individual L. Pocero Fraile
Internet-Draft C. Koulamas
Intended status: Standards Track A.P. Fournaris
Expires: 12 July 2026 E. Haleplidis
ISI, R.C. ATHENA
8 January 2026

KEM-based Authentication for EDHOC in Initiator-Known Responder (IKR) Scenarios
draft-lake-pocero-authkem-ikr-edhoc-01

Abstract

This document specifies a more efficient variant of a Key Encapsulation Mechanism (KEM)-based authentication method for the Ephemeral Diffie-Hellman Over COSE (EDHOC) lightweight protocol, designed for the specific scenario in which the Initiator has prior knowledge of the Responder’ s credentials, a case commonly found in constrained environments. Improving upon the approach described in KEM-based Authentication for EDHOC, this method uses only a mandatory three-message handshake to enable signature-free post-quantum authentication when PQC KEMs, such as the NIST-standardized ML-KEM, are employed, while still providing mutual authentication, forward secrecy, and a degree of identity protection.

Table of Contents

1. Introduction

The purpose of this document is to address a more efficient quantum-resistant transition of the Ephemeral Diffie-Hellman over COSE (EDHOC) protocol by extending with a new Key Encapsulation Mechanism (KEM)-based authentication method that uses a mandatory three-message handshake in Initiator-Known Responder (IKR) Scenarios.

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 KEM-based authentication method for EDHOC, specified in [I-D.pocero-authkem-edhoc], addresses the general mutually unknown peer scenario, similar to the original EDHOC protocol. In this case, the Initiator and Responder, if do not have prior knowledge of each other's credentials can exchange them in the form of X.509 certificate. To maintain this generality an additional round trip is required, resulting in a mandatory five-message handshake.

This document explores the possibility of reducing this overhead in the specific scenario where the Initiator already possesses the credentials of the Responder it wishes to connect in advance. This applies to cases where the Initiator is a constrained device equipped with credentials for the Responder, obtained through pre-provisioning or out-of-band methods. A typical example is during onboarding, where a remote or local server (acting as the Responder) is configured on the device in advance. Such settings are particularly relevant for EDHOC, which targets constrained environments where transmitting credentials can be costly.

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, tailored for a specific scenario where the Initiator knows the credentials of the Responder it intends to communicate with beforehand. In such cases, the method, specified in this document, reduces the standard five-message handshake defined in [I-D.pocero-authkem-edhoc] to a three-message exchange.

This variant, referred to as Initiator Knows Responder (IKR), converts the Noise-XX pattern, used in both static-DH and KEM-based authentication EDHOC methods, into a Noise-IK pattern. The Noise-IK pattern enables a mutual authentication handshake when the Initiator has prior knowledge of the Responder’ s static public key, and the Initiator’ s static keys are sent in the first message. The mechanism provided in [PQNoise-CCS22] for transforming this pattern into the PQ Noise-IK version is integrated into this protocol.

The KEM-based IKR variant provides a more efficient handshake with only three mandatory messages while maintaining mutual authentication, forward secrecy, and a level of identity protection. Notably, the Responder does not require prior knowledge of the Initiator’ s credentials. Instead, the Initiator encrypts its credentials, or an identifier, within message_1 using a key derived from a shared secret, computed over the Responder’ s static KEM public key, which the Initiator already possesses. Consequently, only the legitimate Responder, holding the corresponding static KEM private key, can decrypt this information, ensuring that the Initiator’ s identity remains protected against both passive and active attackers.

The KEM-based EDHOC protocol consists of three mandatory messages (message_1, message_2, message_3), an optional message_4, and an error message, between an Initiator (I) and a Responder (R). Figure 2 illustrates a KEM-based ikr 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.

```
Initiator                                                Responder
|  pk_eph, ct_R, AEAD( METHOD, SUITES_I, ID_CRED_I, C_I, EAD_1)  |
+--------------------------------------------------------------->
|                       message_1                                |
|                                                                |
|       ct_eph, ct_I, AEAD( C_R, ID_CRED_R, MAC_2, EAD_2 )       |
<---------------------------------------------------------------+
|                       message_2                                |
|                                                                |
|                  AEAD( MAC_3, EAD_3 )                          |
+--------------------------------------------------------------->
|                       message_3                                |
|                                                                |
|                      AEAD( EAD_4 )                             |
<----- - - -- - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                         message_4                              |
```

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

The Initiator can derive symmetric application keys after receiving message_2 and Responder after receiving message_3.

* 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].

* AEAD(), and MAC() denote 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 5; 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 are External Authorization Data included in message_1, message_2, message_3 and message_4, 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 ones defined in EDHOC. 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 encrypted in message_1 and message_2, respectively.

2.1.2.1. Method

The protocol extends EDHOC with a new KEM-based authentication method for IKR scenarios, where both parties use static KEM key pairs. The authentication is provided by a Message Authentication Code (MAC) included in message_2 and message_3 to authenticate the Responder and Initiator, respectively. The Initiator and Responder need to have agreed on a method 5.

Method Type Value	Initiator Authentication Key	Responder Authentication Key
5	Static KEM Key (IKR scenario)	Static KEM Key (IKR scenario)

Table 2: 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].

* "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, and ID_CRED_R SHOULD contain an identifier of the credentials, since in the specific IKR scenario, the Responder’ s credentials have been pre-provisioned or acquired out-of-band.

* "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, and ID_CRED_I may contain the authentication credential CRED_R or an identifier of the credentials if these have been pre-provisioned or acquired out-of-band.

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 for IKR scenarios compared to the KEM-based authentication method on the general case [I-D.pocero-authkem-edhoc] and with the original EDHOC authentication methods described in [RFC9528]. An overview of the EDHOC key schedule when using the KEM-based authentication in the IKR scenarios method is shown in Figure 2, and each key derivation step is defined in the following subsections.

```
             +----+
             |TH_1|
             +--+-+
                |
   +----+  +--++  +------+  +---+  +---+  +----+  +---+
   |ss_R|->|Ext|->|PRK_1e|--|Exp|->|K_1|->|AEAD|->|C_1|
   +----+  +---+  +--+---+  +--++  +---+  +--^-+  +---+
                     |         |             |
                 +---+         |          PLAINTEXT_1
                 |           +-------+-----------+
```

```
                    |           |TH_1=H(pk_eph,ct_R)|
                    |           +---------+---------+
                    |                     |
      +------+  +---+---+  +-------+   +--+---+  +-----+
      |ss_eph|->|Extract|->|PRK_2m |---|Expand|->|MAC_2|
      +------+  +-------+  +--+----+   +------+  +-----+
                              |
                    +---------+
      +----+  +---+  +----------+  +---+   +---+   +----+  +---+
      |ss_I|->|Ext|->|PRK_2e3e3m|+-|Exp|-->|K_2|-->|AEAD|->|C_2|
      +----+  +---+  +----------+ |+--++   +---+   +-^--+  +---+
                                  |   |              |
                                  |   |         PLAINTEXT_2
                                  | +--+---------------------------+
                                  | |TH_2=H(H(Message1),ct_eph,ct_I|
                                  | +--+---------------------------+
                                  | +--+---+   +-----+
                                  +-|Expand|-->|MAC_3|
                                  | +------+   +-----+
                                  | +-------------------------------+
                                  | |TH_3=H(TH_2,PLAINTEXT_2,CRED_R)|
                                  | +--+----------------------------+
                                  |    |
                                  | +--++    +---+   +----+  +---+
                                  +-|Exp|--->|K_3|-->|AEAD|->|C_3|
                                  | +---+    +---+   +-^--+  +---+
                                  |                    |
                                  |             PLAINTEXT_3
                                  |      +---+  +---+   +----+  +---+
                                  +-m_4?-|Exp|->|K_4|-->|AEAD|->|C_4|
                                  |      +--++  +---+   +-^--+  +---+
                                  |         |             |
                                  |         |         PLAINTEXT_4
                                  | +----------+--------------------+
                                  | |TH_4=H(TH_3,PLAINTEXT_3,CRED_I)|
                                  | +----+--------------------------+
                                  |      |
                                  |  +--+--+  +-------+
                                  +--|Expand|->|PRK_out|
                                     +------+  +--+----+
                                                  |
                                               +--v---+
                                               |Expand|
                                               +------+
                                                  |
                                             +----v-------+
                                             |PRK_exporter|
                                             +---+--------+
```

```
                                                |
                                            +--v---+
                                            |Expand|
                                            +--+---+
                                               |
                                               v
                                         Aplication Key
```

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 IKR 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. The usage of PRKs differs from the definitions in both [I-D.pocero-authkem-edhoc] and [RFC9528], and their names have been updated to reflect their new roles.

3.1.1.1. PRK_1e

The pseudorandom key PRK_1e is derived with the following input:

* The salt SHALL be TH_1.

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

When SHA-256 is used PRK_1e is produced as follows:

```
PRK_1e = HMAC-SHA-256( TH_1, ss_R )
```

Where the ss_R shared secret is the output of the following functions in the Initiator and the Responder respectively.

Initiator:

```
ss_R, ct_R <-  KEM.Encapsulate( pk_R )
```

Responder:

```
ss_R <-  KEM.Decapsulate( ct_R, sk_R )
```

3.1.1.2. PRK_2m

The pseudorandom key PRK_2m is derived with the following input:

* The salt SHALL be the SALT_2m derived from PRK_1e
* The IKM SHALL be the ephemeral KEM shared secret ss_eph

PRk_2m is derived as follows:

PRK_2m = EDHOC_Extract(SALT_2m, ss_eph)

Where the ephemeral KEM 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)

3.1.1.3. PRK_2e3e3m

The pseudorandom key PRK_2e3e3m is derived with the following input:

* The salt SHALL be the SALT_2e3m, derived from PRK_2m
* The IKM SHALL be the KEM shared secret ss_I, used to authenticate the Initiator

PRK_2e3e3m is derived as follows:

PRK_2e3e3m = EDHOC_Extract(SALT_2e3m, 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 some of the transcription hashes THs input contraction as specified in Section 4.

The OKMs, including keys, initialization vectors (IV), and salts derivations using EDHOC_KDF are shown in Figure 3.

The main difference from the original EDHOC shows in Section 4.1.2 of [RFC9528] Figure 6 are:

* A new K_1/IV_1 is computed to encrypt the message_1 which includes the ID_CRED_I initiator credentials, using a new transcrit hash (TH_1) defined as follows:

  TH_1 = H(pk_eph, ct_I)

  The AEAD encrption of message_1 with K_1/IV_1 provide integrity protection ensuring that the Initiator’ s identity is protected against active attacks. However, this does not provide authentication of the Initiator’ s identity.

* K_2/IV_2 are computed to encrypt and authnticate message_2, derived from the PRK computed over the three shared secrets (ss_eph, ss_I, and ss_R)

* K2/IV_2, K_3/IV_3, and K_4/IV_4 are derived from the same PRK_2e3e4m with different THs as Info.

* The usage of the pseudorandom keys (PRKs) has changed, and the salt names have been selected to reflect their new roles accordingly.

Further details of the key derivation and how the output keying material is used are specified in Section 4

```
K_1          = EDHOC_KDF(PRK_1e,      0,  TH_1,      plaintext_length)
IV_1         = EDHOC_KDF(PRK_1e,      1,  TH_1,      plaintext_length)
SALT_2m      = EDHOC_KDF(PRK_1e,      2,  TH_1,      hash_length)
MAC_2        = EDHOC_KDF(PRK_2m,      3,  context_2, mac_length_2)
SALT_2e3e3m  = EDHOC_KDF(PRK_2m,      4,  TH_2,      hash_length)
K_2          = EDHOC_KDF(PRK_2e3e3m,  5,  TH_2,      key_length)
IV_2         = EDHOC_KDF(PRK_2e3e3m,  6,  TH_2,      key_length)
MAC_3        = EDHOC_KDF(PRK_2e3e3m,  7,  context_3, mac_length_3)
K_3          = EDHOC_KDF(PRK_2e3e3m,  8,  TH_3,      key_length)
IV_3         = EDHOC_KDF(PRK_2e3e3m,  9,  TH_3,      key_length)
PRK_out      = EDHOC_KDF(PRK_2e3e3m,  10,  TH_4,      hash_length)
K_4          = EDHOC_KDF(PRK_2e3e3m,  11,  TH_4,      key_length)
IV_4         = EDHOC_KDF(PRK_2e3e3m,  12, TH_4,      iv_length)
PRK_exporter= EDHOC_KDF(PRK_out,     13, h’’,       hash_length)
```

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

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_2e3e3m, TH_4, hash_length)

The context include the trascription hash TH_4, defined as:

TH_4 = H(TH_3, PLAINTEXT_3, CRED_I)

Instead of reusing the last key-exchange internal key, the final key derivation depends on both PRK_2e3e3m and a newly computed TH_4, which include PLAINTEXT_3. This approach aims to ensure robust session keys that are distinct from the MAC keys and whose confirmation implies the authentication of all the handshake data.

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

The message_1 SHALL be a CBOR Sequence as defined below.

```
message_1 = (
  pk_eph : bstr,
  ct_R : bstr,
  CIPHERTEXT_1: bstr,
)
```

4.1.2. Initiator Composition of Message 1

The Initiator SHALL compose message_1 as following:

* Construct the following elements of PLAINTEX_1:

  - Chose KEM-based ikr authentication method 5.

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

  - Chose a connection identifier C_I and store it during the EDHOC session, as in Section 5.2.2 of [RFC9528]

* Encode PLAINTEXT_1 as a sequence of CBOR-encoded data items, as specified below:

  ```
  PLAINTEX_1 = (
    METHOD : int,
    SUITES_I : suites,
    ID_CRED_I : bstr/ -24..23, C_I,
    C_I : bstr / -24..23,
    ? EAD_1,
  )
  ```

* 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()
  ```

* Encapsulate the known beforehand static KEM public key of the Responder (pk_R) by calculating the corresponding ciphertext (ct_R) and shared secret (ss_R) with the following function:

ss_R, ct_R <- KEM.Encapsulate(pk_R)

* Compute the transcript hash TH_1 = H(pk_eph,ct_R)

* Compute the PRK_1e pseudorandom key from the static KEM shared secret (ss_R) used to authenticate the Responder.

* Compute K_1/IV1 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_1, the initialization vector IV_1 (if used by the AEAD algorithm), the plaintext PLAINTEXT_1, and the following parameters as input:

  - protected = h''

  - external_aad = TH_1

  - K_1 and IV_1 are defined in Section 3.1.2

  - PLAINTEXT_1 = (METHOD, SUITES_I, ID_CRED_I, C_I, ?EAD_2)

  CIPHERTEXT_1 is the 'ciphertext' of COSE_Encrypt0.

* Encode message_1 as a sequence of CBOR-encoded data items 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

2. Compute the KEM shared_secret (ss_R) for the authentication of the Responder by decapsulating the KEM ciphertext (ct_R) received in message_1 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)

3. Compute the transcript hash TH_1 = H(pk_eph,ct_R)

4. Compute the PRK_1e pseudorandom key from the static KEM shared secret (ss_R) used to authenticate the Responder.

5. Compute K_1/IV1 as in Section 3.1.2

6. Decrypt the COSE_Encrypt0 (CIPHERTEXT_1) 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 4.1.2.

7. Process PLAINTEXT_1 as specify in Section 5.2.3 of [RFC9528]

8. If all processing is completed successfully, then make ID_CRED_I and (if present) EAD_1 available to the application.

9. Obtain the authentication credential (CRED_I) from the (ID_CRED_I) and the static KEM authentication key (pk_I) of the Initiator

4.2. KEM-based authentication EDHOC Message 2

4.2.1. Formating of Message 2

The Initiator SHALL compose message_2 as following:

```
message_2 = (
  ct_eph : bstr,
  ct_I : bstr,
  CIPHERTEXT_2: bstr,
)
```

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)
  ```

* Choose a connection identifier C_R and store it for the length of the EDHOC session.

* Compute the PRK_2m pseudorandom key from both the static KEM shared secret (ss_R) and the latest 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(ct_eph, ct_I, H(message_1)).

* Compute MAC_2 as defined in Section 3.1.2, with context_2 =<< C_R, ID_CRED_R, TH_2, CRED_R, ? EAD_2 >>

  - The Responder authenticates with a PRK_2m derived from the KEM ephemeral shared secret and with the shared secret computed over its static KEM public 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_2 and the External Application Data included in message_2 (EAD_2) 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_2e3e3m from a chain that includes the KEM shared secret for the Authentication of the Responder (ss_R), the ephemeral KEM shared secret (ss_eph), 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_2/IV2 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_2, the initialization vector IV_2 (if used by the AEAD algorithm), the plaintext PLAINTEXT_2, and the following parameters as input:

  - protected = h’’

  - external_aad = TH_2

  - K_2 and IV_2 are defined in Section 3.1.2

  - PLAINTEXT_2 = (C_R, ID_CRED_R, MAC_2, ?EAD_2)

  CIPHERTEXT_2 is the ’ciphertext’ of COSE_Encrypt0.

* 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_2m pseudorandom key from both the static KEM shared secret (ss_R) and the latest ephemeral KEM shared secret (ss_eph)

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

6. Compute the KEM shared_secret (ss_I) for the authentication of the Initiator by decapsulating the KEM ciphertext (ct_I) received in message_2 using the Initiator 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 )
   ```

7. Compute the new PRK_2e3e3m from a chain that includes the KEM shared secret for the Authentication of the Responder (ss_R), the ephemeral KEM shared secret (ss_eph), and the latest KEM shared secret for the Authentication of the Initiator (ss_I) as defined in Section 3.1.1.3

8. Derive the session key K_2/IV2 as in Section 3.1.2.

9. Decrypt and verify the COSE_Encrypt0 (CIPHERTEXT_2) 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.

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

11. 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]

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 = (
  CIPHERTEXT_3 : bstr,
)
```

4.3.2. Initiator Composition of Message 3

The Initiator SHALL process the composition of message_3 as follows:

* Compute MAC_3 as defined in Section 3.1.2, with context_3 =<< C_I, ID_CRED_I, TH_2, CRED_I, ? EAD_3 >>

  - The Initaiator authenticate with a PRK_2e3e3m 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 lenght 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 trascript hash TH_3 and the External Aplication Data include it on Message 3 (EAD_3) are used it.

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

* Derive the new session key K_3/IV_3 as defined in Section 3.1.2.

* 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_3, the initialization vector IV_3 (if used by the AEAD algorithm), the plaintext PLAINTEXT_3, and the following parameters as input:

- protected = h''

- external_aad = TH_3

- K_3 and IV_3 are defined in Section 3.1.2

- PLAINTEXT_3 = (MAC_3, ?EAD_5)

CIPHERTEXT_3 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_3 as a CBOR data item as specified in Section 4.3.1

* Make the connection identifiers (C_I and C_R) and the application algorithms in the selected cipher suite available to the application.

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 using available message correlation; see Section 3.4.2 of [RFC9528].

3. Decrypt and verify the COSE_Encrypt0 (CIPHERTEXT_3) 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 4.3.2.

4. Verify MAC_3 as defined in Section 4.3.2, and make the result of the verification available to the application.

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

6. 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

4.4. KEM-based authentication EDHOC Message 4

This section specifies message_4, which is OPTIONAL to support. Confirmation of the latest pseudorandom key (PRK2e3e3m) is already provided by message_2 and message_3, which are encrypted with K_2/IV_2 and K_3/IV_3, respectively. Either message_4 or the first application message from the Responder to the Initiator, protected using a key derived from the EDHOC_Exporter, should also ensure the authenticity of all prior data. This is achieved through the use of a session key derived from PRK2e3e3m and the latest transcript hash TH_4.

4.4.1. Formating of Message 4

message_4 SHALL be a CBOR Sequence as defined below

```
message_4 = (
  CIPHERTEXT_4 : bstr,
)
```

4.4.2. Responder Composition of Message 4

The Responder SHALL process the composition of message_4 as follows:

* Derive the new session key K_4/IV_4 as defined 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 = (EAD_4)

  CIPHERTEXT_4 is the ’ciphertext’ of COSE_Encrypt0.

* 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; see Section 3.4.2 of [RFC9528].

3. 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.

4. Make (if present) EAD_4 available to the application for EAD processing

5. IANA Considerations

5.1. 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 2

Registry Name: EDHOC Method Types

Reference: draft-lake-pocero-authkem-ikr-edhoc-01

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
7 (suggested)	Static KEM Key (IKR)	Static KEM Key (IKR)	[draft-lake-pocero-authkem-ikr-edhoc-01]

Table 2: EDHOC Method Types

6. Security Considerations

The protocol design aligns with the fundamental PQNoise process described in [PQNoise-CCS22]. Specifically, the Initiator transmits its ephemeral public key in the first message, along with a key encapsulation targeting the Responder’s known static public key. The Initiator also includes its own credentials, unknown to the Responder, in the first message using ID_CRED_I. As defined in the I pattern of the Noise framework [Noise], this approach exposes the Initiator’s identity to a passive attacker. To mitigate this, the Initiator’s credentials are encrypted using a key derived from the Responder’s static public key, ensuring that only the intended Responder can decrypt the credentials.

Full forward secrecy and explicit mutual authentication are achieved once the KEM-based ikr EDHOC handshake is completed, as in the static-DH EDHOC handshake. While this mechanism preserves the Initiator’s anonymity against active attackers, the identity is protected only by the Responder public key, which makes it vulnerable if the Responder’s static key is later compromised, and vulnerable to be replayed.

This KEM-based authnetication method does not provide non-repudiation, but only implicit proof of participation as EDHOC with static DH keys. It also maintains an equivalent level of downgrade protection, as the negotiation base of the protocol is unchanged.

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-11, 22 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-kyber-certificates-11>.

[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.pocero-authkem-edhoc]
Fraile, L. P., Koulamas, C., Fournaris, A. P., and E. Haleplidis, "KEM-based Authentication for EDHOC", Work in Progress, Internet-Draft, draft-pocero-authkem-edhoc-01, 24 October 2025, <https://datatracker.ietf.org/doc/html/draft-pocero-authkem-edhoc-01>.

[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>.

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