

SEAT Working Group
Internet-Draft
Intended status: Standards Track
Expires: 31 August 2026

M. U. Sardar
TU Dresden
T. Fossati
Linaro
T. Reddy
Nokia
Y. Sheffer
Intuit
H. Tschofenig
H-BRS
I. Mihalcea
Arm Limited
27 February 2026

Remote Attestation with Exported Authenticators
draft-fossati-seat-expat-02

Abstract

This specification defines a method for two parties in a communication interaction to exchange Evidence and Attestation Results using exported authenticators, as defined in [RFC9261]. Additionally, it introduces the `cmw_attestation` extension, which allows attestation credentials to be included directly in the Certificate message sent during the Exported Authenticator-based post-handshake authentication. The approach supports both the passport and background check models from the RATS architecture while ensuring that attestation remains bound to the underlying communication channel.

About This Document

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

The latest revision of this draft can be found at <https://tls-attestation.github.io/exported-attestation/draft-fossati-seat-expat.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-fossati-seat-expat/>.

Discussion of this document takes place on the SEAT Working Group mailing list (<mailto:seat@ietf.org>), which is archived at <https://datatracker.ietf.org/wg/seat/about/>. Subscribe at <https://www.ietf.org/mailman/listinfo/seat/>.

Source for this draft and an issue tracker can be found at <https://github.com/tls-attestation/exported-attestation>.

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 31 August 2026.

Copyright Notice

Copyright (c) 2026 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	4
3. cmw_attestation Extension to the Authenticator's Certificate message	5
3.1. Negotiation of the cmw_attestation Extension	6
3.2. Usage in Exported Authenticator-based Post-Handshake Authentication	6
3.3. Ensuring Compatibility with X.509 Certificate Validation	7
3.4. Applicability to Client and Server Authentication	7
4. Architecture	8
4.1. Client as Attester	9
4.1.1. Passport Model	9
4.1.2. Background-check Model	10
4.2. Server as Attester	11

4.3. API Requirements for Attestation Support	11
5. Cryptographic Binding of the Evidence to the TLS Connection	12
5.1. Attestation Binder	12
5.2. Binding the Authenticator Identity Key (AIK) to the TEE	13
6. Operational Overview	14
7. Security Considerations	15
7.1. Security Guarantees	15
7.2. Using the TLS Connection	16
7.3. Timing for Remote Attestation	16
7.4. Evidence Freshness	17
8. Privacy Considerations	17
8.1. Client as Attester	17
8.2. Server as Attester	18
9. IANA Considerations	18
9.1. TLS Extension Type Registration	18
9.2. TLS Flags Extension Registry	18
10. References	19
10.1. Normative References	19
10.2. Informative References	20
Acknowledgements	21
Appendix	21
Post-handshake vs. Intra-handshake Privacy	21
Post-handshake vs. Intra-handshake Security	22
Document History	22
Authors' Addresses	22

1. Introduction

There is a growing need to demonstrate to a remote party that cryptographic keys are stored in a secure element, the device is in a known good state, secure boot has been enabled, and that low-level software and firmware have not been tampered with. Remote attestation provides this capability.

More technically, an Attester produces a signed collection of Claims that constitute Evidence about its running environment(s). A Relying Party may consult an Attestation Result produced by a Verifier that has appraised the Evidence to make policy decisions regarding the trustworthiness of the Target Environment being assessed. This is, in essence, what [RFC9334] defines.

At the time of writing, several standard and proprietary remote attestation technologies are in use. This specification aims to remain as technology-agnostic as possible concerning implemented remote attestation technologies. To streamline attestation in TLS, this document introduces the `cmw_attestation` extension, which allows

attestation credentials to be conveyed directly in the Certificate message during the Exported Authenticator-based post-handshake authentication. This eliminates reliance on real-time certificate issuance from a Certificate Authority (CA), reducing handshake delays while ensuring Evidence remains bound to the TLS connection. The extension supports both the passport and background check models from the RATS architecture, enhancing flexibility for different deployment scenarios.

This document builds upon three foundational specifications:

- * RATS (Remote Attestation Procedures) Architecture [RFC9334]: It defines how remote attestation systems establish trust between parties by exchanging Evidence and Attestation Results. These interactions can follow different models, such as the passport or the background check model, depending on the order of data flow in the system.
- * TLS Exported Authenticators [RFC9261]: It offers bi-directional post-handshake authentication. Once a TLS connection is established, both peers can send an authenticator request message at any point after the handshake. This message from the server and the client uses the CertificateRequest and the ClientCertificateRequest messages, respectively. The peer receiving the authenticator request message can respond with an Authenticator consisting of Certificate, CertificateVerify, and Finished messages. These messages can then be validated by the other peer.
- * RATS Conceptual Messages Wrapper (CMW) [I-D.ietf-rats-msg-wrap]: CMW provides a structured encapsulation of Evidence and Attestation Result, abstracting the underlying attestation technology.

This specification introduces the `cmw_attestation` extension, enabling Evidence to be included directly in the Certificate message during the Exported Authenticator-based post-handshake authentication defined in [RFC9261].

2. 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.

The reader is assumed to be familiar with the vocabulary and concepts defined in [RFC9334] and [RFC9261].

"Remote attestation credentials", or "attestation credentials", is used to refer to both Evidence and attestation results, when no distinction needs to be made between them.

3. cmw_attestation Extension to the Authenticator's Certificate message

This document introduces a new extension, called `cmw_attestation`, to the Authenticator's Certificate message. This extension allows Evidence or Attestation Results to be included in the extensions field of the end-entity certificate in the Authenticator's Certificate message.

As defined in Section 4.4.2 of [I-D.ietf-tls-rfc8446bis], the TLS Certificate message consists of a `certificate_list`, which is a sequence of `CertificateEntry` structures. Each `CertificateEntry` contains a certificate and a set of associated extensions. The `cmw_attestation` extension MUST appear only in the first `CertificateEntry` of the Certificate message and applies exclusively to the end-entity certificate. It MUST NOT be included in entries corresponding to intermediate or trust anchor certificates. This design ensures that attestation information is tightly bound to the entity being authenticated.

The `cmw_attestation` extension is only included in the Certificate message during Exported Authenticator-based post-handshake authentication. This ensures that the attestation credentials are conveyed within the Certificate message, eliminating the need for modifications to the X.509 certificate structure.

```
struct {  
    opaque cmw_data<1..2^16-1>;  
} CMWAttestation;
```

`cmw_data`: Encapsulates the attestation credentials in CMW format [I-D.ietf-rats-msg-wrap]. The `cmw_data` field is encoded using CBOR or JSON.

This approach eliminates the need for real-time certificate issuance from a Certificate Authority (CA) and minimizes handshake delays. Typically, CAs require several seconds to minutes to issue a certificate due to verification steps such as validating subject identity, signing the certificate, and distributing it. These delays introduce latency into the TLS handshake, making real-time certificate generation impractical. The `cmw_attestation` extension circumvents this issue by embedding attestation data within the Certificate message itself, removing reliance on external certificate issuance processes.

3.1. Negotiation of the `cmw_attestation` Extension

Negotiation of support `cmw_attestation` extension follows the model defined in Section 5.2 of [RFC9261].

Endpoints that wish to receive attestation credentials using Exported Authenticators MUST indicate support by including an empty `cmw_attestation` extension in the CertificateRequest or ClientCertificateRequest message. The presence of this empty extension indicates that the requester understands this specification and is willing to process an attestation credential in the peer's Certificate message.

An endpoint that supports this extension and receives a request containing it MAY include the `cmw_attestation` extension in its Certificate message, populated with attestation data. If the `cmw_attestation` extension appears in a Certificate message without it having been previously offered in the corresponding request, the receiver MUST abort the authenticator verification with an "unsupported_extension" alert. As specified in Section 9.3 of [I-D.ietf-tls-rfc8446bis], endpoints that do not recognize the `cmw_attestation` extension in a CertificateRequest or ClientCertificateRequest MUST ignore it and continue processing the message as if the extension were absent.

3.2. Usage in Exported Authenticator-based Post-Handshake Authentication

The `cmw_attestation` extension is designed to be used exclusively in Exported Authenticator-based post-handshake authentication as defined in [RFC9261]. It allows attestation credentials to be transmitted in the Authenticator's Certificate message only in response to an Authenticator Request. This ensures that attestation credentials are provided on demand rather than being included in the initial TLS handshake.

To maintain a cryptographic binding between the Evidence and the authentication request, the `cmw_attestation` extension MUST be associated with the `certificate_request_context` of the corresponding `CertificateRequest` or `ClientCertificateRequest` message (from the Server or Client, respectively). This association ensures that the Evidence is specific to the authentication event.

3.3. Ensuring Compatibility with X.509 Certificate Validation

The `cmw_attestation` extension does not modify or replace X.509 certificate validation mechanisms. It serves as an additional source of authentication data rather than altering the trust model of PKI-based authentication. Specifically:

- * Certificate validation (e.g., signature verification, revocation checks) MUST still be performed according to TLS [I-D.ietf-tls-rfc8446bis] and PKIX [RFC5280].
- * The attestation credentials carried in `cmw_attestation` MUST NOT be used as a substitute for X.509 certificate validation but can be used alongside standard certificate validation for additional security assurances.
- * Implementations MAY reject connections where the certificate is valid but the attestation credentials is missing or does not meet security policy.

3.4. Applicability to Client and Server Authentication

The `cmw_attestation` extension is applicable to both client and server authentication in Exported Authenticator-based post-handshake authentication.

In TLS, one party acts as the Relying Party, and the other party acts as the Attester. Either the client or the server may fulfill these roles depending on the authentication direction.

The Attester may respond with either:

- * Evidence (Background Check Model):
 - The Attester generates Evidence and includes it in the `cmw_attestation` extension to the Authenticator's Certificate message.
 - The Relying Party forwards the Evidence to an external Verifier for evaluation and waits for an Attestation Result.

- The Relying Party grants or denies access, or continues or terminates the TLS connection, based on the Verifier's Attestation Result.

* Attestation Result (Passport Model):

- The Attester sends Evidence to a Verifier beforehand.
- The Verifier issues an Attestation Result to the Attester.
- The Attester includes the Attestation Result in the cmw_attestation extension to the Authenticator's Certificate message and sends it to the Relying Party.
- The Relying Party validates the Attestation Result directly without needing to contact an external Verifier.

By allowing both Evidence and Attestation Results to be conveyed within cmw_attestation, this mechanism supports flexible attestation workflows depending on the chosen trust model.

4. Architecture

The cmw_attestation extension enables attestation credentials to be included in the Certificate message during Exported Authenticator-based post-handshake authentication, ensuring that attestation remains bound to the TLS connection.

However, applications using this mechanism still need to negotiate the encoding format (e.g., JOSE or COSE) and specify how attestation credentials are processed. This negotiation can be done via application-layer signaling or predefined profiles. Future specifications may define mechanisms to streamline this negotiation.

Upon receipt of a Certificate message containing the cmw_attestation extension, an endpoint MUST take the following steps to validate the attestation credentials:

* Background Check Model:

- Verify Integrity and Authenticity: The Evidence must be cryptographically verified against a known trust anchor, typically provided by the hardware manufacturer.
- Verify Certificate Request Binding and Freshness: The Evidence must be bound to the active TLS connection by verifying that the exporter value in the Evidence matches the exporter value computed using the label "Attestation Binding" and the

certificate_request_context as the exporter context. This verification ensures correct connection binding, provides freshness, and prevents replay.

- Evaluate Security Policy Compliance: The Evidence must be evaluated against the Relying Party's security policies to determine if the attesting device and the private key storage meet the required criteria.

* Passport Model:

- Verify the Attestation Result: The Relying Party MUST check that the Attestation Result is correctly signed by the issuing authority and that it meets the Relying Party's security requirements.

By integrating cmw_attestation directly into the Certificate message during Exported Authenticator-based post-handshake authentication, this approach reduces latency and complexity while maintaining strong security guarantees.

In the following examples, the server possesses an identity certificate, while the client is not authenticated during the initial TLS exchange.

4.1. Client as Attester

4.1.1. Passport Model

Figure 1 shows the passport model.

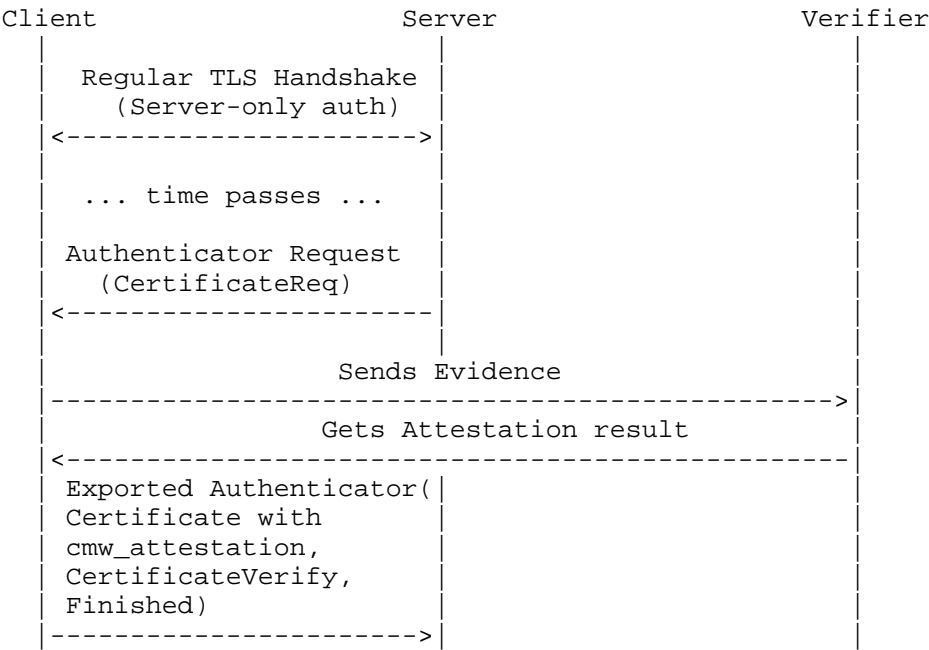


Figure 1: Passport Model with Client as Attester

4.1.2. Background-check Model

Figure 2 shows an example using the background-check model, where TE represents the Target Environment and AE represents the Attesting Environment. The TLS Client within TE establishes a TLS connection with the Server. At a later time, the Server triggers an Authenticator Request. The TE requests Evidence from the AE. The Evidence produced includes the TE’s TCB measurements, including those relating to the TLS Client code and configuration. The TE embeds this Evidence into an Exported Authenticator and sends it to the Server. The Server forwards the Evidence to a Verifier for appraisal and receives an Attestation Result.

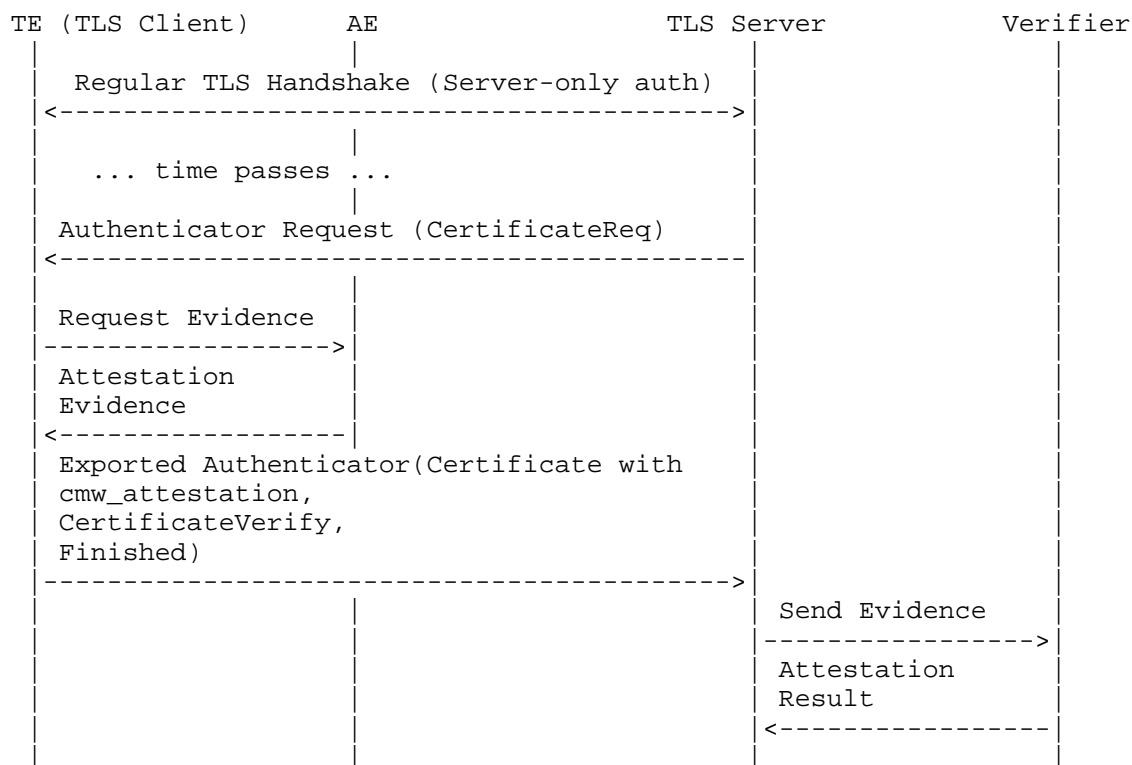


Figure 2: Background Check Model with Client as Attester

4.2. Server as Attester

The flow for the Server as Attester is analogous, except that it uses a ClientCertificateRequest instead of CertificateRequest.

4.3. API Requirements for Attestation Support

To enable attestation workflows, implementations of the Exported Authenticator API MUST support the following:

1. Authenticator Generation

- * The API MUST support the inclusion of attestation credentials within the Certificate message provided as input.

2. Authenticator Validation

- * The API MUST support verification that the Evidence in the Certificate message is cryptographically valid and correctly bound to the TLS connection and the associated `certificate_request_context`.

5. Cryptographic Binding of the Evidence to the TLS Connection

The cryptographic operations defined in this section bind attestation Evidence to a specific TLS connection. This binding prevents replay and relay of attestation Evidence across different TLS connections, and ensures that attestation Evidence presented during a handshake corresponds to the authenticated TLS connection in which it is conveyed.

Attestation Evidence is generated by a TEE and signed using an attestation key. The signed Evidence includes inputs originating from different trust domains.

- * The attestation binder is provided by the TLS stack and serves as an attestation challenge that ensures freshness and binds the attestation to a specific TLS connection.
- * A claim generated by the TEE itself, such as a hash of the AIK public key, which ensures that the attested environment controls the AIK private key used for TLS authentication.

5.1. Attestation Binder

The attester binds the attestation Evidence to the active TLS connection. To do so, the attester derives a binding value using the TLS exporter. The exporter invocation uses:

- * the label "Attestation", and
- * the `certificate_request_context` from the `CertificateRequest` message as the `context_value` (as defined in Section 7.5 of [I-D.ietf-tls-rfc8446bis]). In a Background Check model, this value contains the Verifier-provided nonce; and
- * a `key_length` set to 256-bit (32 bytes).

```
TLS-Exporter("Attestation", certificate_request_context, 32)
```

The binding value is defined as:

```
Hash(public_key || exported_value)
```

where `exported_value` is the output of the TLS exporter and `public_key` is the public key corresponding to the end-entity certificate used for authentication. The Hash function is the hash algorithm associated with the negotiated TLS 1.3 cipher suite (i.e., the HKDF hash).

This binding value is included as an attestation challenge (or nonce) in the attestation Evidence. It binds the Evidence to the specific TLS connection and to the associated `certificate_request_context`, thereby providing freshness and preventing replay across TLS sessions.

The TLS endpoint that receives the attestation Evidence computes the binding value using the same exporter and hash function invocation described for the attester. The endpoint either verifies the exporter binding itself or delegates this check to the Verifier. If it performs the check locally and the values do not match, the attestation Evidence is rejected. If the check is delegated, the endpoint conveys the computed binding value to the Verifier so that the comparison can be carried out during attestation validation.

5.2. Binding the Authenticator Identity Key (AIK) to the TEE

This specification assumes that the private key corresponding to the end-entity certificate carried in the exported authenticator referred to as the Authenticator Identity Key (AIK) is generated inside a TEE and never leaves it. A platform could instead generate the AIK private key outside the TEE and compute the `CertificateVerify` signature using that external key. A Relying Party cannot detect this attack unless additional safeguards are in place.

This risk is particularly relevant in split deployments, where the TLS stack does not reside inside the TEE. In such architectures, attesting the TEE alone does not prove that the AIK private key used by the TLS endpoint was generated, is stored, or is controlled by the TEE.

To address this risk, the Evidence MUST include the hash of the AIK public key (`AIK_pub_hash`). The AIK public key MUST be hashed using the hash algorithm associated with the negotiated TLS cipher suite for the TLS connection in which the Evidence is conveyed.

The Relying Party MUST compute the hash of the AIK public key extracted from the TLS end-entity certificate using the same hash algorithm and verify that it matches the `AIK_pub_hash` included in the Evidence. Successful verification binds the attestation Evidence to the TLS identity used for authentication.

6. Operational Overview

This specification follows the Exported Authenticator model defined in Section 7 of [RFC9261]. In this model, the TLS stack creates and validates post-handshake authentication messages, while the application triggers the exchange and conveys the Authenticator Request and Authenticator messages between peers.

Whenever attestation is required, the application initiates Exported Authenticatorbased post-handshake authentication. The TLS stack constructs an Authenticator Request, using either a CertificateRequest or ClientCertificateRequest message, and indicates support for attestation by including an empty cmw_attestation extension.

On the attesting side, the platform generates RATS conceptual message, either in the form of Evidence or an Attestation Result, depending on the deployment model. The TLS stack embeds the RATS conceptual message in the cmw_attestation extension and completes the Exported Authenticator by generating the corresponding CertificateVerify and Finished messages. The application then conveys the resulting Authenticator to the peer, as defined in [RFC9261].

Upon receipt, the application delivers the Authenticator to the TLS stack for processing. The TLS stack validates the certificate chain, the CertificateVerify signature, and the Finished message, and extracts the cmw_attestation extension. The TLS stack treats the contents of the cmw_attestation extension as opaque.

If the cmw_attestation extension carries Evidence, the application acting as the Relying Party forwards it to a Verifier to obtain an Attestation Result. If the extension carries an Attestation Result, the application validates it directly. Based on the outcome of this processing, the application determines whether to continue using the TLS connection.

While it is technically possible to create or validate Authenticator Requests and Authenticators at the application layer, Section 7 of [RFC9261] recommends that their creation and validation be handled within the TLS library.

7. Security Considerations

This document inherits the security considerations of [RFC9261] and [RFC9334]. The integrity of the exported authenticators must be guaranteed, and any failure in validating Evidence SHOULD be treated as a fatal error in the communication channel. Additionally, in order to benefit from remote attestation, Evidence MUST be protected using dedicated attestation keys chaining back to a trust anchor. This trust anchor will typically be provided by the hardware manufacturer.

This specification assumes that the Hardware Security Module (HSM) or Trusted Execution Environment (TEE) is responsible for generating the Authenticator Identity Key (AIK) key pair and producing either Evidence or Attestation Results. The Evidence or Attestation Results MAY be included in a Certificate Signing Request (CSR), as defined in [I-D.ietf-lamps-csr-attestation], enabling the CA to verify that the AIK private key is securely generated and stored and that the platform meets the required security standards before issuing a certificate.

7.1. Security Guarantees

Note that as a pure cryptographic protocol, attested TLS as-is only guarantees that the identity key used for TLS handshake is known by the confidential environment, such as confidential virtual machine. A number of additional guarantees must be provided by the platform and/or the TLS stack, and the overall security level depends on their existence and quality of assurance:

- * The identity key used for TLS handshake is generated within the trustworthy environment, such as Trusted Platform Module (TPM) or TEE.
- * The identity key used for TLS handshake is never exported or leaked outside the trustworthy environment.
- * For confidential computing use cases, the TLS protocol is implemented within the confidential environment, and is implemented correctly, e.g., it does not leak any session key material.
- * The TLS stack including the code that performs the post-handshake phase must be measured.
- * There must be no other way to initiate generation of evidence except from signed code.

These properties may be explicitly promised ("attested") by the platform, or they can be assured in other ways such as by providing source code, reproducible builds, formal verification etc. The exact mechanisms are out of scope of this document.

7.2. Using the TLS Connection

Remote attestation in this document occurs within the context of a TLS handshake, and the TLS connection remains valid after this process. Care must be taken when handling this TLS connection, as both the client and server must agree that remote attestation was successfully completed before exchanging data with the attested party.

Session resumption presents special challenges since it happens at the TLS level, which is not aware of the application-level Authenticator. The application (or the modified TLS library) must ensure that a resumed session has already completed remote attestation before the session can be used normally, and race conditions are possible.

One possible approach to avoid race conditions is as follows:

- * For any TLS handshake in which remote attestation is required, the TLS client ensures that any NewSessionTicket messages received from the TLS server are ignored or discarded.
- * The client and server perform remote attestation over the established TLS connection.
- * For subsequent TLS connections, the client applies the same policy: if remote attestation is required for a connection before any application traffic is exchanged, PSK-based resumption is prevented.

From a TLS handshake perspective this is possible.

7.3. Timing for Remote Attestation

Remote attestation **MUST** be completed before sending any application data to the peer. For use cases that require only a one-time attestation for the lifetime of a TLS connection, remote attestation can be performed immediately after the TLS handshake completes.

7.4. Evidence Freshness

The Evidence carried in `cmw_attestation` does not require an additional freshness mechanism (such as a nonce [RA-TLS] or a timestamp). Freshness is already ensured by the exporter value derived using the `certificate_request_context`, as described in Section 5. Because this value is bound to the active TLS connection, the Evidence is guaranteed to be fresh for the connection in which it is generated.

The Evidence presented in this protocol is valid only at the time it is generated and presented. To ensure that the attested peer continues to operate in a secure state, remote attestation may be re-initiated periodically. In this protocol, this can be accomplished by initiating a new Exported-Authenticatorbased post-handshake authentication exchange, which results in a new `certificate_request_context` and therefore a newly derived exporter value to maintain freshness.

8. Privacy Considerations

8.1. Client as Attester

In this section, we are assuming that the Attester is a TLS client, representing an individual person. We are concerned about the potential leakage of privacy-sensitive information about that person, such as the correlation of different connections initiated by them.

In background-check model, the Verifier not only has access to detailed information about the Attester's TCB through Evidence, but it also knows the exact time and the party (i.e., the RP) with whom the secure channel establishment is attempted [RA-TLS]. The privacy implications are similar to OCSP [RFC6960]. While the RP may trust the Verifier not to disclose any information it receives, the same cannot be assumed for the Attester, which generally has no prior relationship with the Verifier. Some ways to address this include:

- * Attester-side redaction of privacy-sensitive evidence claims,
- * Using selective disclosure (e.g., SD-JWT [I-D.ietf-oauth-selective-disclosure-jwt] with EAT [RFC9711]),
- * Co-locating the Verifier role with the RP,
- * Utilizing privacy-preserving attestation schemes (e.g., DAA [I-D.ietf-rats-daa]), or

- * Utilizing Attesters manufactured with group identities (e.g., Requirement 4.1 of [FIDO-REQS]).

The last two also have the property of hiding the peer's identity from the RP.

Note that the equivalent of OCSP "stapling" involves using a passport topology where the Verifier's involvement is unrelated to the TLS connection.

8.2. Server as Attester

For the case of the TLS server as the Attester, the server can ask for client authentication and only send the Evidence after successful client authentication. This limits the exposure of server's hardware-level Claims to be revealed only to authorized clients.

9. IANA Considerations

// Note to RFC Editor: in this section, please replace RFCthis with the RFC number assigned to this document and remove this note.

9.1. TLS Extension Type Registration

IANA is requested to register the following new extension type in the "TLS ExtensionType Values" registry [IANA.tls-extensiontype-values]:

Value	Extension Name	TLS	DTLS-Only	Recommended	Reference
		1.3			
TBD	cmw_attestation	CT	N	Yes	RFCthis

Table 1

9.2. TLS Flags Extension Registry

IANA is requested to add the following entry to the "TLS Flags" extension registry established by [I-D.ietf-tls-tlsflags]:

- * Value: TBD1
- * Flag Name: CMW_Attestation
- * Messages: CH, EE
- * Recommended: Y

* Reference: RFCthis

10. References

10.1. Normative References

[I-D.ietf-rats-msg-wrap]

Birkholz, H., Smith, N., Fossati, T., Tschofenig, H., and D. Glaze, "RATS Conceptual Messages Wrapper (CMW)", Work in Progress, Internet-Draft, draft-ietf-rats-msg-wrap-23, 11 December 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-rats-msg-wrap-23>>.

[I-D.ietf-tls-rfc8446bis]

Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-rfc8446bis-14, 13 September 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-tls-rfc8446bis-14>>.

[I-D.ietf-tls-tlsflags]

Nir, Y., "A Flags Extension for TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-tlsflags-16, 14 September 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-tls-tlsflags-16>>.

[IANA.tls-extensiontype-values]

IANA, "Transport Layer Security (TLS) Extensions", <<https://www.iana.org/assignments/tls-extensiontype-values>>.

[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>>.

[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>>.

- [RFC9261] Sullivan, N., "Exported Authenticators in TLS", RFC 9261, DOI 10.17487/RFC9261, July 2022, <<https://www.rfc-editor.org/rfc/rfc9261>>.
- [RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedures (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <<https://www.rfc-editor.org/rfc/rfc9334>>.

10.2. Informative References

- [FIDO-REQS] Peirani, B. and J. Verrept, "FIDO Authenticator Security and Privacy Requirements", March 2025, <<https://fidoalliance.org/specs/fido-security-requirements/>>.
- [I-D.fossati-tls-attestation] Tschofenig, H., Sheffer, Y., Howard, P., Mihalcea, I., Deshpande, Y., Niemi, A., and T. Fossati, "Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", Work in Progress, Internet-Draft, draft-fossati-tls-attestation-09, 30 April 2025, <<https://datatracker.ietf.org/doc/html/draft-fossati-tls-attestation-09>>.
- [I-D.ietf-lamps-csr-attestation] Ounsworth, M., Tschofenig, H., Birkholz, H., Wiseman, M., and N. Smith, "Use of Remote Attestation with Certification Signing Requests", Work in Progress, Internet-Draft, draft-ietf-lamps-csr-attestation-22, 11 February 2026, <<https://datatracker.ietf.org/doc/html/draft-ietf-lamps-csr-attestation-22>>.
- [I-D.ietf-oauth-selective-disclosure-jwt] Fett, D., Yasuda, K., and B. Campbell, "Selective Disclosure for JWTs (SD-JWT)", Work in Progress, Internet-Draft, draft-ietf-oauth-selective-disclosure-jwt-22, 29 May 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-oauth-selective-disclosure-jwt-22>>.
- [I-D.ietf-rats-daa] Birkholz, H., Newton, C., Chen, L., Giannetsos, T., and D. Thaler, "Direct Anonymous Attestation for the Remote Attestation Procedures Architecture", Work in Progress, Internet-Draft, draft-ietf-rats-daa-08, 3 September 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-rats-daa-08>>.

[ID-Crisis]

Sardar, M. U., Moustafa, M., and T. Aura, "Identity Crisis in Confidential Computing: Formal Analysis of Attested TLS", November 2025, <https://www.researchgate.net/publication/398839141_Identity_Crisis_in_Confidential_Computing_Formal_Analysis_of_Attested_TLS>.

[RA-TLS]

Sardar, M. U., Niemi, A., Tschofenig, H., and T. Fossati, "Towards Validation of TLS 1.3 Formal Model and Vulnerabilities in Intel's RA-TLS Protocol", November 2024, <<https://ieeexplore.ieee.org/document/10752524>>.

[RelayAttacks]

Sardar, M. U., "Relay Attacks in Intra-handshake Attestation for Confidential Agentic AI Systems", November 2025, <https://mailarchive.ietf.org/arch/msg/seat/x3eQxFjQFJLceae6l4_NgXnmsDY/>.

[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/rfc/rfc6960>>.

[RFC9711]

Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <<https://www.rfc-editor.org/rfc/rfc9711>>.

Acknowledgements

We would like to thank Chris Patton for his proposal to explore RFC 9261 for attested TLS. We would also like to thank Eric Rescorla, Paul Howard, and Yogesh Deshpande for their input.

Appendix

Post-handshake vs. Intra-handshake Privacy

From the view of the TLS server, post-handshake attestation offers better privacy than intra-handshake attestation when the server acts as the Attester. In intra-handshake attestation, due to the inherent asymmetry of the TLS protocol, a malicious TLS client could potentially retrieve sensitive information from the Evidence without the client's trustworthiness first being established by the server. In post-handshake attestation, the server can ask for client authentication and only send the Evidence after successful client authentication.

Post-handshake vs. Intra-handshake Security

Intra-handshake attestation proposal [I-D.fossati-tls-attestation] is vulnerable to diversion attacks [ID-Crisis]. It also does not bind the Evidence to the application traffic secrets, resulting in relay attacks [RelayAttacks]. Formal analysis of post-handshake attestation is a work-in-progress.

Document History

-00

- * Expanded security considerations, in particular added security guarantees
- * Added privacy considerations
- * Corrected Figure 1

-01

- * Added channel binding
- * Added security analysis of intra-handshake attestation in Appendix

-02

- * Security considerations: Race conditions
- * Security considerations: Timing of remote attestation
- * Updated Client as Attester
- * Added Server as Attester
- * Added operational overview

Authors' Addresses

Muhammad Usama Sardar
TU Dresden
Email: muhammad_usama.sardar@tu-dresden.de

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

Tirumaleswar Reddy
Nokia
Bangalore
Karnataka
India
Email: k.tirumaleswar_reddy@nokia.com

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

Hannes Tschofenig
University of Applied Sciences Bonn-Rhein-Sieg
Germany
Email: Hannes.Tschofenig@gmx.net

Ionut Mihalcea
Arm Limited
Email: ionut.mihalcea@arm.com