Workgroup: CoRE Working Group
Internet-Draft:
draft-ietf-core-oscore-edhoc-04
Published: 11 July 2022
Intended Status: Standards Track
Expires: 12 January 2023
A   F. Palombini   M. Tiloca   R. Hoeglund
uthors: Ericsson   RISE AB   RISE AB

S. Hristozov   G. Selander
Fraunhofer AISEC   Ericsson

Profiling EDHOC for CoAP and OSCORE

Abstract

The lightweight authenticated key exchange protocol EDHOC can be run over CoAP and used by two peers to establish an OSCORE Security Context. This document further profiles this use of the EDHOC protocol, by specifying a number of additional and optional mechanisms. These especially include an optimization approach for combining the execution of EDHOC with the first subsequent OSCORE transaction. This combination reduces the number of round trips required to set up an OSCORE Security Context and to complete an OSCORE transaction using that Security Context.

Discussion Venues

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

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (core@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/core/.

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

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 12 January 2023.

Table of Contents

1. Introduction

Ephemeral Diffie-Hellman Over COSE (EDHOC) [I-D.ietf-lake-edhoc] is a lightweight authenticated key exchange protocol, especially intended for use in constrained scenarios. In particular, EDHOC

messages can be transported over the Constrained Application Protocol (CoAP) [RFC7252] and used for establishing a Security Context for Object Security for Constrained RESTful Environments (OSCORE) [RFC8613].

This document profiles this use of the EDHOC protocol, and specifies a number of additional and optional mechanisms. These especially include an optimization approach, that combines the EDHOC execution with the first subsequent OSCORE transaction (see Section 3). This allows for a minimum number of round trips necessary to setup the OSCORE Security Context and complete an OSCORE transaction, e.g., when an IoT device gets configured in a network for the first time.

This optimization is desirable, since the number of protocol round trips impacts on the minimum number of flights, which in turn can have a substantial impact on the latency of conveying the first OSCORE request, when using certain radio technologies.

Without this optimization, it is not possible, not even in theory, to achieve the minimum number of flights. This optimization makes it possible also in practice, since the last message of the EDHOC protocol can be made relatively small (see Section 1.2 of [I-D.ietf-lake-edhoc]), thus allowing additional OSCORE-protected CoAP data within target MTU sizes.

Furthermore, this document defines a number of parameters corresponding to different information elements of an EDHOC application profile (see Section 7). These can be specified as target attributes in the link to an EDHOC resource associated with that application profile, thus enabling an enhanced discovery of such resource for CoAP clients.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The reader is expected to be familiar with terms and concepts defined in CoAP [RFC7252], CBOR [RFC8949], CBOR sequences [RFC8742], OSCORE [RFC8613] and EDHOC [I-D.ietf-lake-edhoc].

2. EDHOC Overview

The EDHOC protocol allows two peers to agree on a cryptographic secret, in a mutually-authenticated way and by using Diffie-Hellman ephemeral keys to achieve forward secrecy. The two peers are denoted as Initiator and Responder, as the one sending or receiving the initial EDHOC message_1, respectively.

After successful processing of EDHOC message_3, both peers agree on a cryptographic secret that can be used to derive further security material, and especially to establish an OSCORE Security Context [RFC8613]. The Responder can also send an optional EDHOC message_4 to achieve key confirmation, e.g., in deployments where no protected application message is sent from the Responder to the Initiator.

Appendix A.2 of [I-D.ietf-lake-edhoc] specifies how to transfer EDHOC over CoAP. That is, the EDHOC data (referred to as "EDHOC messages") are transported in the payload of CoAP requests and responses. The default message flow consists in the CoAP Client acting as Initiator and the CoAP Server acting as Responder. Alternatively, the two roles can be reversed. In the rest of this document, EDHOC messages are considered to be transferred over CoAP.

Figure 1 shows a CoAP Client and Server running EDHOC as Initiator and Responder, respectively. That is, the Client sends a POST request to a reserved EDHOC resource at the Server, by default at the Uri-Path "/.well-known/edhoc". The request payload consists of the CBOR simple value "true" (0xf5) concatenated with EDHOC message_1, which also includes the EDHOC connection identifier C_I of the Client represented as per Section 3.3 of [I-D.ietf-lake-edhoc]. The Content-Format of the request may be set to application/cid-edhoc+cbor-seq.

This triggers the EDHOC exchange at the Server, which replies with a 2.04 (Changed) response. The response payload consists of EDHOC message_2, which also includes the EDHOC connection identifier C_R of the Server represented as per Section 3.3 of [I-D.ietf-lake-edhoc]. The Content-Format of the response may be set to application/edhoc+cbor-seq.

Finally, the Client sends a POST request to the same EDHOC resource used earlier to send EDHOC message_1. The request payload consists of the EDHOC connection identifier C_R represented as per Section 3.3 of [I-D.ietf-lake-edhoc], concatenated with EDHOC message_3. The Content-Format of the request may be set to application/cid-edhoc+cbor-seq.

After this exchange takes place, and after successful verifications as specified in the EDHOC protocol, the Client and Server can derive an OSCORE Security Context, as defined in Appendix A.1 of [I-D.ietf-lake-edhoc]. After that, they can use OSCORE to protect their communications as per [RFC8613].

The Client and Server are required to agree in advance on certain information and parameters describing how they should use EDHOC. These are specified in an application profile see Section 3.9 of [I-D.ietf-lake-edhoc], associated with the used EDHOC resource.

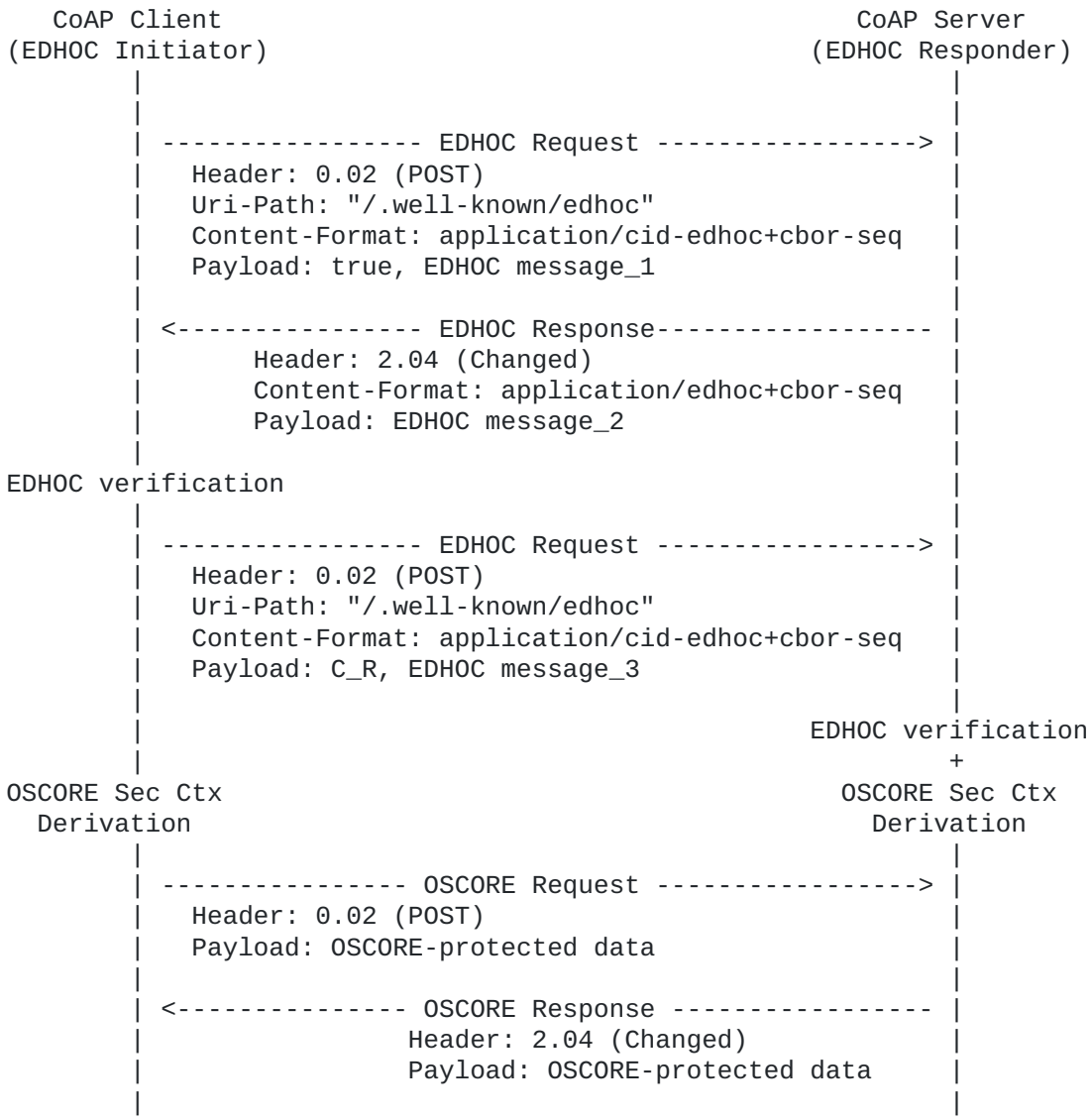

Figure 1: EDHOC and OSCORE run sequentially

As shown in Figure 1, this purely-sequential flow where EDHOC is run first and then OSCORE is used takes three round trips to complete.

Section 3 defines an optimization for combining EDHOC with the first subsequent OSCORE transaction. This reduces the number of round trips required to set up an OSCORE Security Context and to complete an OSCORE transaction using that Security Context.

3. EDHOC Combined with OSCORE

This section defines an optimization for combining the EDHOC exchange with the first subsequent OSCORE transaction, thus minimizing the number of round trips between the two peers.

This approach can be used only if the default EDHOC message flow is used, i.e., when the Client acts as Initiator and the Server acts as Responder, while it cannot be used in the case with reversed roles.

When running the purely-sequential flow of Section 2, the Client has all the information to derive the OSCORE Security Context already after receiving EDHOC message_2 and before sending EDHOC message_3.

Hence, the Client can potentially send both EDHOC message_3 and the subsequent OSCORE Request at the same time. On a semantic level, this requires sending two REST requests at once, as in Figure 2.

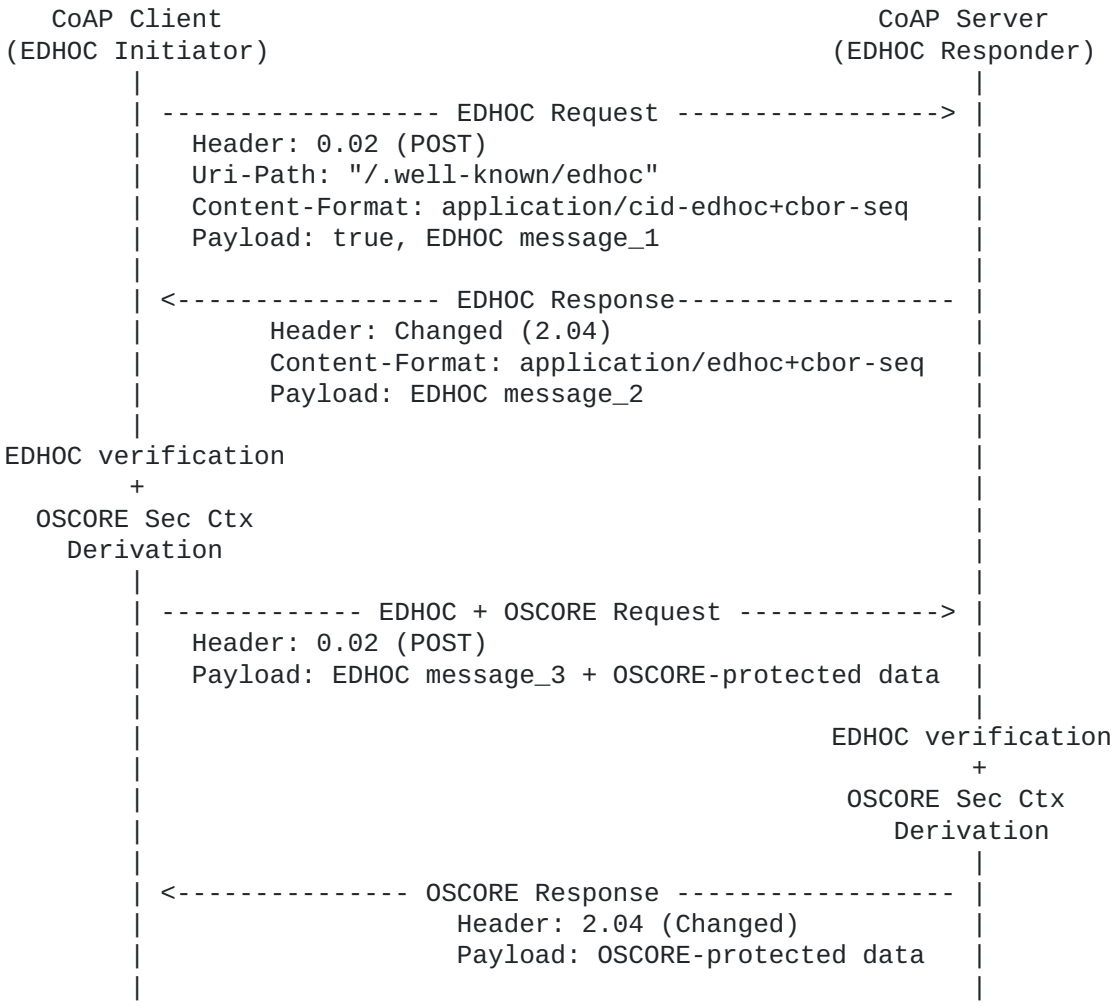

Figure 2: EDHOC and OSCORE combined

To this end, the specific approach defined in this section consists of sending a single EDHOC + OSCORE request, which conveys the pair (C_R, EDHOC message_3) within an OSCORE-protected CoAP message.

That is, the EDHOC + OSCORE request is in practice the OSCORE Request from Figure 1, as still sent to a protected resource and with the correct CoAP method and options intended for accessing that resource. At the same time, the EDHOC + OSCORE request also transports the pair (C_R, EDHOC message_3) required for completing the EDHOC exchange. Note that C_R is not transported precisely in the request payload.

Since EDHOC message_3 may be too large to be included in a CoAP Option, e.g., if conveying a protected large public key certificate

chain as ID_CRED_I (see Section 3.5.3 of [I-D.ietf-lake-edhoc]) or if conveying protected External Authorization Data as EAD_3 (see Section 3.8 of [I-D.ietf-lake-edhoc]), EDHOC message_3 has to be transported in the CoAP payload of the EDHOC + OSCORE request.

The rest of this section specifies how to transport the data in the EDHOC + OSCORE request and their processing order. In particular, the use of this approach is explicitly signalled by including an EDHOC Option (see Section 3.1) in the EDHOC + OSCORE request. The processing of the EDHOC + OSCORE request is specified in Section 3.2 for the Client side and in Section 3.3 for the Server side.

3.1. EDHOC Option

This section defines the EDHOC Option. The option is used in a CoAP request, to signal that the request payload conveys both an EDHOC message_3 and OSCORE-protected data, combined together.

The EDHOC Option has the properties summarized in Figure 3, which extends Table 4 of [RFC7252]. The option is Critical, Safe-to-Forward, and part of the Cache-Key. The option MUST occur at most once and is always empty. If any value is sent, the value is simply ignored. The option is intended only for CoAP requests and is of Class U for OSCORE [RFC8613].

No.	C	U	N	R	Name	Format	Length	Default
TBD21	x				EDHOC	Empty	0	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

Figure 3: The EDHOC Option.

The presence of this option means that the message payload contains also EDHOC data, that must be extracted and processed as defined in Section 3.3, before the rest of the message can be processed.

Figure 4 shows the format of a CoAP message containing both the EDHOC data and the OSCORE ciphertext, using the newly defined EDHOC option for signalling.

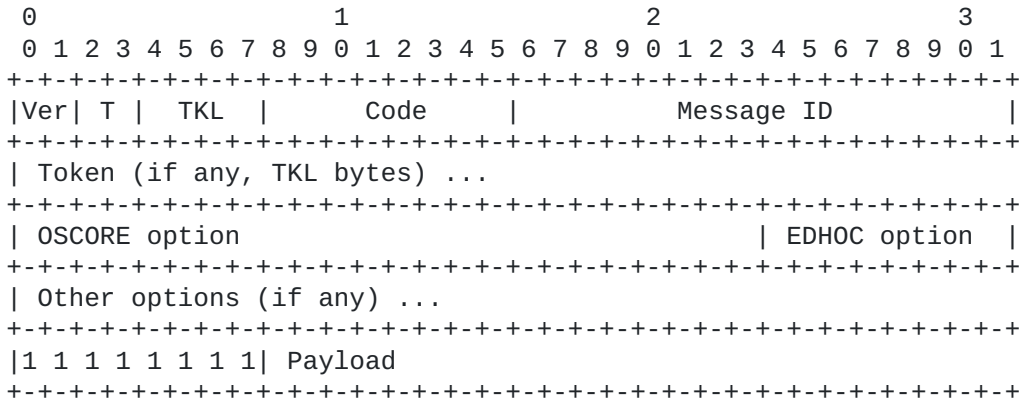

Figure 4: CoAP message for EDHOC and OSCORE combined - signalled with the EDHOC Option

3.2. Client Processing

The Client prepares an EDHOC + OSCORE request as follows.

1. Compose EDHOC message_3 as per Section 5.4.2 of [I-D.ietf-lake-edhoc].

2. Encrypt the original CoAP request as per Section 8.1 of [RFC8613], using the new OSCORE Security Context established after receiving EDHOC message_2.

   Note that the OSCORE ciphertext is not computed over EDHOC message_3, which is not protected by OSCORE. That is, the result of this step is the OSCORE Request as in Figure 1.

3. Build a CBOR sequence [RFC8742] composed of two CBOR byte strings in the following order.

   * The first CBOR byte string is the EDHOC message_3 resulting from step 1.

   * The second CBOR byte string has as value the OSCORE ciphertext of the OSCORE-protected CoAP request resulting from step 2.

4. Compose the EDHOC + OSCORE request, as the OSCORE-protected CoAP request resulting from step 2, where the payload is replaced with the CBOR sequence built at step 3.

   Note that the new payload includes EDHOC message_3, but it does not include the EDHOC connection identifier C_R. As the Client is the EDHOC Initiator, C_R is the OSCORE Sender ID of the Client, which is already specified as 'kid' in the OSCORE Option of the request from step 2, hence of the EDHOC + OSCORE request.

5. Signal the usage of this approach, by including the new EDHOC Option defined in Section 3.1 into the EDHOC + OSCORE request.

   The application/cid-edhoc+cbor-seq media type does not apply to this message, whose media type is unnamed.

6. Send the EDHOC + OSCORE request to the server.

With the same Server, the Client SHOULD NOT have multiple simultaneous outstanding interactions (see Section 4.7 of [RFC7252]) such that: they consist of an EDHOC + OSCORE request; and their EDHOC data pertain to the EDHOC session with the same connection identifier C_R.

3.2.1. Supporting Block-wise

If Block-wise [RFC7959] is supported, the Client may fragment the original CoAP request before protecting it with OSCORE, as defined in Section 4.1.3.4.1 of [RFC8613]. In such a case, the OSCORE

processing in step 2 of Section 3.2 is performed on each inner block of the original CoAP request, and the following also applies.

The Client takes the additional following step between steps 2 and 3 of Section 3.2.

A. If the OSCORE-protected request from step 2 conveys a non-first inner block of the original CoAP request (i.e., the Block1 Option processed at step 2 had NUM different than 0), then the Client skips the following steps and sends the OSCORE-protected request to the Server. In particular, the Client MUST NOT include the EDHOC Option in the OSCORE-protected request.

The Client takes the additional following step between steps 3 and 4 of Section 3.2.

B. If the size of the built CBOR sequence exceeds MAX_UNFRAGMENTED_SIZE (see Section 4.1.3.4.2 of [RFC8613]), the Client MUST stop processing the request and MUST abort the Block-wise transfer. Then, the Client can continue by switching to the purely sequential workflow shown in Figure 1. That is, the Client first sends EDHOC message_3 prepended by the EDHOC Connection Identifier C_R represented as per Section 3.3 of [I-D.ietf-lake-edhoc], and then sends the OSCORE-protected CoAP request once the EDHOC execution is completed.

Further considerations about the use of Block-wise together with the EDHOC + OSCORE request are provided in Section 6.

3.3. Server Processing

In order to process a request containing the EDHOC option, i.e., an EDHOC + OSCORE request, the Server MUST perform the following steps.

1. Check that the EDHOC + OSCORE request includes the OSCORE option and that the request payload is a CBOR sequence composed of two CBOR byte strings. If this is not the case, the Server MUST stop processing the request and MUST reply with a 4.00 (Bad Request) error response.

2. Extract EDHOC message_3 from the payload of the EDHOC + OSCORE request, as the first CBOR byte string in the CBOR sequence.

3. Take the value of 'kid' from the OSCORE option of the EDHOC + OSCORE request (i.e., the OSCORE Sender ID of the Client), and use it as the EDHOC connection identifier C_R.

4. Retrieve the correct EDHOC session by using the connection identifier C_R from step 3.

   If the application profile used in the EDHOC session specifies that EDHOC message_4 shall be sent, the Server MUST stop the EDHOC processing and consider it failed, as due to a client error.

   Otherwise, perform the EDHOC processing on the EDHOC message_3 extracted at step 2 as per Section 5.4.3 of [I-D.ietf-lake-

edhoc], based on the protocol state of the retrieved EDHOC session.

The application profile used in the EDHOC session is the same one associated with the EDHOC resource where the server received the request conveying EDHOC message_1 that started the session. This is relevant in case the server provides multiple EDHOC resources, which may generally refer to different application profiles.

5. Establish a new OSCORE Security Context associated with the client as per Appendix A.1 of [I-D.ietf-lake-edhoc], using the EDHOC output from step 4.

6. Extract the OSCORE ciphertext from the payload of the EDHOC + OSCORE request, as the value of the second CBOR byte string in the CBOR sequence.

7. Rebuild the OSCORE-protected CoAP request, as the EDHOC + OSCORE request where the payload is replaced with the OSCORE ciphertext extracted at step 6. Then, remove the EDHOC option.

8. Decrypt and verify the OSCORE-protected CoAP request rebuilt at step 7, as per Section 8.2 of [RFC8613], by using the OSCORE Security Context established at step 5.

   If the decrypted request includes an EDHOC option but it does not include an OSCORE option, the Server MUST stop processing the request and MUST reply with a 4.00 (Bad Request) error response.

9. Deliver the CoAP request resulting from step 8 to the application.

If steps 4 (EDHOC processing) and 8 (OSCORE processing) are both successfully completed, the Server MUST reply with an OSCORE-protected response (see Section 5.4.2 of [I-D.ietf-lake-edhoc]). The usage of EDHOC message_4 as defined in Section 5.5 of [I-D.ietf-lake-edhoc] is not applicable to the approach defined in this document.

If step 4 (EDHOC processing) fails, the server discontinues the protocol as per Section 5.4.3 of [I-D.ietf-lake-edhoc] and responds with an EDHOC error message with error code 1, formatted as defined in Section 6.2 of [I-D.ietf-lake-edhoc]. In particular, the CoAP response conveying the EDHOC error message MUST have Content-Format set to application/edhoc+cbor-seq defined in Section 9.9 of [I-D.ietf-lake-edhoc].

If step 4 (EDHOC processing) is successfully completed but step 8 (OSCORE processing) fails, the same OSCORE error handling as defined in Section 8.2 of [RFC8613] applies.

3.3.1. Supporting Block-wise

If Block-wise [RFC7959] is supported, the following applies, the Server takes the additional following step before any other in Section 3.3.

A. If Block-wise is present in the request, then process the Outer Block options according to [RFC7959], until all blocks of the request have been received (see Section 4.1.3.4 of [RFC8613]).

3.4. Example of EDHOC + OSCORE Request

Figure 5 shows an example of EDHOC + OSCORE Request. In particular, the example assumes that:

*The used OSCORE Partial IV is 0, consistently with the first request protected with the new OSCORE Security Context.

*The OSCORE Sender ID of the Client is 0x01.

As per Section 3.3.3 of [I-D.ietf-lake-edhoc], this straightforwardly corresponds to the EDHOC connection identifier C_R 0x01.

As per Section 3.3.2 of [I-D.ietf-lake-edhoc], when using the purely-sequential flow shown in Figure 1, the same C_R with value 0x01 would be represented on the wire as the CBOR integer 1 (0x01 in CBOR encoding), and prepended to EDHOC message_3 in the payload of the second EDHOC request.

*The EDHOC option is registered with CoAP option number 21.

o OSCORE option value: 0x090001 (3 bytes)

o EDHOC option value: - (0 bytes)

o EDHOC message_3: 0x52d5535f3147e85f1cfacd9e78abf9e0a81bbf (19 bytes)

o OSCORE ciphertext: 0x612f1092f1776f1c1668b3825e (13 bytes)

From there:

o Protected CoAP request (OSCORE message):

```
0x44025d1f               ; CoAP 4-byte header
  00003974               ; Token
  39 6c6f63616c686f7374  ; Uri-Host Option: "localhost"
  63 090001              ; OSCORE Option
  c0                     ; EDHOC Option
  ff 52d5535f3147e85f1cfacd9e78abf9e0a81bbf
     4d612f1092f1776f1c1668b3825e
(57 bytes)
```

Figure 5: Example of CoAP message with EDHOC and OSCORE combined

4. Use of EDHOC Connection Identifiers with OSCORE

Section 3.3.3 of [I-D.ietf-lake-edhoc] defines the straightforward mapping from an EDHOC connection identifier to an OSCORE Sender/Recipient ID. That is, an EDHOC identifier and the corresponding OSCORE Sender/Recipient ID are both byte strings with the same value.

Therefore, the conversion from an OSCORE Sender/Recipient ID to an EDHOC identifier is equally straightforward. In particular, at step 3 of Section 3.3, the value of 'kid' in the OSCORE Option of the EDHOC + OSCORE request is both the Server's Recipient ID (i.e., the Client's Sender ID) as well as the EDHOC Connection Identifier C_R of the Server.

4.1. Additional Processing of EDHOC Messages

Compared to what is specified in Section 5 of [I-D.ietf-lake-edhoc], the Client and Server MUST perform the additional message processing specified in the rest of this section.

4.1.1. Initiator Processing of Message 1

The Initiator selects C_I as follows. If the Initiator possibly performs multiple EDHOC executions concurrently, the following sequence of steps MUST be atomic.

1. The Initiator initializes a set ID_SET as the empty set.

2. The Initiator selects an available OSCORE Recipient ID, namely ID*, which is not included in ID_SET. Consistently with the requirements in Section 3.3 of [RFC8613], when selecting ID*:

   * The Initiator MUST NOT select a Recipient ID as ID* if this is currently used in a Recipient Context within a Security Context where the ID Context has zero-length.

   * The Initiator SHOULD select ID* only among the Recipient IDs which are currently not used in the sets of all its Recipient Contexts.

3. If ID* is already used as EDHOC Connection Identifier C_I, the Initiator adds ID* to ID_SET and moves back to step 2. Otherwise, it moves to step 4.

4. The Initiator sets ID* as a "not available" OSCORE Recipient ID, and uses it as its EDHOC connection identifier C_I.

4.1.2. Responder Processing of Message 2

The Responder selects C_R as follows. If the Responder possibly performs multiple EDHOC executions concurrently, the following sequence of steps MUST be atomic.

1. The Responder initializes a set ID_SET as the empty set.

2. The Responder selects an available OSCORE Recipient ID, namely ID*, which is not included in ID_SET. Consistently with the requirements in Section 3.3 of [RFC8613], when selecting ID*:

   * The Responder MUST NOT select a Recipient ID as ID* if this is currently used in a Recipient Context within a Security Context where the ID Context has zero-length.

The Responder SHOULD select ID only among the Recipient IDs which are currently not used in the sets of all its Recipient Contexts.

3. If ID* is already used as EDHOC Connection Identifier C_R, the Responder adds ID* to ID_SET and moves back to step 2. Otherwise, it moves to step 5.

4. If ID* is equal to the EDHOC Connection Identifier C_I specified in EDHOC message_1 (i.e., after its decoding as per Section 3.3 of [I-D.ietf-lake-edhoc]), then the Responder adds ID* to ID_SET and moves back to step 2. Otherwise, it moves to step 5.

5. The Responder sets ID* as a "not available" OSCORE Recipient ID, and uses it as its EDHOC connection identifier C_R.

4.1.3. Initiator Processing of Message 2

If the following condition holds, the Initiator MUST discontinue the protocol and reply with an EDHOC error message with error code 1, formatted as defined in Section 6.2 of [I-D.ietf-lake-edhoc].

*The EDHOC Connection Identifier C_I is equal to the EDHOC Connection Identifier C_R specified in EDHOC message_2 (i.e., after its decoding as per Section 3.3 of [I-D.ietf-lake-edhoc]).

5. Extension and Consistency of Application Profiles

The application profile referred by the Client and Server can include the information elements introduced below, in accordance with the specified consistency rules.

If the Server supports the EDHOC + OSCORE request within an EDHOC execution started at a certain EDHOC resource, then the application profile associated with that resource:

*MUST NOT specify that EDHOC message_4 shall be sent.

*SHOULD explicitly specify support for the EDHOC + OSCORE request.

6. Considerations on Using Block-wise

This section provides guidelines and recommendations for Clients supporting both the EDHOC + OSCORE request defined in this document as well as Block-wise [RFC7959].

The following especially considers a Client that may perform only "inner" Block-wise, but not "outer" Block-wise operations. That is, the considered Client does not (further) split an OSCORE-protected request like an intermediary (e.g., a proxy) might do. This is the typical case for OSCORE endpoints (see Section 4.1.3.4 of [RFC8613]).

The rest of this section refers to the following notation.

- *SIZE_APP: the size in bytes of the application data to be included in a CoAP request. When Block-wise is used, this is referred to as the "body" to be fragmented into blocks.

- *SIZE_EDHOC: the size in bytes of EDHOC message_3, if this is sent as part of the EDHOC + OSCORE request. Otherwise, the size of EDHOC message_3 plus the size in bytes of the EDHOC Connection Identifier C_R, represented as per Section 3.3 of [I-D.ietf-lake-edhoc].

- *SIZE_MTU: the maximum amount of transmittable bytes before having to use Block-wise. This is, for example, 64 KiB as maximum datagram size when using UDP, or 1280 bytes as the maximum size for an IPv6 MTU.

- *SIZE_OH: the size in bytes of the overall overhead due to all the communication layers underlying the application. This takes into account also the overhead introduced by the OSCORE processing.

- *LIMIT = (SIZE_MTU - SIZE_OH): the practical maximum size in bytes to be considered by the application before using Block-wise.

- *SIZE_BLOCK: the size in bytes of inner blocks.

- *ceil(): the ceiling function.

6.1. Pre-requirements

Before sending an EDHOC + OSCORE request, the Client has to perform the following checks. Note that, while the Client is able to fragment the application data, it cannot fragment the EDHOC + OSCORE request or the EDHOC message_3 added therein.

- *If inner Block-wise is not used, hence SIZE_APP <= LIMIT, the Client must verify whether all the following conditions hold:

  - -COND1: SIZE_EDHOC <= LIMIT

  - -COND2: (SIZE_APP + SIZE_EDHOC) <= LIMIT

- *If inner Block-wise is used, the Client must verify whether all the following conditions hold:

  - -COND3: SIZE_EDHOC <= LIMIT

  - -COND4: (SIZE_BLOCK + SIZE_EDHOC) <= LIMIT

In either case, if not all the corresponding conditions hold, the Client MUST NOT send the EDHOC + OSCORE request. Instead, the Client can continue by switching to the purely sequential workflow shown in Figure 1. That is, the Client first sends EDHOC message_3 prepended by the EDHOC Connection Identifier C_R represented as per Section 3.3 of [I-D.ietf-lake-edhoc], and then sends the OSCORE-protected CoAP request once the EDHOC execution is completed.

6.2. Effectively Using Block-Wise

In order to avoid further fragmentation at lower layers when sending an EDHOC + OSCORE request, the Client has to use inner Block-wise if *any* of the following conditions holds:

- *COND5: SIZE_APP > LIMIT

- *COND6: (SIZE_APP + SIZE_EDHOC) > LIMIT

In particular, consistently with Section 6.1, the used SIZE_BLOCK has to be such that the following condition also holds:

- *COND7: (SIZE_BLOCK + SIZE_EDHOC) <= LIMIT

Note that the Client might still use Block-wise due to reasons different from exceeding the size indicated by LIMIT.

If *both* the conditions COND5 and COND6 hold, the use of Block-wise results in the following number of round trips for completing both the EDHOC execution and the first OSCORE-protected exchange.

- *If the original workflow shown in Figure 1 is used, the number of round trips RT_ORIG is equal to 1 + ceil(SIZE_EDHOC / SIZE_BLOCK) + ceil(SIZE_APP / SIZE_BLOCK).

- *If the optimized workflow shown in Figure 2 is used, the number of round trips RT_COMB is equal to 1 + ceil(SIZE_APP / SIZE_BLOCK).

It follows that RT_COMB < RT_ORIG, i.e., the optimized workflow always yields a lower number of round trips.

Instead, the conveniency of using the optimized workflow becomes questionable if *both* the following conditions hold:

- *COND8: SIZE_APP <= LIMIT

- *COND9: (SIZE_APP + SIZE_EDHOC) > LIMIT

That is, since SIZE_APP <= LIMIT, using Block-wise would not be required when using the original workflow, provided that SIZE_EDHOC <= LIMIT still holds.

At the same time, using the combined workflow is in itself what actually triggers the use of blockwise, since (SIZE_APP + SIZE_EDHOC) > LIMIT.

Therefore, the following round trips are experienced by the Client.

- *The original workflow shown in Figure 1 and run without using Block-wise results in a number of round trips RT_ORIG equal to 3.

- *The optimized workflow shown in Figure 2 and run using Block-wise results in a number of round trips RT_COMB equal to 1 + ceil(SIZE_APP / SIZE_BLOCK).

It follows that RT_COMB >= RT_ORIG, i.e., the optimized workflow might still be not worse than the original workflow in terms of round trips. This is the case only if the used SIZE_BLOCK is such that ceil(SIZE_APP / SIZE_BLOCK) is equal to 2, i.e., the EDHOC + OSCORE request is fragmented into only 2 inner blocks. However, even in such a case, there would be no advantage in terms or round trips compared to the original workflow, while still requiring the Client and Server to perform the processing due to using the EDHOC + OSCORE request and Block-wise transferring.

Therefore, if both the conditions COND8 and COND9 hold, the Client SHOULD NOT send the EDHOC + OSCORE request. Instead, the Client SHOULD continue by switching to the purely sequential workflow shown in Figure 1. That is, the Client first sends EDHOC message_3 prepended by the EDHOC Connection Identifier C_R represented as per Section 3.3 of [I-D.ietf-lake-edhoc], and then sends the OSCORE-protected CoAP request once the EDHOC execution is completed.

7. Web Linking

Section 9.10 of [I-D.ietf-lake-edhoc] registers the resource type "core.edhoc", which can be used as target attribute in a web link [RFC8288] to an EDHOC resource, e.g., using a link-format document [RFC6690]. This enables Clients to discover the presence of EDHOC resources at a Server, possibly using the resource type as filter criterion.

At the same time, the application profile associated with an EDHOC resource provides a number of information describing how the EDHOC protocol can be used through that resource. While a Client may become aware of the application profile through several means, it would be convenient to obtain its information elements upon discovering the EDHOC resources at the Server. This might aim at discovering especially the EDHOC resources whose associated application profile denotes a way of using EDHOC which is most suitable to the Client, e.g., with EDHOC cipher suites or authentication methods that the Client also supports or prefers.

That is, it would be convenient that a Client discovering an EDHOC resource contextually obtains relevant pieces of information from the application profile associated with that resource. The resource discovery can occur by means of a direct interaction with the Server, or instead by means of the CoRE Resource Directory [RFC9176], where the Server may have registered the links to its resources.

In order to enable the above, this section defines a number of parameters, each of which can be optionally specified as a target attribute with the same name in the link to the respective EDHOC resource, or as filter criteria in a discovery request from the Client. When specifying these parameters in a link to an EDHOC resource, the target attribute rt="core.edhoc" MUST be included, and the same consistency rules defined in Section 5 for the corresponding information elements of an application profile MUST be followed.

The following parameters are defined.

* 'method', specifying an authentication method supported by the Server. This parameter MUST specify a single value, which is taken from the 'Value' column of the "EDHOC Method Type" registry defined in Section 9.3 of [I-D.ietf-lake-edhoc]. This parameter MAY occur multiple times, with each occurrence specifying a different authentication method.

* 'csuite', specifying an EDHOC cipher suite supported by the Server. This parameter MUST specify a single value, which is taken from the 'Value' column of the "EDHOC Cipher Suites" registry defined in Section 9.2 of [I-D.ietf-lake-edhoc]. This parameter MAY occur multiple times, with each occurrence specifying a different cipher suite.

* 'cred_t', specifying a type of authentication credential supported by the Server. This parameter MAY occur multiple times, with each occurrence specifying a different authentication credential type. Possible values are: "x509", for X.509 certificate [RFC5280]; "c509", for C509 certificate [I-D.ietf-cose-cbor-encoded-cert]; "cwt" for CWT [RFC8392]; "ccs" for CWT Claims Set (CCS) [RFC8392].

* 'idcred_t', specifying the type of identifiers supported by the Server for identifying authentication credentials. This parameter MUST specify a single value, which is taken from the 'Label' column of the "COSE Headers Parameters" registry [COSE.Header.Parameters]. This parameter MAY occur multiple times, with each occurrence specifying a different type of identifier for authentication credentials.

  Note that the values in the 'Label' column of the "COSE Headers Parameters" registry are strongly typed. On the contrary, Link Format is weakly typed and thus does not distinguish between, for instance, the string value "-10" and the integer value -10. Thus, if responses in Link Format are returned, string values which look like an integer are not supported. Therefore, such values MUST NOT be used in the 'idcred_t' parameter.

* 'ead_1', 'ead_2', 'ead_3' and 'ead_4', specifying, if present, that the Server supports the use of External Authorization Data EAD_1, EAD_2, EAD_3 and EAD_4, respectively (see Section 3.8 of [I-D.ietf-lake-edhoc]). For each of these parameters, the following applies.

  - It MAY occur multiple times, with its presence denoting support from the server for the respective external authorization data.

  - Each occurrence specifies a value taken from the 'Label' column of the "EDHOC External Authorization Data" registry defined in Section 9.5 of [I-D.ietf-lake-edhoc], thus denoting support from the server for that particular type of external authorization data.

* 'comb_req', specifying, if present, that the server supports the EDHOC + OSCORE request defined in Section 3. A value MUST NOT be

given to this parameter and any present value MUST be ignored by parsers.

The example in Figure 6 shows how a Client discovers two EDHOC resources at a Server, obtaining information elements from the respective application profiles. The Link Format notation from Section 5 of [RFC6690] is used.

```
REQ: GET /.well-known/core

RES: 2.05 Content
    </sensors/temp>;osc,
    </sensors/light>;if="sensor",
    </edhoc/resA>;rt="core.edhoc";csuite="0";csuite="2";method="0";
    cred_t="c509";cred_t="ccs";idcred_t="4";comb_req,
    </edhoc/resB>;rt="core.edhoc";csuite="0";csuite="2";method="0";
    method="3";cred_t="c509";cred_t="x509";idcred_t="34"
```

Figure 6: The Web Link

8. Security Considerations

The same security considerations from OSCORE [RFC8613] and EDHOC [I-D.ietf-lake-edhoc] hold for this document. In addition, the following considerations also apply.

Section 3.2 defines that a Client SHOULD NOT have multiple outstanding EDHOC + OSCORE requests pertaining to the same EDHOC session. Even if a Client did not fulfill this requirement, it would not have any impact in terms of security. That is, the Server would still not process different instances of the same EDHOC message_3 more than once in the same EDHOC session (see Section 5.1 of [I-D.ietf-lake-edhoc]), and would still enforce replay protection of the OSCORE-protected request (see Section 7.4 of [RFC8613]).

TODO: more considerations

9. IANA Considerations

This document has the following actions for IANA.

Note to RFC Editor: Please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

9.1. CoAP Option Numbers Registry

IANA is asked to enter the following option number to the "CoAP Option Numbers" registry within the "CoRE Parameters" registry group.

[

The CoAP option numbers 13 and 21 are both consistent with the properties of the EDHOC Option defined in Section 3.1, and they both

allow the EDHOC Option to always result in an overall size of 1 byte. This is because:

- The EDHOC option is always empty, i.e., with zero-length value; and

- Since the OSCORE option with option number 9 is always present in the CoAP request, the EDHOC option would be encoded with a maximum delta of 4 or 12, depending on its option number being 13 or 21.

At the time of writing, the CoAP option numbers 13 and 21 are both unassigned in the "CoAP Option Numbers" registry, as first available and consistent option numbers for the EDHOC option.

This document suggests 21 (TBD21) as option number to be assigned to the new EDHOC option, since both 13 and 21 are consistent for the use case in question, but different use cases or protocols may make better use of the option number 13.

]

Number	Name	Reference
TBD21	EDHOC	[RFC-XXXX]

10. References

10.1. Normative References

[COSE.Header.Parameters] IANA, "COSE Header Parameters", <https://www.iana.org/assignments/cose/cose.xhtml#header-parameters>.

[I-D.ietf-lake-edhoc] Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-15, 10 July 2022, <https://www.ietf.org/archive/id/draft-ietf-lake-edhoc-15.txt>.

[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>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/info/rfc6690>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959,

DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

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

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

[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>.

[RFC9176] Amsüss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <https://www.rfc-editor.org/info/rfc9176>.

10.2. Informative References

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-04, 10 July 2022, <https://www.ietf.org/archive/id/draft-ietf-cose-cbor-encoded-cert-04.txt>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[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>.

Appendix A. Document Updates

RFC Editor: Please remove this section.

A.1. Version -03 to -04

*Renamed "applicability statement" to "application profile".

*Use the latest Content-Formats.

*Use of SHOULD NOT for multiple simultaneous outstanding interactions.

*No more special conversion from OSCORE ID to EDHOC ID.

*Considerations on using Block-wise.

*Wed Linking signaling of multiple supported EAD labels.

*Added security considerations.

*Editorial improvements.

A.2. Version -02 to -03

*Clarifications on transporting EDHOC message_3 in the CoAP payload.

*At most one simultaneous outstanding interaction as an EDHOC + OSCORE request with the same server for the same session with connection identifier C_R.

*The EDHOC option is removed from the EDHOC + OSCORE request after processing the EDHOC data.

*Added explicit constraints when selecting a Recipient ID as C_X.

*Added processing steps for when Block-wise is used.

*Improved error handling on the Server.

*Improved section on Web Linking.

*Updated figures; editorial improvements.

A.3. Version -01 to -02

*New title, abstract and introduction.

*Restructured table of content.

*Alignment with latest format of EDHOC messages.

*Guideline on ID conversions based on application profile.

*Clarifications, extension and consistency on application profile.

*Section on web-linking.

*RFC8126 terminology in IANA considerations.

*Revised Appendix "Checking CBOR Encoding of Numeric Values".

A.4. Version -00 to -01

*Improved background overview of EDHOC.

*Added explicit rules for converting OSCORE Sender/Recipient IDs to EDHOC connection identifiers following the removal of bstr_identifier from EDHOC.

*Revised section organization.

*Recommended number for EDHOC option changed to 21.

*Editorial improvements.

Acknowledgments

The authors sincerely thank Christian Amsüss, Esko Dijk, Klaus Hartke, Jim Schaad and Mališa Vučinić for their feedback and comments.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; and by the H2020 project SIFIS-Home (Grant agreement 952652).

Authors' Addresses

Francesca Palombini
Ericsson

Email: francesca.palombini@ericsson.com

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Stockholm Kista
Sweden

Email: marco.tiloca@ri.se

Rikard Hoeglund
RISE AB
Isafjordsgatan 22
SE-16440 Stockholm Kista
Sweden

Email: rikard.hoglund@ri.se

Stefan Hristozov
Fraunhofer AISEC

Email: stefan.hristozov@eriptic.com

Goeran Selander
Ericsson

Email: goran.selander@ericsson.com