timeout is suitable for the general Internet.

Regarding the timers used by the Return Routability Check (RRC) functionality, the recommendations in Section 5.5 of [I-D.ietf-tls-dtls-rrc] apply. Just like the handshake initial timers, it is RECOMMENDED that DTLS 1.2 and 1.3 implementations offer an option for their developers to explicitly set the RRC timer.

11. 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. For entropy generation and randomness considerations, implementers should also consult [RFC8937].

12. Server Name Indication

To support edge-to-cloud communication, this specification mandates the implementation of the Server Name Indication (SNI) extension for IoT devices acting as clients. This functionality is not strictly required for constrained-to-constrained communication and could be disabled in such cases. However, figuring out how to deactivate SNI can be difficult in some libraries. Furthermore, in a multi-vendor IoT deployment, some devices may not be aware of whether they are communicating with another device or a cloud service. Therefore, it is best to leave SNI as MTI for all clients.

Where privacy requirements necessitate it, the ECH (Encrypted Client Hello) extension [RFC9849] prevents an on-path attacker from determining the domain name the client is trying to connect to. Since the ECH extension requires the use of Hybrid Public Key Encryption (HPKE) [RFC9180], and additional protocols require further protocol exchanges and cryptographic operations, there is a certain overhead associated with this privacy feature. Note that in industrial IoT deployments, the use of ECH may be disabled because network administrators routinely inspect the SNI to detect malicious behavior. Furthermore, to avoid leaking DNS lookups to network inspection altogether, additional protocols are needed, including DNS-over-HTTPS (DoH) [RFC8484], DNS-over-TLS (DoT) [RFC7858], and DNS-over-QUIC (DoQ) [RFC9250].

Where IoT devices are accepting (D)TLS connections (i.e., they are acting as a server), it is unlikely that there will be a useful name placed into the SNI by the connecting client. Since an IoT server cannot rely on a client to provide a correct SNI, IoT devices in a responding (server) mode SHOULD ignore SNI. In the rare event that an IoT device has multiple server instances responding with different server certificates, the device SHOULD use different IP addresses or port numbers rather than relying on SNI.

13. 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.

14. Crypto Agility

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

15. Key Length Recommendations

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

16. 0-RTT Data

Appendix E.5 of [RFC8446] establishes that:

> Application protocols MUST NOT use 0-RTT data without a profile that defines its use. That profile needs to identify which messages or interactions are safe to use with 0-RTT and how to handle the situation when the server rejects 0-RTT and falls back to 1-RTT.

For any application protocol, 0-RTT MUST NOT be used unless a protocol-specific profile exists. At the time of writing, no such profile has been defined for CoAP [CoAP]. Therefore, 0-RTT MUST NOT be used by CoAP applications.

17. 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.

This profile does not define a specific certificate policy OID; deployments MAY define one if needed for local policy enforcement.

The terminology used in this section is not intended to restrict the scope of this profile to IEEE 802.1AR deployments. It is used because it conveniently distinguishes between manufacturer-provisioned and operational credentials, which is important in many IoT deployments.

A Device Identifier (DevID) consists of:

* a private key,
* a certificate containing the public key and the identifier certified by the certificate issuer, and
* a certificate chain leading up to a trust anchor (typically the root certificate).

The IEEE 802.1AR specification [IEEE-802.1AR] introduces the concept of DevIDs and defines two specialized versions:

* Initial Device Identifiers (IDevIDs): Provisioned during manufacturing to provide a unique, stable identity for the lifetime of the device.
* Locally Significant Device Identifiers (LDevIDs): Provisioned after deployment and typically used for operational purposes within a specific domain.

The IDevID is typically provisioned by a manufacturer and signed by the manufacturer CA. It is then used to obtain operational certificates, the LDevIDs, from the operator or owner of the device. Some protocols also introduce an additional hierarchy with application instance certificates, which are obtained for use with specific applications.

IDevIDs are intended for device identity and initial onboarding or bootstrapping protocols, such as the Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol [RFC8995] or LwM2M Bootstrap [LwM2M-T] [LwM2M-C]. The use of IDevIDs is intentionally limited to such onboarding scenarios even though they often have a long lifetime, or do not expire at all.

There are, however, multiple onboarding and bootstrapping approaches in use. Some of them use TLS and therefore use the IDevID for client authentication, while others, such as FIDO Device Onboarding (FDO) [FDO], do not use TLS/DTLS for client authentication. In many cases, the IDevID profile and content are defined by those specifications. For these reasons, this specification focuses on the description of operational certificates such as LDevIDs.

This document uses the terminology and some of the rules for populating certificate content defined in IEEE 802.1AR. However, this specification does not claim conformance to IEEE 802.1AR, which is broader and mandates hardware, security, and process requirements outside the constraints of many IoT deployments. This profile borrows terminology and selected certificate fields from IEEE 802.1AR but intentionally omits those broader requirements.

17.1. All Certificates

This section outlines the requirements for X.509 certificates that apply to all PKI entities. These requirements apply to certificates issued within the IoT device PKI (i.e., root, subordinate and end entity certificates used to authenticate IoT devices), rather than to public WebPKI server certificates. The section focuses on X.509 v3 certificates.

17.1.1. Version

Certificates MUST be of type X.509 v3.

17.1.2. Serial Number

The serial number MUST be unique for each certificate issued by a given CA (i.e., the issuer name and the serial number uniquely identify a certificate). [RFC5280] limits this field to a maximum of 20 octets. To reduce the risk of predictable serial numbers, CAs SHOULD generate serial numbers of at least eight (8) octets using a cryptographically secure pseudo-random number generator.

17.1.3. Signature

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

Note: In contrast to IEEE 802.1AR this specification does not require end entity certificates, subordinate CA certificates, and CA certificates to use the same signature algorithm. Furthermore, this specification does not utilize RSA for use with constrained IoT devices and networks. For certificates expected to be validated by constrained IoT devices, CAs SHOULD select signature algorithms supported by those devices to ensure successful validation (e.g., ECDSA P-256). Different certificates in the same chain MAY use different signature algorithms when the relying devices support validation of the resulting chain.

17.1.4. Issuer

The issuer field MUST contain a non-empty distinguished name (DN) of the entity that has signed and issued the certificate in accordance with [RFC5280].

17.1.5. Validity

Vendors must determine the expected lifespan of their IoT devices. This decision directly affects how long firmware and software updates are provided for, as well as the level of maintenance that customers can expect. It also affects the maximum validity period of certificates.

Constrained devices often lack precise UTC time; implementations SHOULD treat time checks with coarse granularity (e.g., day- or hour-level) and ignore leap seconds when validating notAfter. For devices without a reliable source of time we advise the use of a device management solution, which typically includes a certificate management protocol, to manage certificates used by the device over their lifecycle. 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.

In many IoT deployments, IDevIDs are provisioned with an unlimited lifetime, as described in [IEEE-802.1AR]. This helps prevent devices from being accidentally bricked due to certificate expiration. A real-world example occurred in 2018, when Oculus Rift headsets became unusable after an Oculus certificate expired [Toms-Hardware-Oculus-Rift-2018]. Oculus later issued a manual patch, as the expired certificate also blocked the standard software update path.

For this purpose, the special GeneralizedTime value 99991231235959Z is used in the notAfter field, as described in Section 4.1.2.5 of [RFC5280]. However, the CA certificate and subordinate CA certificates in the certification path may still have finite validity periods. Careful consideration is therefore required before issuing IDevID certificates with no maximum validity period, since an effectively unlimited certificate lifetime is only useful if the relevant certification path remains usable for the intended lifetime of the device.

LDevID certificates are, however, issued by the operator or owner, and may be renewed at a regular interval using protocols, such as Enrollment over Secure Transport (EST) [RFC7030] or Certificate Management Protocol (CMP) [RFC9810] [RFC9483]. It is therefore RECOMMENDED to limit the lifetime of these LDevID certificates using the notBefore and notAfter fields, as described in Section 4.1.2.5 of [RFC5280]. 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.

17.1.6. Subject Public Key Info

The subjectPublicKeyInfo field 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]. This specification assumes that devices support one of the following algorithms:

* id-ecPublicKey with secp256r1,
* id-ecPublicKey with secp384r1, and
* id-ecPublicKey with secp521r1.

There is no requirement to use CA certificates and certificates of subordinate CAs to use the same algorithm as the end entity certificate. Certificates with longer lifetime may well use a cryptographically stronger algorithm. However, CAs (or their administrators) that issue certificates intended to be validated by constrained IoT devices SHOULD select algorithms supported by those devices to ensure successful validation. Longer-lived CA certificates MAY intentionally use stronger or different algorithms if the target devices are expected to validate such chains successfully.

17.1.7. Certificate Revocation Checks

Constrained IoT devices often lack the resources to perform traditional Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP) checks. Consistent with the guidance in Section 4.4.3 of [RFC7925], neither OCSP nor CRLs are used by constrained IoT devices during the TLS handshake.

Instead, IoT deployments generally rely on short-lived end-entity certificates managed via automated onboarding and management protocols (such as Lightweight Machine-to-Machine [LwM2M-T] [LwM2M-C]). Because these protocols can distribute and update certificates on demand, they make real-time revocation checks largely unnecessary.

Since these checks are bypassed, the CRL Distribution Points extension and the Authority Information Access (AIA) extension for OCSP SHOULD NOT be included in IoT device certificates. If they are present, they MUST NOT be marked critical. However, the AIA extension MAY be used to provide the caIssuer access method, enabling peers with sufficient resources to fetch certificate chains.

When designing the application layer, developers must account for the fact that updating a certificate does not automatically affect existing, long-lived TLS sessions. TLS alone does not mandate continuous validity checks once a connection is established. Furthermore, TLS 1.3 natively supports only client-to-server post-handshake authentication. Achieving mutual post-handshake authentication requires Exported Authenticators [RFC9261], which requires the application-layer protocol to carry the authentication payload. Therefore, if continuous validation is strictly required for a long-lived connection, it is the application's responsibility to enforce this policy by actively triggering re-authentication or tearing down and re-establishing the TLS session.

Ultimately, instead of attempting to perform revocation checks directly on the constrained device, it is RECOMMENDED to delegate this responsibility to the IoT device operator, who can take the necessary administrative actions (such as deploying updated certificates) to keep the network secure and operational. While the above recommendation is valid in most cases, it should be considered carefully on a case-by-case basis, taking into account the security risks associated with not re-authenticating peers and the cost/complexity of implementing an application-layer solution.

17.2. Root CA Certificate

This section outlines the requirements for root CA certificates.

17.2.1. Subject

[RFC5280] mandates that Root CA certificates MUST have a non-empty subject field. The subject field MUST contain the commonName, the organizationName, and the countryName attribute and MAY contain an organizationalUnitName attribute. If a subjectAltName extension is present, it SHOULD be set to a value consistent with the subject and SHOULD NOT be marked critical.

17.2.2. Authority Key Identifier

Section 4.2.1.1 of [RFC5280] defines the Authority Key Identifier as follows: "The authority key identifier extension provides a means of identifying the public key corresponding to the private key used to sign a certificate. This extension is used where an issuer has multiple signing keys."

The Authority Key Identifier extension SHOULD be set to aid path construction. If it is set, it MUST NOT be marked critical, and MUST contain the subjectKeyIdentifier of this certificate.

17.2.3. Subject Key Identifier

Section 4.2.1.2 of [RFC5280] defines the SubjectKeyIdentifier as follows: "The subject key identifier extension provides a means of identifying certificates that contain a particular public key."

The Subject Key Identifier extension MUST be set, MUST NOT be marked critical, and MUST contain the key identifier of the public key contained in the subject public key info field.

The subjectKeyIdentifier is used by path construction algorithms to identify which CA has signed a subordinate certificate.

17.2.4. Key Usage

[RFC5280] defines the key usage field as follows: "The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing) of the key contained in the certificate."

The Key Usage extension SHOULD be set; if it is set, it MUST be marked critical, and the keyCertSign purpose MUST be set. If the Root CA issues CRLs, the cRLSign purpose MUST also be set. Additional key usages MAY be set depending on the intended usage of the public key. The digitalSignature purpose is not required for a Root CA certificate.

[RFC5280] defines the extended key usage as follows: "This extension indicates one or more purposes for which the certified public key may be used, in addition to or in place of the basic purposes indicated in the key usage extension."

This extendedKeyUsage extension MUST NOT be set in CA certificates.

17.2.5. Basic Constraints

[RFC5280] states that "The Basic Constraints extension identifies whether the subject of the certificate is a CA and the maximum depth of valid certification paths that include this certificate. The cA boolean indicates whether the certified public key may be used to verify certificate signatures."

For the pathLenConstraint RFC 5280 makes further statements:

* "The pathLenConstraint field is meaningful only if the cA boolean is asserted and the key usage extension, if present, asserts the keyCertSign bit. In this case, it gives the maximum number of non-self-issued intermediate certificates that may follow this certificate in a valid certification path."
* "A pathLenConstraint of zero indicates that no non-self-issued intermediate CA certificates may follow in a valid certification path."
* "Where pathLenConstraint does not appear, no limit is imposed."
* "Conforming CAs MUST include this extension in all CA certificates that contain public keys used to validate digital signatures on certificates and MUST mark the extension as critical in such certificates."

The Basic Constraints extension MUST be set, MUST be marked critical, the cA flag MUST be set to true and the pathLenConstraint MUST be omitted.

Omitting pathLenConstraint follows common root CA practice but is not meant to encourage arbitrarily deep certification hierarchies in IoT deployments. Shallow hierarchies remain preferable for constrained devices.

17.3. Subordinate CA Certificate

This section outlines the requirements for subordinate CA certificates.

17.3.1. Subject

The subject field MUST be set and MUST contain the commonName, the organizationName, and the countryName attribute and MAY contain an organizationalUnitName attribute.

17.3.2. Authority Key Identifier

The Authority Key Identifier extension MUST be set, MUST NOT be marked critical, and MUST contain the subjectKeyIdentifier of the CA that issued this certificate.

17.3.3. Subject Key Identifier

The Subject Key Identifier extension MUST be set, MUST NOT be marked critical, and MUST contain the key identifier of the public key contained in the subject public key info field.

17.3.4. Key Usage

The Key Usage extension MUST be set, MUST be marked critical, the keyCertSign or cRLSign purposes MUST be set, and the digitalSignature purpose SHOULD be set.

Subordinate certification authorities SHOULD NOT have any extendedKeyUsage. [RFC5280] reserves EKUs to be meaningful only in end entity certificates.

17.3.5. Basic Constraints

The Basic Constraints extension MUST be set, MUST be marked critical, the cA flag MUST be set to true and the pathLenConstraint SHOULD be omitted.

17.3.6. CRL Distribution Point

The CRL Distribution Point extension SHOULD NOT be set. If it is set, it MUST NOT be marked critical and MUST identify the CRL relevant for this certificate.

17.3.7. Authority Information Access

The Authority Information Access (AIA) extension SHOULD NOT be set. If it is set, it MUST NOT be marked critical and MUST identify the location of the certificate of the CA that issued this certificate and the location it provides an online certificate status service (OCSP).

17.4. End Entity Certificate

This section outlines the requirements for end entity certificates.

17.4.1. Subject

This section describes the use of end entity certificate primarily for (D)TLS clients running on IoT devices. Operating (D)TLS servers on IoT devices is possible but less common.

[RFC9525], Section 2 mandates that the subject field not be used to identify a service. However, certain IoT applications (for example, [I-D.ietf-anima-constrained-voucher], [IEEE-802.1AR]) use the subject field to encode the device serial number.

The requirement in Section 4.4.2 of [RFC7925] to only use EUI-64 for end entity certificates as a subject field is lifted.

Two fields are typically used to encode a device identifier, namely the Subject and the subjectAltName fields. Protocol specifications tend to offer recommendations about what identifiers to use and the deployment situation is fragmented.

The subject field MAY include a unique device serial number. If a serial number is included in the Subject DN, it MUST be encoded in the X520SerialNumber attribute. If the serial number is used as an identifier, it SHOULD also be placed in the subjectAltName (e.g., as a URI). e.g., [RFC8995] use requires a serial number in IDevID certificates.

[RFC5280] defines: "The subject alternative name extension allows identities to be bound to the subject of the certificate. These identities may be included in addition to or in place of the identity in the subject field of the certificate."

The subject alternative name extension MAY be set. If it is set, it MUST NOT be marked critical, except when the subject DN contains an empty sequence.

If the EUI-64 format is used to identify the subject of an end entity certificate, it MUST be encoded as a Subject DN using the X520SerialNumber attribute. The contents of the field is a string of the form HH-HH-HH-HH-HH-HH-HH-HH where 'H' is one of the symbols '0'-'9' or 'A'-'F'.

Per [RFC9525] 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. The subjectAltName MUST NOT contain multiple names.

Note: The IEEE 802.1AR recommends to encode information about a Trusted Platform Module (TPM), if present, in the HardwareModuleName (Section 5 of [RFC4108]). This specification does not follow this recommendation.

17.4.2. Authority Key Identifier

The Authority Key Identifier extension MUST be set, MUST NOT be marked critical, and MUST contain the subjectKeyIdentifier of the CA that issued this certificate.

17.4.3. Subject Key Identifier

The Subject Key Identifier MUST NOT be included in end entity certificates, as it can be calculated from the public key, so it just takes up space. End entity certificates are not used in path construction, so there is no ambiguity regarding which certificate chain to use, as there can be with subordinate CAs.

17.4.4. Key Usage

The key usage extension MUST be set and MUST be marked as critical. For signature verification keys the digitialSignature key usage purpose MUST be specified. Other key usages are set according to the intended usage of the key.

As specified in [IEEE-802.1AR], the extendedKeyUsage SHOULD NOT be present in IDevID certificates, as it reduces the utility of the IDevID. For locally assigned LDevID certificates to be usable with TLS, the extendedKeyUsage MUST contain at least one of the following: id-kp-serverAuth or id-kp-clientAuth. The selected EKUs MUST match the intended TLS role of the device or service using the certificate.

18. Update of Trust Anchors

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 [RFC9019]. RFC 7925 recommends to use a software / firmware update mechanism to provision devices with new trust anchors. This approach only addresses the distribution of trust anchors and not end entity certificates or certificates of subordinate CAs.

As an alternative, certificate management protocols like CMP and EST have also offered ways to update trust anchors. See, for example, Section 2.1 of [RFC7030] for an approach to obtaining CA certificates via EST.

19. 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 subordinate 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 led to various standardization efforts. Below is a brief summary of what options an implementer has to reduce the bandwidth requirements of a public key-based key exchange. Note that many of the standardized extensions are not readily available in TLS/DTLS stacks since optimizations typically get implemented last.

* Use elliptic curve cryptography (ECC) instead of RSA-based certificate due to the smaller certificate size. This document recommends the use of elliptic curve cryptography only.
* Avoid deep and complex CA hierarchies to reduce the number of subordinate CA certificates that need to be transmitted and processed. See [I-D.irtf-t2trg-taxonomy-manufacturer-anchors] for a discussion about CA hierarchies. Most security requirements can be satisfied with a PKI depth of 3 (root CA, one subordinate CA, and end entity certificates).
* 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 [RFC9146], 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 Section 5 of [RFC6066] instead of full certificates for clients. When applications perform TLS client authentication via DNS-Based Authentication of Named Entities (DANE) TLSA records then the [I-D.ietf-dance-tls-clientid] specification may be used to reduce the packets on the wire. Note: The term "TLSA" does not stand for anything; it is just the name of the RRtype, as explained in [RFC6698].
* Use certificate compression as defined in [RFC8879].
* 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 is a form of caching or compressing certificates as well.

Although the TLS specification does not explicitly prohibit a server from including trust anchors in the Certificate message - and some implementations do - trust anchors SHOULD NOT be transmitted in this way. Trust anchors are intended to be provisioned through out-of-band mechanisms, and any trust anchor included in the TLS Certificate message cannot be assumed trustworthy by the client. Including them therefore serves no functional purpose and unnecessarily consumes bandwidth.

However, due to limited or asymmetric knowledge between client and server, omitting trust anchors entirely is not always straightforward. Several scenarios highlight this challenge:

* Pinned Server Certificates: In many device-to-cloud deployments (see Section 2.2 of [RFC7452]), clients pin a specific server certificate. If the client has pinned the server certificate, retransmitting it is unnecessary - but the server cannot reliably determine this.
* Root Key Transitions: During root key rollover events (see Section 4.4 of [RFC9810]), new trust anchors may not yet be fully distributed across all devices. This is especially relevant in device-to-device communication Section 2.1 of [RFC7452]), where server roles are determined dynamically and trust anchor distribution may be inconsistent.
* Non-Root Trust Anchors: In some deployments, the client’s trust anchor may be an intermediate CA rather than a root certificate. The server, lacking knowledge of the client’s trust store, cannot always select a certificate chain that aligns with the client’s trust anchor. To mitigate this, the client MAY include the Trusted CA Indication extension (see Section 6 of [RFC6066]) in its ClientHello to signal the set of trust anchors it supports.

[RFC9810] assumes the presence of a shared directory service for certificate retrieval. In constrained or isolated IoT environments, this assumption does not hold. Trust anchors are often distributed via firmware updates or fetched periodically using certificate management protocols, such as EST (e.g., the /cacerts endpoint).

To support transitional trust states during trust anchor updates, devices MUST handle both:

* newWithOld: a certificate where the new trust anchor is signed by the old one, enabling communication with peers that have not yet received the update.
* oldWithNew: a certificate where the old trust anchor is signed by the new one, enabling verification of peers that still rely on the older anchor.

These certificates may be presented as an unordered set, and devices may not be able to distinguish their roles without additional metadata.

A complication arises when the client's trust anchor is not a widely trusted root CA. In that case, the server cannot determine in advance which trust anchors the client has. To address this, the client MAY include the Trusted CA Indication extension [RFC6066] in its ClientHello to signal the set of trust anchors it supports, allowing the server to select an appropriate certificate chain.

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).

20. Ciphersuites

According to Section 4.5.3 of [RFC9147], the use of AES-CCM with 8-octet authentication tags (CCM_8) is considered unsuitable for general use with DTLS. This is because it has low integrity limits (i.e., high sensitivity to forgeries) which makes endpoints that negotiate ciphersuites based on such AEAD vulnerable to a trivial DoS attack. See also Sections 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, [RFC9147] 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 required by [RFC7925] and [CoAP] rely on CCM_8, there is no alternate ciphersuite available for applications that aim to eliminate the security and availability threats related to CCM_8 while retaining interoperability with the larger ecosystem.

In order to ameliorate the situation, it is RECOMMENDED that implementations support the following two ciphersuites for TLS 1.3:

* TLS_AES_128_GCM_SHA256
* TLS_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 [RFC9147]. 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.

TLS 1.3 enforces deterministic nonce generation for all AEAD cipher suites. However, this is not the case for TLS 1.2. Therefore, when using the GCM-based cipher suite with TLS 1.2, the recommendations in Section 7.2.1 of [RFC9325] relating to deterministic nonce generation apply. In addition, the integrity limits on key usage detailed in Section 4.4 of [RFC9325] also apply.

Table 1 summarizes the recommendations regarding ciphersuites:

Ciphersuite	MTI Requirement
TLS_AES_128_CCM_8_SHA256	MUST-
TLS_AES_128_CCM	SHOULD+
TLS_AES_128_GCM_SHA256	SHOULD+

Table 1: TLS 1.3 Ciphersuite Requirements

21. Fault Attacks on Deterministic Signature Schemes

A number of passive side-channel attacks as well as active fault-injection attacks (e.g., [Ambrose2017]) have been demonstrated to be successful in allowing a malicious third party to gain information about the signing key if a fully deterministic signature scheme (e.g., ECDSA [RFC6979] or EdDSA [RFC8032]) is used.

Most of these attacks assume physical access to the device and are therefore especially relevant to smart cards as well as IoT deployments with poor or non-existent physical security.

In this security model, it is recommended to combine both randomness and determinism, for example, as described in [I-D.irtf-cfrg-det-sigs-with-noise].

22. Post-Quantum Cryptography (PQC) Considerations

This section is informational and provides deployment guidance only; it does not add normative requirements to this profile.

The recommendations and ciphersuites in this profile are based on classical cryptography and are not quantum-resistant.

As detailed in [I-D.ietf-pquip-pqc-engineers], the IETF is actively working to address the challenges of adopting PQC in various protocols, including TLS. The document highlights key aspects engineers must consider, such as algorithm selection, performance impacts, and deployment strategies. It emphasizes the importance of gradual integration of PQC to ensure secure communication while accounting for the increased computational, memory, and bandwidth requirements of PQC algorithms. These challenges are especially relevant in the context of IoT, where device constraints limit the adoption of larger key sizes and more complex cryptographic operations [PQC-PERF]. Besides, any choice need to careful evaluate the associated energy requirements [PQC-ENERGY].

The work of incorporating PQC into TLS [I-D.ietf-uta-pqc-app] [I-D.ietf-pquip-pqc-hsm-constrained] is still ongoing, with key exchange message sizes increasing due to larger public keys. These larger keys demand more flash storage and higher RAM usage, presenting significant obstacles for resource-constrained IoT devices. The transition from classical cryptographic algorithms to PQC will be a significant challenge for constrained IoT devices, requiring careful planning to select hardware suitable for the task considering the lifetime of an IoT product.

23. Privacy Considerations

The privacy considerations in Section 22 of [RFC7925] largely continue to apply. However, compared to TLS 1.2 and DTLS 1.2, TLS 1.3 and DTLS 1.3 encrypt a larger portion of the handshake, which reduces the amount of identity and credential metadata observable on the wire by passive attackers. Extensions, such as the encrypted ClientHello, further increase privacy protection.

Certificate fields can expose stable device identifiers and other metadata. In particular, IDevIDs and LDevIDs may reveal manufacturer identity, device serial numbers, or other information to peers. Protection against passive observers is, however, substantially improved since certificates are not transmitted in the clear in TLS 1.3 and DTLS 1.3.

Some deployments use the mechanisms discussed in the Certificate Overhead section, such as certificate URLs or external certificate retrieval, instead of always transmitting full certificates in the handshake. In these cases, the privacy properties differ because stable identifiers may be exposed to retrieval services, directories, or to observers of those retrieval transactions.

Where privacy is a deployment requirement, implementations and PKI profiles should include only the minimum identity information needed for authorization and interoperability.

When Connection IDs are used with DTLS 1.3, CID negotiation in post-handshake messages is encrypted and integrity protected. In addition, record sequence numbers are encrypted. Compared to DTLS 1.2 CID, this makes tracking by on-path adversaries more difficult and improves privacy in multi-home and mobile deployments (Section 11 of [RFC9147]).

24. Security Considerations

This entire document is about security. One specific trade-off concerns root certificates that are either very long-lived or never expire. While they can reduce maintenance pressure in long-lived IoT deployments, they also increase the consequences of key compromise, policy errors and inadequate rollover planning. Furthermore, they can make cryptographic transitions more operationally expensive.

25. IANA Considerations

This document makes no requests to IANA.

26. References

26.1. Normative References

[I-D.ietf-tls-dtls-rrc]
Tschofenig, H., Kraus, A., and T. Fossati, "Return Routability Check for DTLS 1.2 and DTLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls-rrc-20, 14 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-dtls-rrc-20>.

[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>.

[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>.

[RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <https://www.rfc-editor.org/rfc/rfc8221>.

[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>.

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

[RFC9258] Benjamin, D. and C. A. Wood, "Importing External Pre-Shared Keys (PSKs) for TLS 1.3", RFC 9258, DOI 10.17487/RFC9258, July 2022, <https://www.rfc-editor.org/rfc/rfc9258>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/rfc/rfc9325>.

[RFC9525] Saint-Andre, P. and R. Salz, "Service Identity in TLS", RFC 9525, DOI 10.17487/RFC9525, November 2023, <https://www.rfc-editor.org/rfc/rfc9525>.

26.2. Informative References

[Ambrose2017] Ambrose, C., Bos, J. W., Fay, B., Joye, M., Lochter, M., and B. Murray, "Differential Attacks on Deterministic Signatures", 2017, <https://eprint.iacr.org/2017/975.pdf>.

[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>.

[FDO] FIDO Alliance, "FIDO Device Onboard Specification 1.1", April 2022, <https://fidoalliance.org/specifications/download-iot-specifications/>.

[I-D.ietf-anima-constrained-voucher] Richardson, M., Van der Stok, P., Kampanakis, P., and E. Dijk, "Constrained Bootstrapping Remote Secure Key Infrastructure (cBRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-voucher-30, 27 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-constrained-voucher-30>.

[I-D.ietf-cose-cbor-encoded-cert] Mattsson, J. P., Selander, G., Raza, S., H旦glund, J., Furuhed, M., and L. Liao, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-19, 11 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-19>.

[I-D.ietf-dance-tls-clientid] Huque, S. and V. Dukhovni, "TLS Extension for DANE Client Identity", Work in Progress, Internet-Draft, draft-ietf-dance-tls-clientid-07, 17 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-dance-tls-clientid-07>.

[I-D.ietf-httpbis-secondary-server-certs]
Gorbaty, E. and M. Bishop, "Secondary Certificate Authentication of HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-httpbis-secondary-server-certs-01, 12 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-secondary-server-certs-01>.

[I-D.ietf-iotops-7228bis]
Bormann, C., Ersue, M., Ker辰nen, A., and C. Gomez, "Terminology for Constrained-Node Networks", Work in Progress, Internet-Draft, draft-ietf-iotops-7228bis-08, 15 May 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-iotops-7228bis-08>.

[I-D.ietf-pquip-pqc-engineers]
Banerjee, A., Reddy.K, T., Schoinianakis, D., Hollebeek, T., and M. Ounsworth, "Post-Quantum Cryptography for Engineers", Work in Progress, Internet-Draft, draft-ietf-pquip-pqc-engineers-14, 25 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqc-engineers-14>.

[I-D.ietf-pquip-pqc-hsm-constrained]
Reddy.K, T., Wing, D., S, B., and K. Kwiatkowski, "Adapting Constrained Devices for Post-Quantum Cryptography", Work in Progress, Internet-Draft, draft-ietf-pquip-pqc-hsm-constrained-05, 1 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqc-hsm-constrained-05>.

[I-D.ietf-uta-pqc-app]
Reddy.K, T. and H. Tschofenig, "Post-Quantum Cryptography Recommendations for TLS-based Applications", Work in Progress, Internet-Draft, draft-ietf-uta-pqc-app-01, 24 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-uta-pqc-app-01>.

[I-D.irtf-cfrg-aead-limits]
G端nther, F., Thomson, M., and C. A. Wood, "Usage Limits on AEAD Algorithms", Work in Progress, Internet-Draft, draft-irtf-cfrg-aead-limits-11, 4 December 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-aead-limits-11>.

[I-D.irtf-cfrg-det-sigs-with-noise]
Mattsson, J. P., Thormarker, E., and S. Ruohomaa, "Hedged ECDSA and EdDSA Signatures", Work in Progress, Internet-

Draft, draft-irtf-cfrg-det-sigs-with-noise-05, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-det-sigs-with-noise-05>.

[I-D.irtf-t2trg-taxonomy-manufacturer-anchors] Richardson, M., "A Taxonomy of operational security considerations for manufacturer installed keys and Trust Anchors", Work in Progress, Internet-Draft, draft-irtf-t2trg-taxonomy-manufacturer-anchors-16, 22 April 2026, <https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-taxonomy-manufacturer-anchors-16>.

[IEEE-802.1AR] "ISO/IEC/IEEE International Standard for Telecommunications and exchange between information technology systems--Requirements for local and metropolitan area networks--Part 1AR:Secure device identity", IEEE, DOI 10.1109/ieeestd.2020.9052099, ISBN ["9781504465885"], March 2020, <https://doi.org/10.1109/ieeestd.2020.9052099>.

[LwM2M-C] OMA SpecWorks, "Lightweight Machine to Machine (LwM2M) V.1.2.2 Technical Specification: Core", June 2024, <https://www.openmobilealliance.org/release/LightweightM2M/V1_2_2-20240613-A/>.

[LwM2M-T] OMA SpecWorks, "Lightweight Machine to Machine (LwM2M) V.1.2.2 Technical Specification: Transport Bindings", June 2024, <https://www.openmobilealliance.org/release/LightweightM2M/V1_2_2-20240613-A/>.

[PQC-ENERGY] Tasopoulos, G., Dimopoulos, C., Fournaris, A., Zhao, R., Sakzad, A., and R. Steinfeld, "Energy Consumption Evaluation of Post-Quantum TLS 1.3 for Resource-Constrained Embedded Devices", ACM, Proceedings of the 20th ACM International Conference on Computing Frontiers pp. 366-374, DOI 10.1145/3587135.3592821, May 2023, <https://doi.org/10.1145/3587135.3592821>.

[PQC-PERF] Tasopoulos, G., Li, J., Fournaris, A., Zhao, R., Sakzad, A., and R. Steinfeld, "Performance Evaluation of Post-Quantum TLS 1.3 on Resource-Constrained Embedded Systems", Springer International Publishing, Lecture Notes in Computer Science pp. 432-451, DOI 10.1007/978-3-031-21280-2_24, ISBN ["9783031212796", "9783031212802"], 2022, <https://doi.org/10.1007/978-3-031-21280-2_24>.

[RFC4108] Housley, R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, DOI 10.17487/RFC4108, August 2005, <https://www.rfc-editor.org/rfc/rfc4108>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <https://www.rfc-editor.org/rfc/rfc4279>.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, DOI 10.17487/RFC5216, March 2008, <https://www.rfc-editor.org/rfc/rfc5216>.

[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/rfc/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/rfc/rfc6066>.

[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/rfc/rfc6698>.

[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/rfc/rfc6979>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <https://www.rfc-editor.org/rfc/rfc7030>.

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

[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>.

[RFC7452] Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson, "Architectural Considerations in Smart Object Networking", RFC 7452, DOI 10.17487/RFC7452, March 2015, <https://www.rfc-editor.org/rfc/rfc7452>.

[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>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/rfc/rfc8032>.

[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>.

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

[RFC8937] Cremers, C., Garratt, L., Smyshlyaev, S., Sullivan, N., and C. Wood, "Randomness Improvements for Security Protocols", RFC 8937, DOI 10.17487/RFC8937, October 2020, <https://www.rfc-editor.org/rfc/rfc8937>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/rfc/rfc8995>.

[RFC9019] Moran, B., Tschofenig, H., Brown, D., and M. Meriac, "A Firmware Update Architecture for Internet of Things", RFC 9019, DOI 10.17487/RFC9019, April 2021, <https://www.rfc-editor.org/rfc/rfc9019>.

[RFC9146] Rescorla, E., Ed., Tschofenig, H., Ed., Fossati, T., and A. Kraus, "Connection Identifier for DTLS 1.2", RFC 9146, DOI 10.17487/RFC9146, March 2022, <https://www.rfc-editor.org/rfc/rfc9146>.

[RFC9150] Cam-Winget, N. and J. Visoky, "TLS 1.3 Authentication and Integrity-Only Cipher Suites", RFC 9150, DOI 10.17487/RFC9150, April 2022, <https://www.rfc-editor.org/rfc/rfc9150>.

[RFC9180] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/rfc/rfc9180>.

[RFC9190] Preu Mattsson, J. and M. Sethi, "EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3", RFC 9190, DOI 10.17487/RFC9190, February 2022, <https://www.rfc-editor.org/rfc/rfc9190>.

[RFC9250] Huitema, C., Dickinson, S., and A. Mankin, "DNS over Dedicated QUIC Connections", RFC 9250, DOI 10.17487/RFC9250, May 2022, <https://www.rfc-editor.org/rfc/rfc9250>.

[RFC9261] Sullivan, N., "Exported Authenticators in TLS", RFC 9261, DOI 10.17487/RFC9261, July 2022, <https://www.rfc-editor.org/rfc/rfc9261>.

[RFC9483] Brockhaus, H., von Oheimb, D., and S. Fries, "Lightweight Certificate Management Protocol (CMP) Profile", RFC 9483, DOI 10.17487/RFC9483, November 2023, <https://www.rfc-editor.org/rfc/rfc9483>.

[RFC9810] Brockhaus, H., von Oheimb, D., Ounsworth, M., and J. Gray, "Internet X.509 Public Key Infrastructure -- Certificate Management Protocol (CMP)", RFC 9810, DOI 10.17487/RFC9810, July 2025, <https://www.rfc-editor.org/rfc/rfc9810>.

[RFC9849] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", RFC 9849, DOI 10.17487/RFC9849, March 2026, <https://www.rfc-editor.org/rfc/rfc9849>.

[Toms-Hardware-Oculus-Rift-2018] Colaner, S., "How To Patch Your Oculus Rift", March 2018, <https://www.tomshardware.com/news/oculus-rift-runtime-error-fix%2C36629.html>.

Acknowledgments

We would like to thank Henk Birkholz, Hendrik Brockhaus, Ben Kaduk, John Mattsson, Daniel Migault, Tiru Reddy, Rich Salz, and Marco Tiloca.

Furthermore, we would like to thank our security area director Deb Cooley for her detailed review comments.

Contributors

Juliusz Sosinowicz

Achim Kraus

Authors' Addresses

Hannes Tschofenig
University of the Bundeswehr Munich
85577 Neubiberg
Germany
Email: hannes.tschofenig@gmx.net

Thomas Fossati
Linaro
Email: Thomas.Fossati@linaro.org

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca

Daniel Migault
Ericsson
Canada
Email: daniel.migault@ericsson.com