LAKE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 21 March 2026

E. Lopez-Perez
Inria
G. Selander
J. Preu Mattsson
Ericsson
R. Marin-Lopez
F. Lopez-Gomez
University of Murcia

17 September 2025

EDHOC Authenticated with Pre-Shared Keys (PSK)
draft-ietf-lake-edhoc-psk-05

Abstract

This document specifies a Pre-Shared Key (PSK) authentication method for the Ephemeral Diffie-Hellman Over COSE (EDHOC) key exchange protocol. The PSK method enhances computational efficiency while providing mutual authentication, ephemeral key exchange, identity protection, and quantum resistance. It is particularly suited for systems where nodes share a PSK provided out-of-band (external PSK) and enables efficient session resumption with less computational overhead when the PSK is provided from a previous EDHOC session (resumption PSK). This document details the PSK method flow, key derivation changes, message formatting, processing, and security considerations.

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://lake-wg.github.io/psk/#go.draft-ietf-lake-edhoc-psk.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc-psk/.

Discussion of this document takes place on the LAKE Working Group mailing list (mailto:lake@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/lake/. Subscribe at https://www.ietf.org/mailman/listinfo/lake/.

Source for this draft and an issue tracker can be found at https://github.com/lake-wg/psk.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a Pre-Shared Key (PSK) authentication method for the Ephemeral Diffie-Hellman Over COSE (EDHOC) key exchange protocol [RFC9528]. The PSK method balances the complexity of credential distribution with computational efficiency. While symmetric key distribution is more complex than asymmetric approaches, PSK authentication offers greater computational efficiency compared to the methods outlined in [RFC9528]. The PSK method retains mutual authentication, asymmetric ephemeral key exchange, and identity protection established by [RFC9528].

EDHOC with PSK authentication benefits use cases where two nodes share a Pre-Shared Key (PSK) provided out-of-band (external PSK). Examples include the Authenticated Key Management Architecture (AKMA) in mobile systems or the Peer and Authenticator in Extensible Authentication Protocol (EAP) systems. The PSK method enables the nodes to perform ephemeral key exchange, achieving Perfect Forward Secrecy (PFS). This ensures that even if the PSK is compromised, past communications remain secure against active attackers, while future communications are protected against passive attackers. Additionally, by leveraging the PSK for both authentication and key derivation, the method provides quantum-resistant key exchange and authentication even when used with ECDHE.

Another important use case of PSK authentication in the EDHOC protocol is session resumption. This allows previously connected parties to quickly reestablish secure communication using pre-shared keys from a prior session, reducing the overhead associated with key exchange and asymmetric authentication. By using PSK authentication, EDHOC allows session keys to be refreshed with significantly lower computational overhead compared to public-key authentication. In this case, the resumption PSK is provisioned after the establishment of a previous EDHOC session by using EDHOC_Exporter. Thus, the external PSK serves as a long-term credential while the resumption PSK acts as a session key.

Section 3 provides an overview of the PSK method flow and credentials. Section 4 outlines the changes to key derivation compared to [RFC9528]. Section 5 details message formatting and processing, and Section 6 describes the usage of PSK for resumption. Section 7 defines the use of EDHOC-PSK with OSCORE. Security considerations are described in Section 8, and Section 9 outlines the IANA considerations.

2. Conventions and Definitions

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.

Readers are expected to be familiar with the terms and concepts described in EDHOC [RFC9528], CBOR [RFC8949], CBOR Sequences [RFC8742], COSE Structures and Processing [RFC9052], COSE Algorithms [RFC9053], CWT and CCS [RFC8392], and the Concise Data Definition Language (CDDL) [RFC8610], which is used to express CBOR data structures.

3. Protocol

This document specifies a new EDHOC authentication method (see Section 3.2 of [RFC9528]) referred to as the Pre-Shared Key method (EDHOC-PSK). This method shares some features with, and differs in other respects from, the authentication methods previously defined in EDHOC.

Authentication is based on a Pre-Shared Key (PSK) shared between the Initiator and the Responder. As in the methods defined in [RFC9528], CRED_I and CRED_R are authentication credentials containing identifying information for the Initiator and Responder, respectively. However, unlike those methods, there is a single shared authentication credential identifier, ID_CRED_PSK, which the Responder uses to retrieve the PSK and the associated authentication credentials.

3.1. Credentials

The Initiator and Responder are assumed to share a PSK (either an external PSK or a resumption PSK) with high entropy that meets the following requirements:

* Only the Initiator and the Responder have access to the PSK.

* The Responder can retrieve the PSK, CRED_I, and CRED_R, using ID_CRED_PSK.

3.1.1. ID_CRED_PSK

ID_CRED_PSK is a COSE header map containing header parameters that can identify a pre-shared key. For example:

ID_CRED_PSK = {4 : h'0f' }; 4 = 'kid'

The purpose of ID_CRED_PSK is to facilitate retrieval of the correct PSK. While ID_CRED_PSK use encoding and representation patterns from [RFC9528], it differs fundamentally in that it identifies a symmetric key rather than a public authentication key.

It is RECOMMENDED that ID_CRED_PSK uniquely or stochastically identifies the corresponding PSK. Uniqueness avoids ambiguity that could require the recipient to try multiple keys, while stochasticity reduces the risk of identifier collisions and supports stateless processing. These properties align with the requirements for rKID in session resumption.

3.1.2. CRED_I and CRED_R

CRED_I and CRED_R are authentication credentials associated with the PSK. The notation CRED_x refers to either CRED_I or CRED_R. Authentication is achieved implicitly through the successful use of the PSK to derive keying material, and to encrypt and subsequently decrypt protected messages.

When using an external PSK, a common representation of CRED_I and CRED_R is a CBOR Web Token (CWT) or CWT Claims Set (CCS) [RFC8392], where the 'cnf' claim includes the confirmation method COSE_Key. An example of CRED_I and CRED_R is shown below:

```
{                                                  /CCS/
  2 : "42-50-31-FF-EF-37-32-39",                   /sub/
  8 : {                                            /cnf/
    1 : {                                          /COSE_Key/
       1 : 4,                                      /kty/
       2 : h'0f',                                  /kid/
    }
  }
}

{                                                  /CCS/
  2 : "23-11-58-AA-B3-7F-10",                      /sub/
  8 : {                                            /cnf/
    1 : {                                          /COSE_Key/
       1 : 4,                                      /kty/
       2 : h'0f',                                  /kid/
    }
  }
}
```

Alternative formats for CRED_I and CRED_R MAY be used. When a resumption PSK is employed, CRED_I and CRED_R MUST be the same credentials used in the initial EDHOC exchange, for example, public-key credentials such as X.509 certificates.

Implementations MUST ensure that CRED_I and CRED_R are distinct, for example by including different identities in their sub-claims (e.g., "42-50-31-FF-EF-37-32-39" and "23-11-58-AA-B3-7F-10"). Ensuring distinct credentials simplifies correct party identification and prevents reflection and misbinding attacks, as described in Appendix D.2 of [RFC9528].

3.1.3. Encoding and processing guidelines

The following guidelines apply to the encoding and handling of CRED_x and ID_CRED_PSK. Requirements on CRED_x applies both to CRED_I and to CRED_R.

* If CRED_x is CBOR-encoded, it SHOULD use deterministic encoding as specified in Sections 4.2.1 and 4.2.2. of [RFC8949]. Deterministic encoding ensures consistent identification and avoids interoperability issues caused by non-deterministic CBOR variants.

* If CRED_x is provisioned out-of-band and transported by value, it SHOULD be used as received without re-encoding. Re-encoding can cause mismatches when comparing identifiers such as hash values or 'kid' references.

* ID_CRED_PSK SHOULD uniquely identify the corresponding PSK to avoid ambiguity. When ID_CRED_PSK contains a key identifier, care must be taken to ensure that 'kid' is unique for the PSK.

* When ID_CRED_PSK consists solely of a 'kid' parameter (i.e., { 4 : kid }), the compact encoding optimization defined in Section 3.5.3.2 of [RFC9528] MUST be applied in plaintext fields (such as PLAINTEXT_3A). For example:

  - { 4 : h'0f' } encoded as h'0f' (CBOR byte string)

  - { 4 : 21 } encoded as 0x15 (CBOR integer)

  These optimizations MUST NOT be applied in COSE header parameters or in other contexts where the full map structure is required.

* To mitigate misbinding attacks, identity information such as a 'sub' (subject) claim MUST be included in both CRED_I and CRED_R.

3.2. Message Flow of EDHOC-PSK

The message flow of EDHOC-PSK follows the structure defined in [RFC9528], with authentication based on symmetric keys rather than public keys. For identity protection, credential-related message fields appear first in message_3.

ID_CRED_PSK is encrypted using a key derived from a shared secret obtained through the first two messages. If Diffie-Hellman key exchange is used, G_X and G_Y are the ephemeral public keys, and the shared secret G_XY is the DH shared secret, as in [RFC9528]. If the Diffie-Hellman procedure is replaced by a KEM, then G_X and G_Y are encapsulation key and ciphertext, respectively, and the shared secret G_XY is derived by the KEM, see [I-D.spm-lake-pqsuites].

The Responder authenticates the Initiator first. Figure 1 illustrates the message flow of the EDHOC-PSK authentication method.

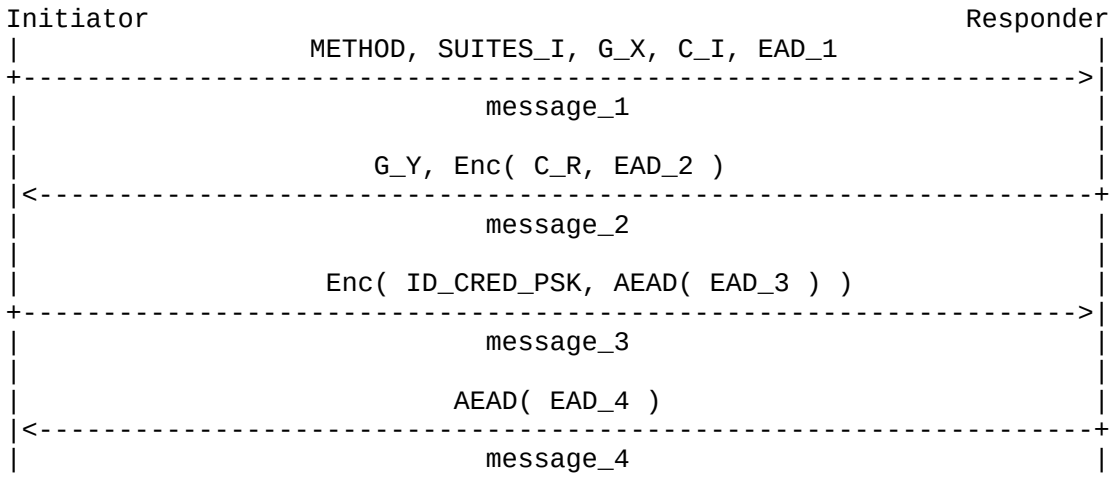

Figure 1: Overview of Message Flow of EDHOC-PSK.

This approach provides identity protection against passive attackers for both Initiator and Responder. EDHOC message_4 remains OPTIONAL, but is needed to authenticate the Responder and achieve mutual authentication in EDHOC when external applications (e.g., OSCORE) are not relied upon. In either case, the inclusion of a fourth message provides mutual authentication and explicit key confirmation (see Section 5.4).

4. Key Derivation

The pseudorandom keys (PRKs) used in the EDHOC-PSK authentication method are derived with EDHOC_Extract, as in [RFC9528].

PRK = EDHOC_Extract(salt, IKM)

where salt and input keying material (IKM) are defined for each key. The definition of EDHOC_Extract depends on the EDHOC hash algorithm selected in the cipher suite, see Section 4.1.1 of [RFC9528].

To maintain a uniform key schedule across all EDHOC authentication methods, the same pseudorandom key notation (PRK_2e, PRK_3e2m, and PRK_4e3m) is retained. The index notation is preserved for consistency with other EDHOC authentication variants, even though it does not fully reflect the functional role of the keys in this method; for example, no MACs are used in EDHOC-PSK.

PRK_2e is extracted as in [RFC9528] with

* salt = TH_2, and

* IKM = G_XY,

where the transcript hash TH_2 = H(G_Y, H(message_1)) is defined in Section 5.3.2 of [RFC9528].

SALT_4e3m is derived from PRK_3e2m and TH_3, as shown in Figure 6 of [RFC9528].

The other PRKs and transcript hashes are modified as specified below. Figure 2 lists the key derivations that differ from Section 4.1.2 of [RFC9528].

```
PRK_3e2m     = PRK_2e
KEYSTREAM_2A = EDHOC_KDF( PRK_2e,   0, TH_2,  plaintext_length_2a )
PRK_4e3m     = EDHOC_Extract( SALT_4e3m, PSK )
KEYSTREAM_3A = EDHOC_KDF( PRK_3e2m, 12, TH_3, plaintext_length_3a )
K_3          = EDHOC_KDF( PRK_4e3m, 3, TH_3, key_length )
IV_3         = EDHOC_KDF( PRK_4e3m, 4, TH_3, iv_length )
```

Figure 2: Key Derivation of EDHOC-PSK.

where:

* KEYSTREAM_2A is used to encrypt PLAINTEXT_2A in message_2.

  - plaintext_length_2a is the length of PLAINTEXT_2A in message_2.

* KEYSTREAM_3A is used to encrypt PLAINTEXT_3A (the concatenation of ID_CRED_PSK and CIPHERTEXT_3B) in message_3.

  - plaintext_length_3a is the length of PLAINTEXT_3A in message_3.

* TH_3 = H(TH_2, PLAINTEXT_2A).

The definition of the transcript hash TH_4 is modified as follows:

* TH_4 = H(TH_3, ID_CRED_PSK, PLAINTEXT_3B, CRED_I, CRED_R)

5. Message Formatting and Processing

This section specifies the differences in message formatting and processing compared to Section 5 of [RFC9528]. Note that if any processing step fails, then the Responder MUST send an EDHOC error message back as defined in Section 6 of [RFC9528], and the EDHOC session MUST be aborted.

5.1. Message 1

Message 1 is formatted and processed as specified in Section 5.2 of [RFC9528].

5.2. Message 2

5.2.1. Formatting of Message 2

Message 2 is formatted as specified in Section 5.3.1 of [RFC9528].

5.2.2. Responder Composition of Message 2

CIPHERTEXT_2A is calculated with a binary additive stream cipher, using a keystream generated with EDHOC_Expand, and the following plaintext:

* PLAINTEXT_2A = (C_R, ? EAD_2)

* CIPHERTEXT_2A = PLAINTEXT_2A XOR KEYSTREAM_2A

C_R, EAD_2 are defined in Section 5.3.2 of [RFC9528]. In contrast to [RFC9528], ID_CRED_R, MAC_2, and Signature_or_MAC_2 are not included in message_2. This omission is the primary difference from the signature and MAC-based authentication methods defined in [RFC9528], as authentication in EDHOC-PSK relies solely on the shared PSK and the successful decryption of protected messages. KEYSTREAM_2A is defined in Section 4.

5.2.3. Initiator Processing of Message 2

Upon receiving message_2, the Initiator processes it as follows:

* Compute KEYSTREAM_2A as defined in Section 4.

* Decrypt CIPHERTEXT_2A using binary XOR, i.e., PLAINTEXT_2A = CIPHERTEXT_2A XOR KEYSTREAM_2A

In contrast to Section 5.3.3 of [RFC9528], ID_CRED_R is not made available to the application in step 4, and steps 5 and 6 are skipped.

5.3. Message 3

5.3.1. Formatting of Message 3

Message 3 is formatted as specified in Section 5.4.1 of [RFC9528].

5.3.2. Initiator Composition of Message 3

* CIPHERTEXT_3A is computed using a binary additive stream cipher with a keystream generated by EDHOC_Expand, applied to the following plaintext:
  - PLAINTEXT_3A = (ID_CRED_PSK / bstr / -24..23, CIPHERTEXT_3B)
    - o If ID_CRED_PSK contains a single 'kid' parameter, i.e., ID_CRED_PSK = { 4 : kid_PSK }, then the compact encoding is applied, see Section 3.5.3.2 of [RFC9528].
    - o For the case of plaintext_length exceeding the EDHOC_KDF output size, see Appendix G of [RFC9528].
  - Compute KEYSTREAM_3A as in Section 4.
  - CIPHERTEXT_3A = PLAINTEXT_3A XOR KEYSTREAM_3A
* CIPHERTEXT_3B is the 'ciphertext' of COSE_Encrypt0 object as defined in Section 5.2 and Section 5.3 of [RFC9528], with the EDHOC AEAD algorithm of the selected cipher suite, using the encryption key K_3, the initialization vector IV_3 (if used by the AEAD algorithm), the parameters described in Section 5.2 of [RFC9528], plaintext PLAINTEXT_3B and the following parameters as input:
  - protected = h''
  - external_aad = << ID_CRED_PSK, TH_3, CRED_I, CRED_R >>
  - K_3 and IV_3 as defined in Section 4
  - PLAINTEXT_3B = (? EAD_3)

The Initiator computes TH_4 as defined in Section 4.

There is no need for MAC_3 or signature, since AEAD's built-in integrity and the use of PSK-based key derivation provides implicit authentication of the Initiator.

5.3.3. Responder Processing of Message 3

Upon receiving message_3, the Responder proceeds as follows:

* Derive K_3 and IV_3 as defined in Section 4.

* Parse the structure of message_3, which consists of a stream-cipher encrypted structure, CIPHERTEXT_3A = PLAINTEXT_3A XOR KEYSTREAM_3A, where PLAINTEXT_3A = (ID_CRED_PSK, CIPHERTEXT_3B) and CIPHERTEXT_3B is the inner AEAD-encrypted object.

* Generate KEYSTREAM_3A with the same method the Initiator used.

* Decrypt CIPHERTEXT_3A using binary XOR with KEYSTREAM_3A to recover PLAINTEXT_3A.

* Use ID_CRED_PSK to identify the authentication credentials and retrieve PSK.

* AEAD-decrypt CIPHERTEXT_3B using:

  - K_3, IV_3

  - external_aad = << ID_CRED_PSK, TH_3, CRED_I, CRED_R >>

  - protected = h''

  - AEAD algorithm from cipher suite

If AEAD verification fails, this indicates a processing problem or that the message was tampered with. If it succeeds, the Responder concludes that the Initiator possesses the PSK, correctly derived TH_3, and is actively participating in the protocol.

Finally, the Responder computes TH_4 as defined in Section 4.

No MAC_3 or signature is needed, as the AEAD tag guarantees both integrity and authenticity in this symmetric setting.

5.4. Message 4

Message 4 is formatted and processed as specified in Section 5.5 of [RFC9528].

After successfully verifying message_4, or another fourth message from the Responder protected with an exported application key such as an OSCORE message, the Initiator is assured that the Responder has derived PRK_out (key confirmation) and that no other party can derive this key.

The Initiator MUST NOT persistently store PRK_out or application keys until it has successfully verified such a fourth message and the application has authenticated the Responder.

Compared to [RFC9528], the fourth message not only provides key confirmation but also authenticates the Responder. For mutual authentication a fourth message is therefore mandatory.

6. PSK usage for Session Resumption

This section specifies how EDHOC-PSK is used for session resumption in EDHOC. The EDHOC_Exporter, as defined in Section 4.2 of [RFC9528], is used to derive the resumption parameters rPSK and rKID:

```
rPSK         = EDHOC_Exporter( TBD2, h'', resumption_psk_length )
rKID         = EDHOC_Exporter( TBD3, h'', id_cred_psk_length )
rID_CRED_PSK = { 4 : rKID }
```

Figure 3: Resumption Parameters.

where:

* resumption_psk_length defaults to the key_length, i.e., the length of the encryption key of the EDHOC AEAD algorithm in the selected cipher suite of the session where the EDHOC_Exporter is invoked.

* id_cred_psk_length defaults to 2 bytes.

A peer that has successfully completed an EDHOC session, regardless of the authentication method used or whether the session was a PSK resumption, MUST generate a resumption key for the next resumption within the current "session series", provided that PSK resumption is supported.

To ensure both peers share the same resumption key, when a resumption session is run using rPSK_i as the resumption key:

* The Responder MAY delete the previous resumption key rPSK_(i-1), if present, after successfully verifying message_3. At that point the Responder can be certain that the Initiator has access to the current resumption key rPSK_i.

* The Initiator MAY delete rPSK_i after successfully verifying the fourth message. At that point, the Initiator can be certain that the Responder already has derived the next resumption key, rPSK_(i+1).

* The Responder MAY delete rPSK_i after successfully verifying a fifth message from the Initiator protected with an exported application key such as an OSCORE message, if present. At that point, the Initiator can be certain that the Responder already has derived the next resumption key, rPSK_(i+1).

6.1. Cipher Suite Requirements for Resumption

When using a resumption PSK derived from a previous EDHOC exchange:

1. The resumption PSK MUST only be used with the same cipher suite from which it was derived, or with a cipher suite that provides stronger security guarantees.

2. Implementations MUST maintain a mapping between each resumption PSK and its originating cipher suite to enforce this requirement.

3. If a resumption PSK is offered with a cipher suite that provides weaker security, the Responder MUST reject the ongoing EDHOC session.

6.2. Privacy Considerations for Resumption

When using resumption PSKs:

* ID_CRED_PSK is not exposed to passive attackers, and under normal operation it is not reused. Reuse of the same ID_CRED_PSK can occur due to transmission errors or when a peer loses its stored resumption key. An active attacker can obtain the value of ID_CRED_PSK and force its reuse. This aligns with the security goals of EDHOC-PSK, which are to provide identity protection against passive attackers, but not against active attackers.

6.3. Security Considerations for Resumption

* Resumption PSKs MUST NOT be used for purposes other than EDHOC session resumption.

* Resumption PSKs MUST be securely stored with the same level of protection as the session keys.

* Parties SHOULD prevent excessive reuse of the same resumption PSK.

7. EDHOC-PSK and Extensible Authentication Protocol (EAP)

EDHOC with PSK authentication has several important use cases within the Extensible Authentication Protocol (EAP).

One use case is the resumption of a session established with the EAP method EAP-EDHOC [I-D.ietf-emu-eap-edhoc], regardless of the EDHOC-based authentication method originally used in that session. This is similar to the resumption mechanism in EAP-TLS 1.3 [RFC9190]. Resumption reduces the number of round trips and allows the EAP-EDHOC server to avoid database lookups that might be required during an initial handshake. If the server accepts resumption, the resumed session is considered authenticated and securely bound to the prior authentication or resumption.

The use of resumption with EAP-EDHOC is optional for the peer, but it is RECOMMENDED whenever a valid rPSK is available. On the server side, resumption acceptance is also optional, but it is RECOMMENDED if the rPSK remains valid. The server may, however, require a new initial handshake by refusing resumption. It is further RECOMMENDED to use Network Access Identifiers (NAIs) with the same realm in the identity response during both the full handshake and resumption. For example, the NAI @realm can safely be reused since it does not expose information that links a user' s resumption attempt with the original full handshake.

EAP-EDHOC-PSK also provides a significant improvement over EAP-PSK [RFC4764], which lacks support for identity protection, cryptographic agility, and ephemeral key exchange, now considered essential for meeting current security requirements. Without perfect forward secrecy, compromise of the PSK enables a passive attacker to decrypt both past and future sessions. Note that PSK authentication is not allowed in EAP-TLS [RFC9190].

8. EDHOC-PSK and OSCORE

Before sending message_3 the Initiator can derive PRK_out and create an OSCORE-protected request. The request payload MAY convey both an EDHOC message_3 and OSCORE-protected data combined together, as described in Section 3 of [RFC9668].

9. Security Considerations

The EDHOC-PSK authentication method introduces deviations from the initial specification of EDHOC [RFC9528]. This section analyzes the security implications of these changes.

9.1. Identity Protection

In EDHOC-PSK, ID_CRED_PSK in message_3 is encrypted with a keystream derived from the ephemeral shared secret G_XY. As a result, unlike the asymmetric authentication methods in Section 9.1 of [RFC9528], which protect the Initiator's identity against active attackers and the Responder's identity against passive attackers, EDHOC-PSK protects both the Initiator and the Responder identities against passive attackers.

9.2. Mutual Authentication

EDHOC-PSK provides mutual authentication and explicit key confirmation through an additional message that demonstrates possession of the PSK. This may be the optional message_4 or an application message (e.g., an OSCORE message) protected with a key derived from EDHOC.

To mitigate reflection or Selfie attacks, the identities in CRED_I and CRED_R MUST be distinct.

9.3. Protection of External Authorization Data (EAD)

As in [RFC9528], EDHOC-PSK ensures the confidentiality and integrity of External Authorization Data (EAD). The security guarantees for EAD fields remain unchanged from the original EDHOC specification.

9.4. Post Quantum Considerations

Advances in quantum computing suggest that a Cryptographically Relevant Quantum Computer (CRQC) may eventually be realized. Such a machine would render many asymmetric algorithms, including Elliptic Curve Diffie-Hellman (ECDH), insecure.

EDHOC-PSK derives authentication and session keys primarily from a symmetric PSK, which provides quantum resistance even when combined with ECDHE. However, if a CRQC is realized, the ECDHE contribution degenerates to providing only randomness. In that case, EDHOC-PSK with ECDHE offers neither identity protection nor Perfect Forward Secrecy (PFS) against quantum adversaries. Moreover, if the PSK is compromised, a passive quantum attacker could decrypt both past and future sessions.

By contrast, combining EDHOC-PSK with a quantum-resistant Key Encapsulation Mechanism (KEM), such as ML-KEM, ensures both identity protection and PFS even against quantum-capable attackers. Future EDHOC cipher suites incorporating ML-KEM are expected to be registered; see [I-D.spm-lake-pqsuites].

9.5. Independence of Session Keys

NIST requires that an ephemeral private key be used in only one key-establishment transaction ([SP-800-56A], Section 5.6.3.3). This requirement preserves session key independence and forward secrecy, and EDHOC-PSK complies with it.

In other protocols, reuse of ephemeral keys, especially when combined with missing public key validation, has led to severe vulnerabilities, enabling attackers to recover “ephemeral” private keys and compromise both past and future sessions between two legitimate parties. Assuming breach and minimizing the impact of compromise are fundamental principles of zero-trust security.

9.6. Unified Approach and Recommendations

For use cases where application data is transmitted, it can be sent together with message_3, maintaining efficiency. In applications such as EAP-EDHOC [I-D.ietf-emu-eap-edhoc], where no application data is exchanged between Initiator and Responder, message_4 is mandatory. In such cases, EDHOC-PSK does not increase the total number of messages. Other implementations may replace message_4 with an OSCORE-protected message. In this case, the following requirement applies: The Initiator SHALL NOT persistently store PRK_out or derived application keys until successfully verifying message_4 or a message protected with an exported application key (e.g., an OSCORE message). This ensures that key material is stored only after its authenticity is confirmed, thereby strengthening privacy by preventing premature storage of potentially compromised keys. Finally, the order of authentication (i.e., whether the Initiator or the Responder authenticates first) is not relevant in EDHOC-PSK. While this ordering affects privacy properties in the asymmetric methods of [RFC9528], it has no significant impact in EDHOC-PSK.

10. IANA Considerations

This document requires the following IANA actions.

10.1. EDHOC Method Type Registry

IANA is requested to register the following entry in the "EDHOC Method Type" registry under the group name "Ephemeral Diffie-Hellman Over COSE (EDHOC)".

Value	Initiator Authentication Key	Responder Authentication Key
TBD4	PSK	PSK

Table 1: Addition to the EDHOC Method Type Registry.

NOTE: Suggested value: TBD4 = 4. RFC Editor: Remove this note.

10.2. EDHOC Exporter Label Registry

IANA is requested to register the following entry in the "EDHOC Exporter Label" registry under the group name "Ephemeral Diffie-Hellman Over COSE (EDHOC)".

Label	Description	Change Controller	Reference
TBD2	Resumption PSK	IETF	Section 7
TBD3	Resumption kid	IETF	Section 7

Table 2: Additions to the EDHOC Exporter Label Registry.

NOTE: Suggested values: TBD2 = 2, TBD3 = 3. RFC Editor: Remove this note.

11. References

11.1. Normative References

[I-D.ietf-emu-eap-edhoc]
Garcia-Carrillo, D., Marin-Lopez, R., Selander, G., Mattsson, J. P., and F. Lopez-Gomez, "Using the Extensible Authentication Protocol (EAP) with Ephemeral Diffie-Hellman over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-emu-eap-edhoc-05, 30 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-emu-eap-edhoc-05>.

[I-D.spm-lake-pqsuites]
Selander, G. and J. P. Mattsson, "Quantum-Resistant Cipher Suites for EDHOC", Work in Progress, Internet-Draft, draft-spm-lake-pqsuites-00, 6 July 2025, <https://datatracker.ietf.org/doc/html/draft-spm-lake-pqsuites-00>.

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

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

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/rfc/rfc8392>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/rfc/rfc8742>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

[RFC9528] Selander, G., Preu Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, DOI 10.17487/RFC9528, March 2024, <https://www.rfc-editor.org/rfc/rfc9528>.

[RFC9668] Palombini, F., Tiloca, M., Hglund, R., Hristozov, S., and G. Selander, "Using Ephemeral Diffie-Hellman Over COSE (EDHOC) with the Constrained Application Protocol (CoAP) and Object Security for Constrained RESTful Environments (OSCORE)", RFC 9668, DOI 10.17487/RFC9668, November 2024, <https://www.rfc-editor.org/rfc/rfc9668>.

[SP-800-56A] Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography", NIST Special Publication 800-56A Revision 3, April 2018, <https://doi.org/10.6028/NIST.SP.800-56Ar3>.

11.2. Informative References

[RFC4764] Bersani, F. and H. Tschofenig, "The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method", RFC 4764, DOI 10.17487/RFC4764, January 2007, <https://www.rfc-editor.org/rfc/rfc4764>.

[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>.

Appendix A. CDDL Definitions

This section compiles the CDDL definitions for convenience, incorporating errata filed against [RFC9528].

```
suites = [ 2* int ] / int

ead = (
  ead_label : int,
  ? ead_value : bstr,
)

EAD_1 = (1* ead)
EAD_2 = (1* ead)
EAD_3 = (1* ead)
EAD_4 = (1* ead)

message_1 = (
  METHOD : int,
  SUITES_I : suites,
  G_X : bstr,
  C_I : bstr / -24..23,
```

```
    ? EAD_1,
  )

  message_2 = (
    G_Y_CIPHERTEXT_2 : bstr,
  )

  PLAINTEXT_2A = (
    C_R : bstr / -24..23,
    ? EAD_2,
  )

  message_3 = (
    CIPHERTEXT_3 : bstr,
  )

  PLAINTEXT_3A = (
    ID_CRED_PSK : header_map / bstr / -24..23,
    CIPHERTEXT_3B : bstr,
  )

  PLAINTEXT_3B = (
    ? EAD_3
  )

  message_4 = (
    CIPHERTEXT_4 : bstr,
  )

  PLAINTEXT_4 = (
    ? EAD_4,
  )

  error = (
    ERR_CODE : int,
    ERR_INFO : any,
  )

  info = (
    info_label : int,
    context : bstr,
    length : uint,
  )
```

Appendix B. Test Vectors

B.1. message_1

Both endpoints are authenticated with Pre-Shared Keys (METHOD = 4)

NOTE: Assuming TBD4 = 4, to be confirmed by IANA. RFC Editor: Remove this note.

METHOD (CBOR Data Item) (1 byte)
04

The initiator selects cipher suite 02. A single cipher suite is encoded as an int:

SUITES_I (CBOR Data Item) (1 byte)
02

The Initiator creates an ephemeral key pair for use with the EDHOC key exchange algorithm:

Initiator’s ephemeral private key
X (Raw Value) (32 bytes)
09 97 2D FE F1 EA AB 92 6E C9 6E 80 05 FE D2 9F
70 FF BF 4E 36 1C 3A 06 1A 7A CD B5 17 0C 10 E5

Initiator’s ephemeral public key
G_X (Raw Value) (32 bytes)
7E C6 81 02 94 06 02 AA B5 48 53 9B F4 2A 35 99
2D 95 72 49 EB 7F 18 88 40 6D 17 8A 04 C9 12 DB

The Initiator selects its connection identifier C_I to be the byte string 0xA, which is encoded as 0xA since it is represented by the 1-byte CBOR int 10:

Connection identifier chosen by the Initiator
C_I (CBOR Data Item) (1 byte)
0A

No external authorization data

EAD_1 (CBOR Sequence) (0 bytes)

The Initiator constructs message_1:

message_1 (CBOR Sequence) (37 bytes)
04 02 58 20 7E C6 81 02 94 06 02 AA B5 48 53 9B
F4 2A 35 99 2D 95 72 49 EB 7F 18 88 40 6D 17 8A
04 C9 12 DB 0A

B.2. message_2

The Responder supports the most preferred and selected cipher suite 02, so SUITES_I is acceptable.

The Responder creates an ephemeral key pair for use with the EDHOC key exchange algorithm:

```
Responder’s ephemeral private key
Y (Raw Value) (32 bytes)
1E 1C 8F 2D F1 AA 71 10 B3 9F 33 BA 5E A8 DC CF
31 41 1E B3 3D 4F 9A 09 4C F6 51 92 D3 35 A7 A3

Responder’s ephemeral public key
G_Y (Raw Value) (32 bytes)
ED 15 6A 62 43 E0 AF EC 9E FB AA BC E8 42 9D 5A
D5 E4 E1 C4 32 F7 6A 6E DE 8F 79 24 7B B9 7D 83
```

The Responder selects its connection identifier C_R to be the byte string 0x05, which is encoded as 0x05 since it is represented by the 1-byte CBOR int 05:

```
Connection identifier chosen by the Responder
C_R (CBOR Data Item) (1 byte)
05
```

The transcript hash TH_2 is calculated using the EDHOC hash algorithm: TH_2 = H(G_Y, H(message_1)), where H(message_1) is:

```
H(message_1) (Raw Value) (32 bytes)
19 CC 2D 2A 95 7E DD 80 10 90 42 FD E6 CC 20 C2
4B 6A 34 BC 21 C6 D4 9F EA 89 5D 4C 75 92 34 0E

H(message_1) (CBOR Data Item) (34 bytes)
58 20 19 CC 2D 2A 95 7E DD 80 10 90 42 FD E6 CC 20
C2 4B 6A 34 BC 21 C6 D4 9F EA 89 5D 4C 75 92 34 0E

TH_2 (Raw Value) (32 bytes)
5B 48 34 AE 63 0A 8A 0E D0 B0 C6 F3 66 42 60 4D
01 64 78 C4 BC 81 87 BB 76 4D D4 0F 2B EE 3D DE

TH_2 (CBOR Data Item) (34 bytes)
58 20 5B 48 34 AE 63 0A 8A 0E D0 B0 C6 F3 66 42 60
4D 01 64 78 C4 BC 81 87 BB 76 4D D4 0F 2B EE 3D DE
```

PRK_2e is specified in Section 4.1.2 of [RFC9528]. To compute it, the Elliptic Curve Diffie-Hellman (ECDH) shared secret G_XY is needed. It is computed from G_X and Y or G_Y and X:

G_XY (Raw Value) (ECDH shared secret) (32 bytes)
2F 4A 79 9A 5A B0 C5 67 22 0C B6 72 08 E6 CF 8F
4C A5 FE 38 5D 1B 11 FD 9A 57 3D 41 60 F3 B0 B2

Then, PRK_2e is calculated as defined in Section 4.1.2 of [RFC9528]

PRK_2e (Raw Value) (32 bytes)
D0 39 D6 C3 CF 35 EC A0 CD F8 19 E3 25 79 C7 7E
1F 30 3E FC C4 36 20 50 99 48 A9 FD 47 FB D9 29

Since the Responder authenticates using PSK, PRK_3e2m = PRK_2e.

PRK_3e2m (Raw Value) (32 bytes)
D0 39 D6 C3 CF 35 EC A0 CD F8 19 E3 25 79 C7 7E
1F 30 3E FC C4 36 20 50 99 48 A9 FD 47 FB D9 29

No external authorization data:

EAD_2 (CBOR Sequence) (0 bytes)

The Responder constructs PLAINTEXT_2A:

PLAINTEXT_2A (CBOR Sequence) (1 byte)
05

The Responder computes KEYSTREAM_2 as defined in Section 4.1.2 of [RFC9528]

KEYSTREAM_2A (Raw Value) (1 byte)
EC

The Responder calculates CIPHERTEXT_2B as XOR between PLAINTEXT_2A and KEYSTREAM_2:

CIPHERTEXT_2B (CBOR Sequence) (1 byte)
E9

The Responder constructs message_2 as defined in Section 5.3.1 of [RFC9528]:

message_2 (CBOR Sequence) (35 bytes)
58 21 ED 15 6A 62 43 E0 AF EC 9E FB AA BC E8 42
9D 5A D5 E4 E1 C4 32 F7 6A 6E DE 8F 79 24 7B B9
7D 83 E9

B.3. message_3

The Initiator computes PRK_4e3m, as described in Section 4, using SALT_4e3m and PSK:

SALT_4e3m (Raw Value) (32 bytes)
ED E0 76 12 14 83 19 EB 72 59 52 71 2A 54 2C 20
97 61 0A 13 9C 4A 14 1C 8E C5 7A 5F 62 E5 E9 DD

PSK (Raw Value) (16 bytes)
50 93 0F F4 62 A7 7A 35 40 CF 54 63 25 DE A2 14

PRK_4e3m (Raw Value) (32 bytes)
C6 2C C0 4F 55 D0 08 CF EB 8A 68 1E 84 63 FD DD
A2 FF 6C A8 4B 9E D6 11 6C 86 5C D8 1E 06 24 60

The transcript hash TH_3 is calculated using the EDHOC hash algorithm:

TH_3 = H(TH_2, PLAINTEXT_2A)

TH_3 (Raw Value) (32 bytes)
38 6A 9D 05 2B 25 59 92 EE E5 FF B5 94 34 7D 32
74 18 A2 EA 51 83 48 6C 0C 9E 20 42 6E 0B CA 2F

TH_3 (CBOR Data Item) (34 bytes)
58 20 38 6A 9D 05 2B 25 59 92 EE E5 FF B5 94 34 7D
32 74 18 A2 EA 51 83 48 6C 0C 9E 20 42 6E 0B CA 2F

No external authorization data:

EAD_3 (CBOR Sequence) (0 bytes)

The Initiator constructs firstly PLAINTEXT_3B as defined in Section 5.3.1.:

PLAINTEXT_3B (CBOR Sequence) (0 bytes)

It then computes CIPHERTEXT_3B as defined in Section 5.3.2. It uses ID_CRED_PSK, CRED_I, CRED_R and TH_3 as external_aad:

ID_CRED_PSK (CBOR Data Item) (1 byte)
10

CRED_I (Raw Value) (38 bytes)
A2 02 69 69 6E 69 74 69 61 74 6F 72 08 A1 01 A3
01 04 02 41 10 20 50 50 93 0F F4 62 A7 7A 35 40
CF 54 63 25 DE A2 14

CRED_R (Raw Value) (38 bytes)
A2 02 69 72 65 73 70 6F 6E 64 65 72 08 A1 01 A3
01 04 02 41 10 20 50 50 93 0F F4 62 A7 7A 35 40
CF 54 63 25 DE A2 14

TH_3 (Raw Value) (32 bytes)
38 6A 9D 05 2B 25 59 92 EE E5 FF B5 94 34 7D 32
74 18 A2 EA 51 83 48 6C 0C 9E 20 42 6E 0B CA 2F

The Initiator computes K_3 and IV_3

K_3 (Raw Value) (16 bytes)
96 6A 57 9C EA 26 CA 3C EB 44 2A C7 27 EA B2 32

IV_3 (Raw Value) (13 bytes)
5B F1 AD 0E 4F FB 96 76 D7 8D F2 3F 6E

It then computes CIPHERTEXT_3B:

CIPHERTEXT_3B (CBOR Sequence) (9 bytes)
48 04 88 B7 F2 A6 66 B6 29

The Initiator computes KEYSTREAM_3A as defined in Section 4:

KEYSTREAM_3A (Raw Value) (10 bytes)
03 E5 D1 57 1B BC 93 32 47 1B

It then calculates PLAINTEXT_3A as stated in Section 5.3.2.:

PLAINTEXT_3A (CBOR Sequence) (10 bytes)
10 48 04 88 B7 F2 A6 66 B6 29

It then uses KEYSTREAM_3A to derive CIPHERTEXT_3A:

CIPHERTEXT_3A (CBOR Sequence) (10 bytes)
13 AD D5 DF AC 4E 35 54 F1 32

The Initiator computes message_3 as defined in Section 5.3.2.:

message_3 (CBOR Sequence) (11 bytes)
4A 13 AD D5 DF AC 4E 35 54 F1 32

The transcript hash TH_4 is calculated using the EDHOC hash algorithm: TH_4 = H(TH_3, ID_CRED_PSK, ? EAD_3, CRED_I, CRED_R)

TH_4 (Raw Value) (32 bytes)
11 48 1B 9A FE F9 5C 67 9A 52 03 82 17 EE DD 0E
0C E0 8F AA 86 5B DC 82 55 11 CA 6D C3 91 94 13

TH_4 (CBOR Data Item) (34 bytes)
58 20 11 48 1B 9A FE F9 5C 67 9A 52 03 82 17 EE DD
0E 0C E0 8F AA 86 5B DC 82 55 11 CA 6D C3 91 94 13

B.4. message_4

No external authorization data:

EAD_4 (CBOR Sequence) (0 bytes)

The Responder constructs PLAINTEXT_4:

PLAINTEXT_4 (CBOR Sequence) (0 bytes)

The Responder computes K_4 and IV_4:

K_4 (Raw Value) (16 bytes)
BC AB 1D F0 13 8D C0 5C 88 5F D3 71 E9 50 C6 7F

IV_4 (Raw Value) (13 bytes)
41 11 34 D0 E0 C5 08 D9 5D A7 C3 AC DC

The Responder computes message_4:

message_4 (CBOR Sequence) (9 bytes)
48 8A DD 93 DB 40 48 59 F9

B.5. PRK_out and PRK_exporter

After the exchange, the following PRK_out and PRK_exporter are derived by both entities:

PRK_out (Raw Value) (32 bytes)
BB A6 DE D3 B0 38 D2 32 37 74 D8 92 14 A5 13 A2
49 16 F0 42 29 6C 7C 72 9C D1 A6 7B 43 6F B4 14

PRK_exporter (Raw Value) (32 bytes)
2F CD 08 C0 C0 10 77 C6 D6 48 6B 9F 9B 67 70 20
E8 D6 8F 04 BC DC CE 71 5D D2 77 ED 25 93 1B EF

B.6. rPSK and rKID

Both peers generate a resumption key for use in the next resumption attempt, as explained in Section 6:

rPSK (Raw Value) (16 bytes)
5B 0B C7 63 F6 EA D1 7E 0E EA ED FD D3 36 A5 EE

rKID (Raw Value) (1 byte)
55

Appendix C. Change Log

RFC Editor: Please remove this appendix.

* From -04 to -05
  - Fixed misbinding attacks and resumption
  - Updated privacy considerations
  - Added EDHOC-PSK and EAP section
  - Editorial changes
  - Fixed test vectors
* From -03 to -04
  - Test Vectors
  - Editorial changes
* From -02 to -03
  - Updated abstract and Introduction
  - Changed message_3 to hide the identity length from passive attackers
  - CDDL Definitions
  - Security considerations of independence of Session Keys
  - Editorial changes
* From -01 to -02
  - Changes to message_3 formatting and processing
* From -00 to -01
  - Editorial changes and corrections

Acknowledgments

The authors want to thank Christian Amsss, Scott Fluhrer, Charlie Jacomme, and Marco Tiloca for reviewing and commenting on intermediate versions of the draft.

This work has been partly funded by PID2023-148104OB-C43 funded by MICIU/AEI/10.13039/501100011033 (ONOFRE4), FEDER/UE EU HE CASTOR under Grant Agreement No 101167904 and EU CERTIFY under Grant Agreement No 101069471.

This work was supported partially by Vinnova - the Swedish Agency for Innovation Systems - through the EUREKA CELTIC-NEXT project CYPRESS.

Authors' Addresses

Elsa Lopez-Perez
Inria
Email: elsa.lopez-perez@inria.fr

Gran Selander
Ericsson
Email: goran.selander@ericsson.com

John Preu Mattsson
Ericsson
Email: john.mattsson@ericsson.com

Rafael Marin-Lopez
University of Murcia
Email: rafa@um.es

Francisco Lopez-Gomez
University of Murcia
Email: francisco.lopezg@um.es