individual
Internet-Draft
Intended status: Standards Track
Expires: 8 January 2026

L. Pocero Fraile
C. Koulamas
A.P. Fournaris
E. Haleplidis
ISI, R.C. ATHENA

7 July 2025

KEM-based Authentication for EDHOC
draft-pocero-authkem-edhoc-00

Abstract

This document specifies extensions to the Ephemeral Diffie-Hellman over COSE (EDHOC) protocol to provide resistance against quantum computer adversaries by incorporating Post-Quantum Cryptography (PQC) mechanisms for both key exchange and authentication. It defines a Key Encapsulation Mechanism (KEM)-based authentication method to enable signature-free post-quantum authentication when PQC KEMs, such as NIST-standardized ML-KEM, are used.

Status of This Memo

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

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

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

This Internet-Draft will expire on 8 January 2026.

Copyright Notice

Table of Contents

1. Introduction

The purpose of this document is to address the quantum-resistant transition of the Ephemeral Diffie-Hellman over COSE (EDHOC) protocol by extending with a new Key Encapsulation Mechanism (KEM)-based authentication method and Post-Quantum Cryptography cipher suits.

The specific protocol is part of a more extensive analysis of the PQ transition for the EDHOC protocol, which is currently in the process of being published.

1.1. Motivation

The emerging Quantum Computing technologies bring new potential risks to the existing cryptographic infrastructures. Security mechanisms that rely on integer factorization or the discrete logarithm problem will be vulnerable to attacks by a Cryptographically Relevant Quantum Computer (CRQC). The European Commission recently issued a roadmap for the transition to Post-Quantum Cryptography (PQC), establishing a 2030 deadline for high-risk use cases and 2035 for medium-risk use cases, in alignment with the 2035 deadline set by the U.S. government for completing the transition to PQC in federal systems.

The U.S. National Institute of Standards and Technology (NIST) has concluded its post-quantum cryptography standardization process with the release of its first standardized post-quantum algorithms in three new Federal Information Processing Standards (FIPS): FIPS 203 (ML-KEM, based on CRYSTALS-Kyber), FIPS 204 (ML-DSA, based on CRYSTALS-Dilithium), and FIPS 205 (SLH-DSA, based on SPHINCS+). Additionally, FALCON has been selected for future standardization, and NIST has launched a new initiative to evaluate alternative post-quantum signature schemes with compact signatures and efficient verification speeds. Complementing these efforts, the Post-Quantum Use in Protocols (PQUIC) IETF Working Group (WG) is developing operational and design guidelines to support the transition. For example, [RFC9794] defines terminology for post-quantum/traditional Hybrid schemes, while ongoings draft such as

[I-D.ietf-pquip-pqc-engineers] analyze the impact of CRQCs on existing systems and the challenges involved in transitioning to post-quantum algorithms.

The growing urgency to transition to PQC highlights the need to adapt EDHOC, whose current security relies on traditional Elliptic-Curve Cryptography(ECC), based on the discrete logarithm problem that is known to be vulnerable to attacks by CRQCs. The integration of the PQC mechanism into EDHOC raises important considerations around performance, as the protocol is explicitly designed for constrained environments where the number of handshake message rounds, network overhead, processing time, and power consumption are critical factors.

PQC algorithms generally have higher computational and memory costs compared to the classical cryptography algorithms they aim to replace because they often involve complex calculations and require larger byte sizes. Notably, the PQC digital signature schemes standardized by NIST, such as ML-KEM and ML-DSA, use significantly large public keys and signatures, which can be difficult to transmit over constrained networks. It is important to note that while FALCON, also selected for standardization by NIST, provides much shorter signatures than the lattice-based schemes, its current implementations have been shown to be vulnerable to side-channel attacks. The new compact schemes under NIST evaluation should be more suitable for constrained environments. However, the current Cortex-M4 implementations of some of the most compact PQC signature schemes, like SNOVA and MAYO, still demand substantial memory resources, making them impractical for many constrained devices. Additionally, others, such as SQISign, have only recently been supported on such platforms, and performance benchmarks for their signature operations are still unavailable.

On the other hand, the standardized ML-KEM offers significantly higher computational efficiency compared to all other PQC KEMs (order of magnitude faster) and is at least three times more efficient than the fastest PQC signature schemes. Therefore, extending EDHOC with a new authentication method that enables a signature-free KEM-based EDHOC has the potential to reduce memory and processing requirements when ML-KEM is used. The approach can also result in lower network overhead compared to signature-based EDHOC implementations that rely on standardized PQC signature-based algorithms.

Some standardization efforts propose adopting the KEM-based authentication mechanism to mitigate the overhead introduced by PQC digital signatures. For example, [I-D.celi-wiggers-tls-authkem] specifies a KEM-based authentication scheme for TLS 1.3, while [I-D.uri-lake-pquake] aims to define a general Post-Quantum Authentication Key exchange protocoll, which based on the same approach.

This document describes a KEM-based authentication mechanism specifically for the EDHOC protocol, introducing a new authentication method intended to provide a PQC signature-free variant as the static DH authentication method intends. The static-DH authentication of EDHOC is based on the XX pattern of the Noise framework protocol [Noise], where channel security guarantees are increasingly established by encrypting transmitted messages with keys derived from chains of shared secrets, as soon as those secrets become available. To align with this model, the KEM-based authentication Method defined in this document follows the approach outlined in [PQNoise-CCS22], which provides a recipe for transforming classical Noise patterns into PQ variants. This specification defines the necessary modifications to the EDHOC protocol to support the PQ-Noise framework while preserving security properties comparable to those of the static-DH authentication method.

1.2. Terminology and Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [RFC2119].

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] and COSE Algorithms [RFC9053], When referring to CBOR, this specification always refers to Deterministically Encoded CBOR, as specified in Section 4.2.1 and 4.2.2 of [RFC8949]. The single output from authenticated encryption (including the authentication tag) is called "ciphertext", following [RFC5116].

1.2.1. Key Encapsulation Mechanisms (KEMs)

The Key Encapsulation Mechanism consists of 3 algorithms:

* *(pk, sk) <- KEM.KeyGen( )*: The probabilistic key generation algorithm generates a KEM key pair consisting of a public encapsulation key (pk) and secret decapsulation key (sk).

* *(ss , ct) <- KEM.Encapsulate(pk)*: The probabilistic encapsulation algorithm takes as input a public encapsulation key (pk) and produces a shared secret (ss) and ciphertext (ct).

* *(ss) <- KEM.Decapsulate(ct, sk)*: The decapsulation algorithm takes as input a secret encacpsulation key (sk) and produce a shared secret (ss).

2. Protocol Overview

This document defines a KEM-based authentication method for EDHOC in a general scenario where both parties may be mutually unknown. It aims to provide a free-signature authentication scheme as the static DH authentication EDHOC method 3 does, which relies on the XX pattern from the Noise framework [Noise], supporting mutual authentication and the transmission of encrypted public credentials. The proposed protocol adopts the approach provided by [PQNoise-CCS22] to transform the classical Noise XX pattern in EDHOC into a PQ Noise XX variant. This results in a quantum-resistant, KEM-only version of EDHOC when a PQC KEM is used.

The PQ translation of the Noise XX pattern requires introducing up to one additional round trip. With KEMs, the owner of the static key cannot combine their static private key with the ephemeral public key belonging to the other party to immediately prove their identity in the next message, as is possible with DH. Instead, the party must first receive a ciphertext that encapsulates its static public key, generated by the peer, before it can authenticate itself. This necessitates an additional key-confirmation message from the key owner, using the key derived from the encapsulated value.

The KEM-based EDHOC protocol consists of five mandatory messages (message_1, message_2, message_3, message_4, and message_5), and an error message, between an Initiator (I) and a Responder (R). Error handling and cipher suit negotiation mechanisms are the same as defined in Section 6 of [RFC9528]. All EDHOC messages are CBOR Sequences as specified in [RFC9528]. Figure 1 illustrates a KEM-based authentication EDHOC message flow as well as the content of each message. The protocol elements in Figure 1 are introduced in this Section and in Section 4. Message formatting and processing are specified in Section 4

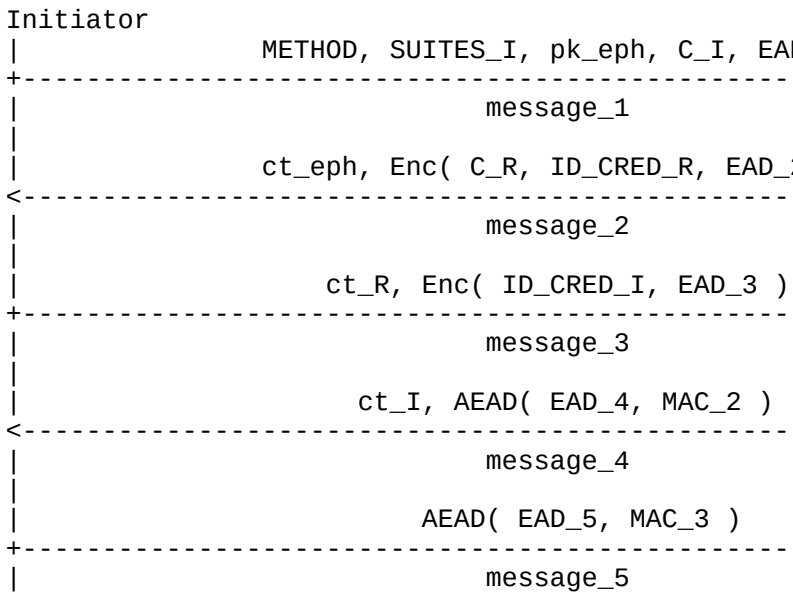


Figure 1: EDHOC Message Flow using the KEM-based Authentication Method

The parties exchange ephemeral and static KEM public keys, along with ciphertexts that encapsulate these keys, compute shared secrets and pseudorandom keys PRK, and derive symmetric session keys to encrypt message elements contained in intermediate handshake messages. All handshake messages include encrypted components protected with these derived session keys, offering varying levels of confidentiality and authenticity, except for the first message, which is sent in plaintext. The parties compute a shared secret session key, PRK_out, from which symmetric application keys are derived to protect application data. The Initiator derives these keys after receiving message_4, and the Responder after receiving message_5.

* pk_eph is the ephemeral KEM public key generated by the Initiator.

* ct_eph is the ephemeral ciphertext computed by the Responder with the KEM.encapsulation algorithm over the received ephemeral public key (pk_eph).

* ct_R is the responder ciphertext computed by the Initiator with the KEM.encapsulation algorithm over the static KEM public key of the Responder, retrieved from the received ID_CRED_R in message_2.

* ct_I is the Iniatiator ciphertext computed by the Responder with the KEM.encapsulation algorithm over the static KEM public key of the Initiator, retrieved from the received ID_CRED_I in message_1.

* "CRED_I and CRED_R are the authentication credentials containing the public authentication keys of I and R, respectively", as defined in Section 2 of [RFC9528] .

* "ID_CRED_I and ID_CRED_R are used to identify and optionally transport the credentials of I and R, respectively", as defined in Section 2 of [RFC9528].

* "Enc(), AEAD(), and MAC() denote encryption, Authenticated Encryption with Associated Data, and Message Authentication Code, crypto algorithms applied with keys derived from one or more shared secrets calculated during the protocol", as defined in Section 2 of [RFC9528]

* "SUITES_I contains cipher suites supported by the Initiator and formatted and processed as specified in Section 3.6 and 6.3.2 of [RFC9528]"".

* "METHOD is an integer specifying the authentication method",as defined in Section 3.2 of [RFC9528]. In this case method 4; see Section 2.1.2.1.

* C_I and C_R are Connection Identifiers chosen by the Initiator and Responder, respectively, as specified in Section 3.3 of [RFC9528]

* EAD_1, EAD_2, EAD_3, EAD_4, EAD_5 are External Authorization Data included in message_1, message_2, message_3, message_4 and message_5 respectively

This protocol is designed so that it follows the provisions of [RFC9528], that is, to encrypt and integrity protect as much information as possible and derive symmetric keys and random material using EDHOC_KDF with as much previous information as possible

2.1. Protocol Elements

This section describes the principal protocol elements that differ from the definitions of EDHOC and highlights the most important similarities. For the missing elements, the definitions in Section 3 of [RFC9528] SHOULD be consulted.

2.1.1. Ephemeral KEM

The ephemeral KEM is used to provide forward secrecy. The Initiator generates a new ephemeral KEM key pair in every new session to ensure that the compromise of long-term keys does not compromise past communications. The elements of the Ephemeral KEM are:

* The ephemeral KEM key pair (pk_eph, sk_eph) is generated by the Initiator using the following function:

  pk_eph, sk_eph <- KEM.KeyGen()

* The ephemeral shared secret (ss_eph) and the ephemeral ciphertext (ct_eph) are generated using the encapsulation and decapsulation functions: in the Responder

  ss_eph, ct_eph <- KEM.Encapsulate(pk_eph)

  in the Initiator

  ss_eph <- KEM.decapsulation(ct_eph, sk_eph)

2.1.2. Authentication Parameters

The protocol performs the same authentication-related operations as described in Section 3.5 of [RFC9528]

The protocol transports information about credentials ID_CRED_I and ID_CRED_R in message_2 and message_3, respectively. The authentication of these credentials is verified through MAC_2 and MAC_3, sent by the Responder and the Initiator in message_4 and message_5, respectively.

2.1.2.1. Method

The protocol extends EDHOC with a new KEM-based authentication method, where both parties use static KEM key pairs. The authentication is provided by a Message Authentication Code (MAC) included in message_4 and message_5 to authenticate the Responder and Initiator, respectively. The Initiator and Responder need to have agreed on a method 4.

Method Type Value	Initiator Authentication Key	Responder Authentication Key
4	Static KEM Key	Static KEM Key

Table 1: Authentication Keys for Method Types

2.1.2.2. Authentication Keys

The authentication key MUST be a static KEM authentication key pair.

* The Initiator static KEM authentication key pair: (pk_I, sk_I)

* The Responder static KEM authentication key pair: (pk_R, sk_R)

2.1.2.3. Authentication Credentials

The authentication credentials, CRED_I and CRED_R, contain the static KEM authentication public key of the Initiator and Responder, respectively, as described in Section 3.5.2 of [RFC9528].

* The authentication credentials can be X.509 certificates seconded as bstr, as defined in Section 3.5.2 of [RFC9528], using [RFC9360]. [I-D.ietf-lamps-kyber-certificates] describes the conventions for using the ML-KEM in X.509 Public Key Infrastructure.

* Additionally, the authentication credential may include a COSE_key, formatted as specified in [RFC8392], to reduce the credential size and avoid the PQC signature verification needed when X.509 certificates are used. New IANA value registries should be defined to extend COSE Algorithms with the corresponding KEMs algorithm values.

2.1.2.4. Identification of Credentials

The ID_CRED fields are used to identify and optionally transport credentials as defined in Section 3.5.3 of [RFC9528]. The authentication method defined in this document operates within the general EDHOC framework described in Section 3.5.3 of [RFC9528], where ID_CRED_X can either contain the full CRED_X credentials or an identifier of those credentials if they have already been provided out-of-band.

* "ID_CRED_R is intended to facilitate for the Initiator retrieving the authentication credential CRED_R and the authentication key of R", as defined in Section 3.5.3 of [RFC9528]. For the authentication method defined in this document, the authentication key is the static KEM public key.

* "ID_CRED_I is intended to facilitate for the Responder retrieving the authentication credential CRED_I and the authentication key of I", as defined in Section 3.5.3 of [RFC9528]. For the authentication method defined in this document, the authentication key is the static KEM public key.

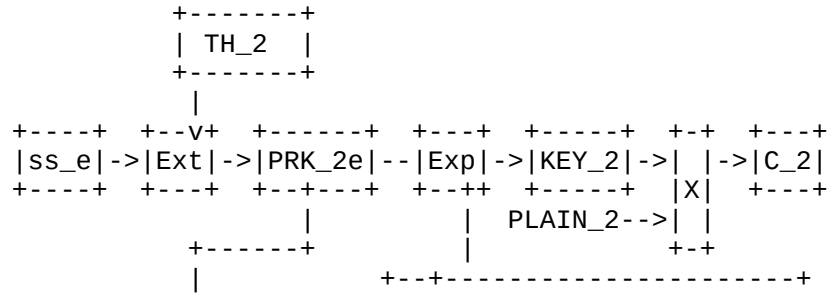
2.1.3. Cipher Suites

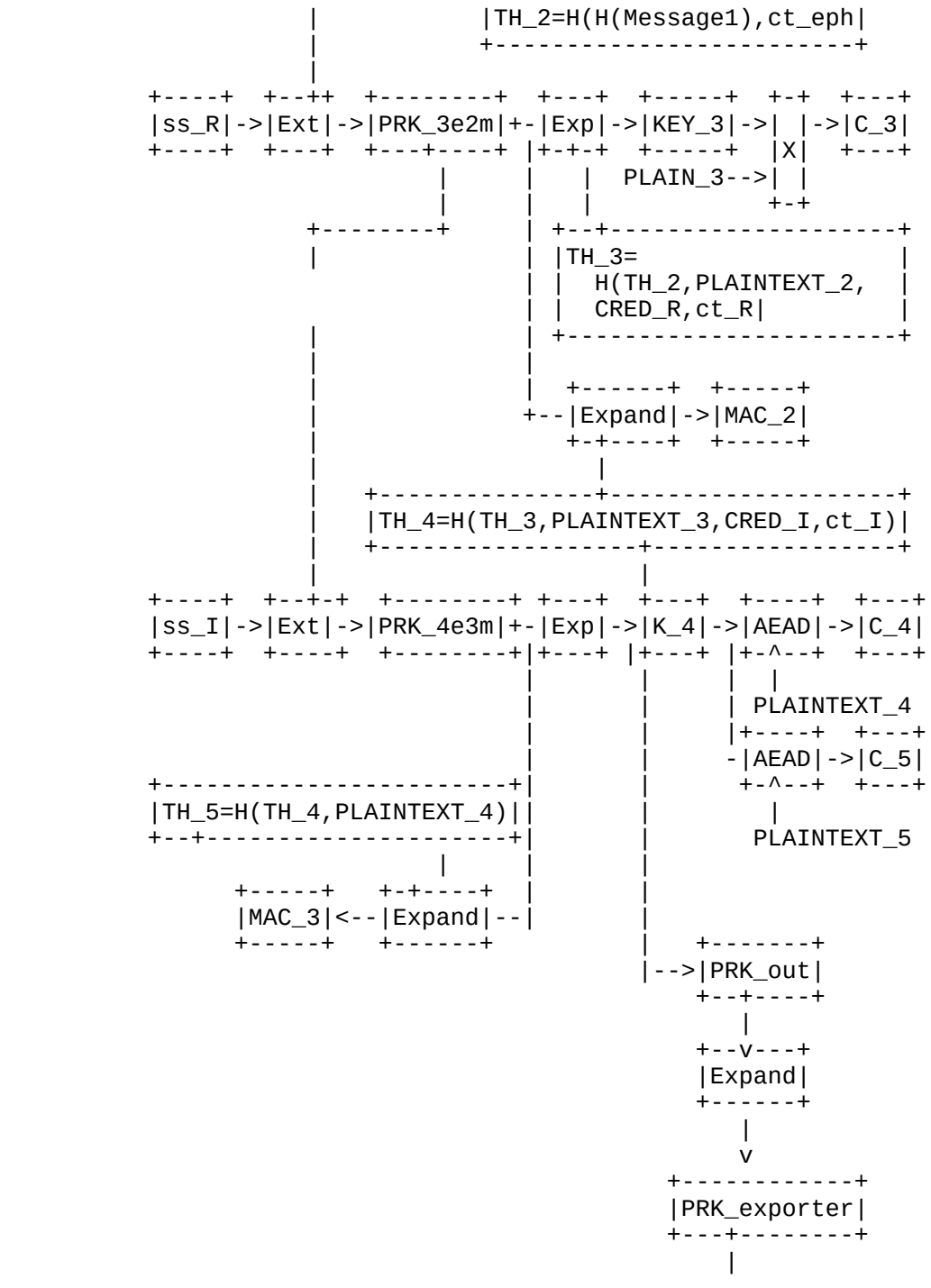
The authentication method specified in this document uses the EDHOC cipher suites element, as defined in Section 3.6 of [RFC9528]. An EDHOC cipher suit consists of an ordered set of algorithms from the "COSE Algorithms" IANA registry [RFC9053]. The predefined EDHOC cipher suites are also listed in the IANA registry, as specified in Section 10.2 of [RFC9528].

A new predefined cipher suite SHOULD be added to the IANA registry, specifying each supported KEM in the EDHOC Key Exchange Algorithm parameter. An example of this, when ML-KEM is used, is shown in Section 5. The same KEM algorithm selected for key exchange SHOULD also be used for KEM-based authentication when method 4 is selected. Furthermore, the KEM algorithms used SHOULD also be added to the COSE Algorithms IANA registry to identify them, as is shown in Section 5.

3. Key Derivation

This section highlights the differences and similarities in the key derivation process of the KEM-based authentication method compared to [RFC9528]. An overview of the EDHOC key schedule when using the KEM-based authentication method is shown in Figure 2, and each key derivation step is explained in the following subsections.




|TH_2=H(H(Message1),ct_eph|
ss_R
Ext
PRK_3e2m
Exp
KEY_3
C_3
X
PLAIN_3
TH_3=
H(TH_2,PLAINTEXT_2,
CRED_R,ct_R|
Expand
MAC_2
TH_4=H(TH_3,PLAINTEXT_3,CRED_I,ct_I)|
ss_I
Ext
PRK_4e3m
Exp
K_4
AEAD
C_4
PLAINTEXT_4
AEAD
C_5
TH_5=H(TH_4,PLAINTEXT_4)|
PLAINTEXT_5
MAC_3
Expand
PRK_out
Expand
PRK_exporter

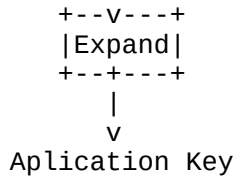


Figure 2: EDHOC Message Key Derivation using the KEM-based Authentication Method

3.1. Keys for EDHOC Message Processing

3.1.1. EDHOC_Extract

The pseudorandom keys (PRKs) used for KEM-based authentication are derived using the same EDHOC_Extract function defined in [RFC9528], where the input keying material (IKM) and Salt are specified for each PRK below.

3.1.1.1. PRK_2e

The pseudorandom key PRK_2e is derived with the following input:

* The salt SHALL be TH_2.

* The IKM SHALL be the ephemeral KEM shared secret (ss_eph)

When SHA-256 is used PRK_2e is produced as follows:

PRK_2e = HMAC-SHA-256(TH_2, ss_eph)

Where the ephemeral shared secret ss_eph is the output of the following functions in the Initiator and Responder respectively

Initiator:

ss_eph <- KEM.Decapsulate(ct_eph, sk_eph)

Responder:

ss_eph, ct_eph <- KEM.Encapsulate(pk_eph)

3.1.1.2. PRK_3e2m

The pseudorandom key PRK_3e2m is derived with the following input:

* The salt SHALL be the SALT_3e2m derived from PRK_2e

* The IKM SHALL be the KEM shared secret ss_R, used to authenticate the Responder

PRk_3e2m is derived as follows:

PRK_3e2m = EDHOC_Extract(SALT_3e2m, ss_R)

Where the KEM shared secret ss_R used to authenticate the Responder is the output of the following functions in the Initiator and Responder, respectively

Initiator:

ss_R, ct_R <- KEM.Encapsulate(pk_R)

Responder:

ss_R <- KEM.Decapsulate(ct_R, sk_R)

3.1.1.3. PRK_4e3m

The pseudorandom key PRK_4e3m is derived with the following input:

* The salt SHALL be the SALT_4e3m, derived from PRK_3e2m

* The IKM SHALL be the KEM shared secret ss_I, used to authenticate the Initiator

PRk_4e3m is derived as follows:

PRK_4e3m = EDHOC_Extract(SALT_4e3m, ss_I)

Where the KEM shared secret ss_I used to authenticate the Initiator is the output of the following functions in the Initiator and Responder, respectively

Initiator:

ss_I <- KEM.Decapsulate(ct_I, sk_I)

Responder:

ss_I, ct_I <- KEM.Encapsulate(pk_I)

3.1.2. EDHOC_Expand and EDHOC_KDF

The output key materials (OKMs) are derived from the PRKs in the same way as described in Section 4.1.2 of [RFC9528], with modifications in the transcript hashes THs input contraction as specified in Section 4.

The same OKMs, including keys, initialization vectors (IV), and salts as those shows in Section 4.1.2 of [RFC9528] Figure 6 are derived, with one exception:

* KEYSTREAM_3 is computed instead K3/IV3 because the derived key at this stage can not yet be used to authenticate the Initiator in message_2.

The final key derivations using EDHOC_KDF is shwon in Figure 3 . Further details of the key derivation and how the output keying material is used are specified in Section 4

```
KEYSTREAM_2   = EDHOC_KDF( PRK_2e,   0, TH_2,      plaintext_length )
SALT_3e2m     = EDHOC_KDF( PRK_2e,   1, TH_2,      hash_length )
MAC_2         = EDHOC_KDF( PRK_3e2m, 2, context_2, mac_length_2 )
KEYSTREAM_3   = EDHOC_KDF( PRK_3e2m, 3, TH_3,      key_length )
SALT_4e3m     = EDHOC_KDF( PRK_3e2m, 5, TH_4,      hash_length )
MAC_3         = EDHOC_KDF( PRK_4e3m, 6, context_3, mac_length_3 )
PRK_out       = EDHOC_KDF( PRK_4e3m, 7, TH_4,      hash_length )
K_4           = EDHOC_KDF( PRK_4e3m, 8, TH_4,      key_length )
IV_4          = EDHOC_KDF( PRK_4e3m, 9, TH_4,      iv_length )
PRK_exporter  = EDHOC_KDF( PRK_out, 10, h'',       hash_length )
```

Figure 3: Key Derivations Using EDHOC_KDF for the KEM-based Authentication Method

Notice that a new key session (K_5/IV_5) can be derived from the same PRK_4e3m, using Th_5 as the info parameter, to encrypt message_5, if separate keys are shown to enhance security in any way. The initial version of this protocol adopts a simpler approach by deriving a single session key to protect both messages, which are different from the MAC keys.

3.1.3. PRK_out

The pseudorandom key PRK_out is the output session key of a completed EDHOC session and is derived as follows:

```
PRK_out = EDHOC_KDF( PRK_4e3m, TH_4, hash_length )
```

3.2. Keys for EDHOC Applications

Keying material for the application can be derived using the same EDHOC_Exporter interface defined in Section 4.2.1 of [RFC9528]

4. Message Formatting and Processing

This section outlines the message format and the procedures for composing and processing each message.

4.1. KEM-based Authentication EDHOC Message 1

4.1.1. Formating of Message 1

message_1 retains the same format as defined in Section 5.2.1 of [RFC9528]. The same fields are used, except that GX is replaced by the KEM ephemeral public key (pk_eph) computed by the Initiator.

```
message_1 = (
  METHOD : int,
  SUITES_I : suites,
  pk_eph : bstr,
  C_I : bstr / -24..23,
  ? EAD_1,
)

suites = [ 2* int ] / int
EAD_1 = 1* ead
```

The new KEM-based authentication method (method 4) shoould be seletect in the METHOD field.

4.1.2. Initiator Composition of Message 1

The Initiator SHALL compose message_1 as follows:

* Construct SUITES_I following the Section 5.2.2 of [RFC9528] specifications

* Generate an ephemeral KEM Key pair (pk_eph) using the KEM algorithm from the selected cipher suit. The ephemeral key pair is computed by the Initiator using the following function:

  pk_eph, sk_eph <- KEM.KeyGen()

* Choose a conection identifier as in Section 5.2.2 of [RFC9528].

* Encode message_1 as sequence of CBOR-encoded elements, as specified in Section 4.1.1

4.1.3. Responder Processing of Message 1

The Responder SHALL process message_1 in the following order:

1. "Decode message_1", as specified in Section 5.2.3 of [RFC9528]

2. "Process message_1", as specify in Section 5.2.3 of [RFC9528]

3. "If all processing is completed successfully, and if EAD_1 is present, then make it available to the application", as specified in Section 5.2.3 of [RFC9528]

4.2. KEM-based authentication EDHOC Message 2

4.2.1. Formating of Message 2

message_2 keeps the same formatting as Section 5.3.1 of [RFC9528]. The same fields are used instead GY is replaced with the ephemeral KEM ciphertext (ct_eph) computed on the Responder..

```
message_2 = (
  ct_eph_CIPHERTEXT_2 : bstr,
)
```

where cc_eph_CIPHERTEXT_2 is the concatenation of ct_eph and CIPHERTEXT_2.

4.2.2. Responder Composition of Message 2

The Responder SHALL compose message_2 as follows:

* Encapsulate the ephemeral KEM key received within message_1 using the KEM algorithm in the selected cipher suit. The ephemeral KEM ciphertext and the KEM ephemeral shared secret are computed by the Responder using the following function:

```
ss_eph, ct_eph <-  KEM.Encapsulate(pk_eph)
```

* Compute the PRK_2e pseudorandom key from the ephemeral KEM shared secret (ss_eph)

* "Choose a connection identifier C_R", as specified in Section 5.3.2 of [RFC9528]

* Compute the transcript hash TH_2 = H(pk_eph,H(message_1)) as specified in Section 5.3.2 of [RFC9528]

* At this point, the Responder is not jet able to authenticate itself, so MAC_2 is not computed

* CIPHERTEXT_2 is calculated, with a binary additive stream cipher as in Section 5.3.2 of [RFC9528], using a keystream (KEYSTREAM_2) generated with EDHOC_Expand and the following plaintext:

  - Compute PLAINTEXT_2 as:

    PLAINTEXT_2 = (C_R,ID_CRED_R,?EAD_2)

    where C_R, ID_CRED_R and EAD_2 elements corresponds with the ones in Section 5.3.2 of [RFC9528].

  - Compute KEYSTREAM_2 as in Section 3.1.2

  - Compute CIPHERTEXT_2 as in Section 5.3.2 of [RFC9528]

    CIPHERTEXT_2 = PLAINTEXT_2 XOR KEYSTREAM_2

* Encode message_2 as a sequence of CBOR-encoded data items as specified in Section 4.2.1

4.2.3. Initiator Processing of Message 2

The Initiator SHALL process message_2 in the following order:

1. Decode message_2

2. "Retrieve the protocol state" as proposed in Section 5.3.3 of [RFC9528]

3. Compute the ephemeral KEM shared_secret (ss_eph) by decapsulating the KEM ciphertext (ct_eph) received in message_2 using the ephemeral secret key (sk_eph). The ephemeral KEM shared secret is computed by the Initiator using the following function:

   ss_eph <- KEM.Decapsulate(ct_eph, sk_eph)

4. Compute the PRK_2e pseudorandom key from the ephemeral KEM shared secret (ss_eph)

5. Compute the transcript hash TH_2 = H(pk_eph,H(message_1))

6. Derive KEYSTREAM_2 as in Section 3.1.2

7. Decrypt CIPHERTEXT_2; see Section 4.2.2

8. If all processing is completed successfully, then make ID_CRED_R and (if present) EAD_2 available to the application as in Section 5.3.3 of [RFC9528]

9. Obtain the authentication credential (CRED_R) from the (ID_CRED_R) as in Section 5.3.3 of [RFC9528], and the static KEM authentication key (pk_R) of the Responder

10. Encapsulate the retrieved static KEM authentication key of the Responder (pk_R) calculating the corresponding ciphertext (ct_R) and shared secret (ss_R) with the following function:

    ss_R, ct_R <- KEM.Encapsulate(pk_R)

11. Compute the new PRK_3e2m from a chain that includes both the ephemeral KEM shared secret (ss_eph) and the latest KEM shared secret for the Authentication of the Responder (ss_R), as defined in Section 3.1.1.2

4.3. KEM-based authentication EDHOC Message 3

4.3.1. Formating of Message 3

message_3 SHALL be a CBOR Sequence as defined below:

```
message_3 = (
  ct_R : bstr,
  CIPHERTEXT_3 : bstr,
)
```

4.3.2. Initiator Composition of Message 3

The Initiator SHALL process the composition of message_3 as follows:

* Compute the transcript hash TH_3=H(ct_R,TH_2,PLAINTEXT_2,CRED_R) as specified in Section 5.4.2 of [RFC9528].

* Derive the new session key KEYSTREAM_3 as defined in Section 3.1.2. The Initiator can use this key to compute CIPHERTEXT_3, but it cannot be used to authenticate itself.

* At this point, the Responder is not jet able to authenticate itself, so MAC_3 is not computed.

* CIPHERTEXT_3 is calculated with a binary additive stream cipher, using a keystream (KEYSTREAM_3) generated with EDHOC_Expand and the following plaintext:

  - Compute PLAINTEXT_3 as:

    PLAINTEXT_3 = (C_I,ID_CRED_I,?EAD_3)

    where C_I, ID_CRED_I and EAD_3 elements corresponds with the ones in Section 5.3.3 of [RFC9528].

  - Compute KEYSTREAM_3 as in Section 3.1.2, where plaintext_length is the length of PLAINTEXT_3

  - Compute CIPHERTEXT_3 as follows:

    CIPHERTEXT_3 = PLAINTEXT_3 XOR KEYSTREAM_3

* Encode message_3 as a CBOR data item as specified in Section 4.3.1

4.3.3. Responder Processing of Message 3

The Responder SHALL process message_3 in the following order:

1. Decode message_3

2. "Retrieve the protocol state", as defined in Section 5.4.3 of [RFC9528]

3. Compute the KEM shared_secret (ss_R) for the authentication of the Responder by decapsulating the KEM ciphertext (ct_R) received in message_3 using the Responder static KEM secret key (sk_R). The KEM shared secret is computed by the Responder using the following function:

   ss_R <- KEM.Decapsulate(ct_R, sk_R)

4. Compute the new PRK_3e2m from a chain that includes both the ephemeral KEM shared secret (ss_eph) and the latest KEM shared secret for the Authentication of the Responder (ss_R), as defined in Section 3.1.1.2

5. Compute the transcript hash TH_3=H(ct_R,TH_2,PLAINTEXT_2,CRED_R)

6. Compute KEYSTREAM_3 as in Section 3.1.2, where plaintext_length is the length of PLAINTEXT_3

7. Decrypt CIPHERTEXT_3; see Section 4.3.2

8. "If all processing is completed successfully, then make ID_CRED_I and (if present) EAD_2 available to the application", as in Section 5.3.4 of [RFC9528]

9. "Obtain the authentication credential (CRED_I) from the (ID_CRED_I)" as in Section 5.3.4 of [RFC9528] and the static KEM authentication key (pk_I) of the Initiator.

4.4. KEM-based authentication EDHOC Message 4

4.4.1. Formating of Message 4

message_4 SHALL be a CBOR Sequence as defined below:

```
message_3 = (
  ct_I : bstr,
  CIPHERTEXT_4 : bstr,
)
```

4.4.2. Responder Composition of Message 4

The Responder SHALL process the composition of message_4 as follows:

* Compute the transcript hash TH_4 = H(ct_I,TH_3, PLAINTEXT_3, CRED_I)

* Compute MAC_2 as defined in Section 3.1.2, with context_2 =<< C_R, ID_CRED_R, TH_4, CRED_R, ? EAD_4 >>

  - The Responder authenticates with a PRK_3e2m derived from the KEM ephemeral shared secret and with the shared secret computed over its static KEM key.

  - The mac_lenght_2 is equal to the EDHOC MAC length of the selected cipher suit.

  - The C_R, ID_CRED_R and CRED_R elements corresponds with the ones in Section 5.3.2 of [RFC9528]

  - The latest transcript hash TH_4 and the External Application Data included in Message 4 (EAD_4) are used.

* Encapsulate the retrieved static KEM authentication key of the Initiator (pk_I) calculating the corresponding ciphertext (ct_I) and shared secret (ss_I) with the following function:

  ss_I, ct_I <- KEM.Encapsulate(pk_I)

* Compute the new PRK_4e3m from a chain that includes the ephemeral KEM shared secret (ss_eph), the KEM shared secret for the Authentication of the Responder (ss_R) , and the latest KEM shared secret for the Authentication of the Initiator (ss_I) as defined in Section 3.1.1.3

* Derive the session key K_4/IV4 as in Section 3.1.2.

* Compute a COSE_Encrypt0 object as defined in Section 5.2 and 5.3 of [RFC9052], with the EDHOC AEAD algorithm of the selected cipher suite, using the encryption key K_4, the initialization vector IV_4 (if used by the AEAD algorithm), the plaintext PLAINTEXT_4, and the following parameters as input:

  - protected = h''

  - external_aad = TH_4

  - K_4 and IV_4 are defined in Section 3.1.2

  - PLAINTEXT_4 = (MAC_2, ?EAD_4)

  CIPHERTEXT_4 is the 'ciphertext' of COSE_Encrypt0.

* Compute the transcript hash TH_5 = H(TH_4, PLAINTEXT_4)

* Encode message_4 as a CBOR data item as specified in Section 4.4.1

4.4.3. Initaitor Processing of Message 4

The Initiator SHALL process message_4 in the following order:

1. Decode message_4

2. "Retrieve the protocol state using available message correlation", as in Section 3.4.2 of [RFC9528].

3. Compute the KEM shared secret (ss_I) for the authentication of the Initiator by decapsulating the KEM ciphertext (ct_I) received in message_4 using the Responder static KEM secret key (sk_I). The KEM shared secret is computed by the Initiator using the following function:

   ss_I <- KEM.Decapsulate(ct_I, sk_I)

4. Compute the transcript hash TH_4 = H(ct_I,TH_3, PLAINTEXT_3, CRED_I)

5. Compute the new PRK_4e3m from a chain that includes the ephemeral KEM shared secret (ss_eph), the KEM shared secret for the Authentication of the Responder (ss_R), and the latest KEM shared secret for the Authentication of the Initiator (ss_I) as defined in Section 3.1.1.3

6. Derive the session key K_4/IV4 as in Section 3.1.2.

7. Decrypt and verify the COSE_Encrypt0 (CIPHERTEXT_4) as defined Section 5.2 and 5.3 of [RFC9052]], with the EDHOC AEAD algorithm in the selected cipher suite and the parameters defined in Section 4.4.2.

8. Verify MAC_2 as defined in Section 4.4.2, and make the result of the verification available to the application.

4.5. KEM-based authentication EDHOC Message 5

4.5.1. Formating of Message 5

message_5 SHALL be a CBOR Sequence as defined below:

```
message_3 = (
  CIPHERTEXT_5 : bstr,
)
```

4.5.2. Initiator Composition of Message 5

The Responder SHALL process the composition of message_5 as follows:

* Compute the transcript hash TH_5 = H(TH_4, PLAINTEXT_4)

* Compute MAC_3 as defined in Section 3.1.2, with context_3 =<< C_I, ID_CRED_I, TH_5, CRED_I, ? EAD_5 >>

  - The Initiator authenticates with a PRK_4e3m derived from the three shared secrets, including the shared secret computed over its static KEM key (ss_I).

  - The mac_lenght_3 is equal to the EDHOC MAC length of the selected cipher suit.

  - The C_I, ID_CRED_I and CRED_I elements corresponds with the ones in Section 5.4.2 of [RFC9528]

  - The latest transcript hash TH_5 and the External Application Data included on Message 5 (EAD_5) are used.

* Compute a COSE_Encrypt0 object as defined inSection 5.2 and 5.3 of [RFC9052], with the EDHOC AEAD algorithm of the selected cipher suite, using the encryption key K_4, the initialization vector IV_4 (if used by the AEAD algorithm), the plaintext PLAINTEXT_5, and the following parameters as input:

  - protected = h''

  - external_aad = TH_5

  - K_5 and IV_5 are defined in Section 3.1.2

  - PLAINTEXT_5 = (MAC_3, ?EAD_5)

  CIPHERTEXT_5 is the 'ciphertext' of COSE_Encrypt0.

* Calculate PRK_out as defined in Section 3.1.3. The Initiator can now derive application keys using the EDHOC_Exporter interface; see Section 3.2

* Encode message_5 as a CBOR data item as specified in Section 4.5.1

* "Make the connection identifiers (C_I and C_R) and the application algorithms in the selected cipher suite available to the application" as in Section 5.4.2 of [RFC9528]

After creating message_5, the Initiator can compute PRK_out and derive application keys using the EDHOC_Exporter interface. The Initiator SHOULD now persistently store PRK_out or application keys and send protected application data, since it has already verified message_4, which is protected with a derived application key by the Responder, and the application has authenticated the Responder.

4.5.3. Responder Processing of Message 5

The Initiator SHALL process message_5 in the following order:

1. Decode message_5

2. "Retrieve the protocol state using available message correlation" as in Section 3.4.2 of [RFC9528].

3. Decrypt and verify the COSE_Encrypt0 (CIPHERTEXT_5) as defined in Section 5.2 and 5.3 of [RFC9052], with the EDHOC AEAD algorithm in the selected cipher suite and the parameters defined in Section 4.5.2.

4. Verify MAC_3 as defined in Section 4.5.2, and make the result of the verification available to the application.

5. Calculate PRK_out as defined in Section 3.1.3. The Initiator can now derive application keys using the EDHOC_Exporter interface; see Section 3.2

After verifying message_5, the Responder can compute PRK_out and derive application keys using the EDHOC_Exporter interface. The Responder SHOULD now persistently store PRK_out or application keys and send protected application data, since it has already verified message_5, which is protected with a derived application key by the Initiator, and the application has authenticated the Initiator.

5. IANA Considerations

5.1. COSE Algorithms Registry

The "COSE Algorithms" Registry from "CBOR Object Signing and Encryption (COSE)" group SHOULD be extended with new values to include PQC KEM algorithms. The extension of values from the "Standards Action with Expert Review" range for ML-KEM algorithms at NIST security levels 1 and 3, is proposed in Table 2

Registry Name: COSE Algorithms

Reference: draft-pocero-authkem-edhoc-00

The columns of the registry are Name, Value, Description, Capabilities, Change Controller, Reference and Recommended, where Value is an integer and the other columns are text strings. The new values proposed are:

Name	Value	Description	Change Controller	Reference
Unassigned	-256 to -56			
ML-KEM-512	-54	CBOR object KEM Algorithm for ML-KEM-512	IETF	[draft-pocero-authkem-edhoc-00]
ML-KEM-1024	-55	CBOR object KEM Algorithm for ML-KEM-1024	IETF	[draft-pocero-authkem-edhoc-00]

Table 2: COSE Algorithms

5.2. EDHOC Cipher Suites Registry

The "EDHOC Cipher Suites" Registry from group "Ephemeral Diffie-Hellman Over COSE (EDHOC)" SHOULD be extended with new values to include the cipher suits with the KEM algorithm used. While the KEM-based authentication protocol specified in this document can support different KEM algorithms, the NIST-standardized ML-KEM is RECOMMENDED. The extension of values from the "Standards Action with Expert Review" range, when the ML-KEM algorithm is used at NIST security levels 1 and 3, is proposed in Table 3

Registry Name: EDHOC Cipher Suites

Reference: draft-pocero-authkem-edhoc-00

The columns of the registry are Value, Array, Description, and Reference, where Value is an integer and the other columns are text strings. The new values proposed are:

Value	Array	Description	Reference
7	30, -16, 16, -48 , 10, -16	AES-CCM-16-128-128, SHA-256, 16, ML-KEM-512, ML-DSA-44, AES-CCM-16-64-128, SHA-256	draft-pocero-authkem-edhoc-00
8	10, -16, 8, 1, , -49, -16	A256GCM, SHA-384, 16, ML-KEM-1024, ML-DSA-65, A256GCM, SHA-384	draft-pocero-authkem-edhoc-00
Unassigned	9 to 22		

Table 3: EDHOC Cipher Suites

The PQC ML-DSA signature algorithms are selected as the EDHOC signature algorithm parameter to verify X.509 certificate signatures when the X.509 credential type is used.

5.3. EDHOC Method Types Registry

The "EDHOC Method Types" Registry from group "Ephemeral Diffie-Hellman Over COSE (EDHOC)" SHOULD be extended with a new value that identifies the KEM-based authentication method. The extension value from the "Standards Action with Expert Review" range, is proposed in Table 3

Registry Name: EDHOC Method Types

Reference: draft-pocero-authkem-edhoc-00

The columns of the registry are Value, Initiator Authentication Key, Responder Authentication Key and Reference, where Value is an integer and the other columns are text strings. The new value proposed is:

Value	Initiator Authentication Key	Responder Authentication Key	Reference
4	Static KEM Key	Static KEM Key	[draft-pocero-authkem-edhoc-00]

Table 4: EDHOC Method Types

6. Security Considerations

EDHOC protocol with static DH keys enables the Initiator and Responder to generate an ephemeral-static shared secret using the other party's ephemeral public keys and their own credentials. This shared secret is then used to derive a session key for authentication. Messages 2 and 3 provide explicit authentication through MACs, which also bind the exchanged credentials to prevent misbinding attacks, as is described in Section 9.1 of [RFC9528]

In contrast, the KEM-based authentication mechanism requires an initial action from the other party. The Responder must first receive the encapsulation of its static public key generated by the Initiator to authenticate itself. To perform this encapsulation, the Initiator must retrieve the static KEM public key of the Responder from the ID_CRED_R sent in Message 2. As a result, the Responder cannot authenticate itself until Message 3 is processed, which contains the ct_R ciphertext necessary to derive the ss_R shared secret. Then it cannot generate MAC_2 or authenticate itself until then. Similarly, the Initiator cannot generate MAC_3 or authenticate itself before sending Message 3. This highlights the main challenge in integrating KEM-based authentication method within the EDHOC handshake.

To address this issue and maintain a level of identity protection, against active attacks on the Initiator and passive attacks on the Responder, the credentials continue to be sent encrypted in Messages 2 and 3. The credentials of the Initiator (ID_CRED_I) are encrypted using a key derived from both ephemeral KEM shared secret (ss_eph) and the Responder static KEM shared secret (ss_R), used to authenticate the Responder. At this stage, the specific encryption provided a form of weak forward secrecy, as the static KEM public key of the Responder has not yet been verified by the Initiator. However, the credentials are still protected against active attacks, as only the legitimate Responder, who possesses the corresponding private key (sk_R) is capable of deriving the session key and decrypting message_3.

Additionally the protocol is extended with two additional messages (Messages 4 and 5) to enable both parties to:

* Prove possession of the final session key, ensuring key confirmation to the other party

* Ensure mutual authentication by explicitly authenticating themselves using the final session key, which incorporates all three shared secrets: the ephemeral KEM shared secret (ss_eph) and the KEM shared secrets ss_I and ss_R used to authenticate the Initiator and Responder, respectively.

* Provide credential binding by including MAC_2 and MAC_3, ensuring the integrity and authenticity of the credentials exchanged in messages 2 and 3.

In [RFC9528], the transcript hashes (THs) are constructed as an accumulative hash, combining previous TH values with the current plain-text message. Each new plain-text message in the handshake is concatenated with the previous TH value, and the resulting hash forms the new TH. This process links each message in the sequence to all prior messages, creating a verifiable and continuous chain. As a result, any changes to the message content are detected during subsequent integrity verification using the transcript hashes. The KEM-based authentication method described in this document extends this approach. Both parties only need to verify the integrity and authenticity of the latest TH_4 and TH_5, which encompass all previous messages in the handshake. To facilitate this, the MAC-protected data in Messages 4 and 5 is modified to include TH_4 and TH_5 respectively. At the end of the handshake, both the Initiator and Responder can verify the integrity and authenticity of the entire handshake by checking the received MACs.

The payload security properties for the static DH authentication method and the KEM-based authentication method differ during the handshake. Unlike the static DH authentication method, the KEM-based method exhibits no authentication until the final two messages. It provides the same level of destination confidentiality for the first two and the last two messages, while message_3 offers weaker forward secrecy. The Initiator’ s credentials are encrypted within message_3 using a key derived from the Responder’ s static public key and the ephemeral key, ensuring that only the intended Responder can decrypt the credential, and protect them against active attacks.

Full forward secrecy and explicit mutual authentication are achieved once the KEM-based method handshake is completed, similar to the static-DH method handshake (described in Section 9.1 of [RFC9528]). Additionally, a potential misbinding attack will not be detected

until the handshake concludes, specifically when the Initiator verifies Message 4 and the Responder verifies Message 5. Therefore, EAD data should be treated as unprotected, and keying materials should not be persistently stored until the protocol is complete, as with the static-DH method (described in Section 9.1 of [RFC9528]). The final Application Session Key should only be derived at the end of the handshake, after ensuring mutual authentication, message handshake integrity, credentials authenticity, and proof of key possession.

Regarding non-repudiation, the KEM-based authnetication method provides the same implicit proof as EDHOC with static DH keys. It also maintains an equivalent level of downgrade protection, as the negotiation base of the protocol is unchanged.

The proposed KEM-based authentication method with a 5-message handshake is designed to meet the same security requirements as static-DH method. However, it can be adapted to reduce the number of round trips while remaining suitable for scenarios where neither party knows the other beforehand. This is achieved by transmitting the ID_CRED_I credentials of the Initiator in plain-text within the message_1, similar to the Noise IX pattern, where the Initiator’s static key is immediately transmitted to the Responder, despite or absent identity protection. This modification allows the Initiator to authenticate itself in Message 2, eliminating the need for Message 5. Since the Responder includes its credentials in the first message, Message 4 remains necessary to ensure explicit authentication of the Responder. This adaptation reduces the message exchange to four but sacrifices identity protection for the Initiator’s credentials.

7. References

7.1. Normative References

[I-D.ietf-lamps-kyber-certificates]
Turner, S., Kampanakis, P., Massimo, J., and B. Westerbaan, "Internet X.509 Public Key Infrastructure - Algorithm Identifiers for the Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM)", Work in Progress, Internet-Draft, draft-ietf-lamps-kyber-certificates-10, 16 April 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-kyber-certificates-10>.

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

[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/info/rfc8392>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/info/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/info/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/info/rfc9052>.

[RFC9360] Schaad, J., "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates", RFC 9360, DOI 10.17487/RFC9360, February 2023, <https://www.rfc-editor.org/info/rfc9360>.

[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/info/rfc9528>.

7.2. Informative References

[I-D.celi-wiggers-tls-authkem] Wiggers, T., Celi, S., Schwabe, P., Stebila, D., and N. Sullivan, "KEM-based Authentication for TLS 1.3", Work in Progress, Internet-Draft, draft-celi-wiggers-tls-authkem-05, 22 April 2025, <https://datatracker.ietf.org/doc/html/draft-celi-wiggers-tls-authkem-05>.

[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-13, 1 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqc-engineers-13>.

[I-D.uri-lake-pquake] Blumenthal, U., Luo, B., O'Melia, S., Torres, G., and D. A. Wilson, "PQuAKE - Post-Quantum Authenticated Key

Exchange", Work in Progress, Internet-Draft, draft-uri-lake-pquake-00, 22 April 2025, <https://datatracker.ietf.org/doc/html/draft-uri-lake-pquake-00>.

[Noise] Perrin, T., "The Noise Protocol Framework", Revision 34, July 2018, <https://noiseprotocol.org/noise.html>.

[PQNoise-CCS22] Angel, Y., Dowling, B., Hulsing, A., Schwabe, P., and F. Weber, "Post Quantum Noise", Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), pages 97109. Association for Computing Machinery, New York, NY, USA. DOI: https://doi.org/10.1145/3548606.3560577, 2022, <https://doi.org/10.1145/3548606.3560577>.

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

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

[RFC9794] Driscoll, F., Parsons, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", RFC 9794, DOI 10.17487/RFC9794, June 2025, <https://www.rfc-editor.org/info/rfc9794>.

Authors' Addresses

Lidia Pocero Fraile
ISI, R.C. ATHENA
Cyber-physical and Networked Embedded Systems
26504 Patras
Greece
Email: pocero@isi.gr

Christos Koulamas
ISI, R.C. ATHENA
Cyber-physical and Networked Embedded Systems
26504 Patras
Greece
Email: koulamas@isi.gr

Apostolos P. Fournaris
ISI, R.C. ATHENA
Security and Protection of Systems, Networks and Infrastructures
26504 Patras
Greece
Email: fournaris@isi.gr

Evangelos Haleplidis
ISI, R.C. ATHENA
Department of Digital Systems
26504 Patras
Greece
Email: haleplidis@isi.gr