LAKE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

G. Selander
J. Preu Mattsson
Ericsson AB
M. Vuini
G. Fedrecheski
INRIA
M. Richardson
Sandelman Software Works

2 March 2026

Lightweight Authorization using Ephemeral Diffie-Hellman Over COSE (ELA) draft-ietf-lake-authz-07

Abstract

Ephemeral Diffie-Hellman Over COSE (EDHOC) is a lightweight authenticated key exchange protocol intended for use in constrained scenarios. This document specifies Lightweight Authorization using EDHOC (ELA). The procedure allows authorizing enrollment of new devices using the extension point defined in EDHOC. ELA is applicable to zero-touch onboarding of new devices to a constrained network leveraging trust anchors installed at manufacture time.

About This Document

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

The latest revision of this draft can be found at https://lake-wg.github.io/authz/draft-ietf-lake-authz.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-lake-authz/.

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

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

Status of This Memo

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

Copyright Notice

Table of Contents

1. Introduction

For constrained IoT deployments [RFC7228] the overhead and processing contributed by security protocols may be significant, which motivates the specification of lightweight protocols that are optimizing, in particular, message overhead (see [I-D.ietf-lake-reqs]). This document describes Lightweight Authorization using EDHOC (ELA), a procedure for augmenting the lightweight authenticated Diffie-Hellman key exchange EDHOC [RFC9528] with third party-assisted authorization.

ELA involves a device, a domain authenticator, and an enrollment server. The device and domain authenticator perform mutual authentication and authorization, assisted by the enrollment server that provides relevant authorization information to the device (a "voucher") and to the authenticator. The high-level model is similar to BRSKI [RFC8995].

In this document, we consider the target interaction for which authorization is needed to be "enrollment", for example joining a network for the first time (e.g., [RFC9031]), but it can be applied to authorize other target interactions.

The enrollment server may represent the manufacturer of the device, or some other party with information about the device from which a trust anchor has been pre-provisioned into the device. The (domain) authenticator may represent the service provider or some other party controlling access to the network in which the device is enrolling.

ELA assumes that authentication between device and authenticator is performed with EDHOC [RFC9528], and defines the integration of a lightweight authorization procedure using the External Authorization Data (EAD) fields defined in EDHOC.

ELA enables a low message count by performing authorization and enrollment in parallel with authentication, instead of in sequence, which is common for network access. It further reuses protocol elements from EDHOC, leading to reduced message sizes on constrained links.

This protocol is applicable to a wide variety of settings, and can be mapped to different authorization architectures.

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.

Readers are expected to have an understanding of CBOR [RFC8949], CDDL [RFC8610], and EDHOC [RFC9528]. Appendix C.1 of [RFC9528] contains some basic info about CBOR.

2. Protocol Outline

The goal of ELA is to enable a (potentially constrained) device (U) to enroll into a domain over a constrained link. The device authenticates and enforces authorization of the (non-constrained) domain authenticator (V) with the help of a voucher conveying authorization information. The voucher has a similar role as in [RFC8366] but should be considerably more compact. The domain authenticator, in turn, authenticates the device and authorizes its enrollment into the domain. ELA also enables scenarios where only V needs to authorize U, by allowing the voucher to be optional.

The procedure is assisted by a (non-constrained) enrollment server (W) located in a non-constrained network behind the domain authenticator, e.g., on the Internet, providing information to the device (conveyed in the voucher) and to the domain authenticator as part of the protocol.

The objective of this document is to specify such a protocol that is lightweight over the constrained link, by reusing elements of EDHOC [RFC9528] and by shifting message overhead to the non-constrained side of the network. See illustration in Figure 1.

Note the cardinality of the involved parties. It is expected that the domain authenticator needs to handle a large unspecified number of devices, but for a given device type or manufacturer it is expected that one or a few nodes host enrollment servers.

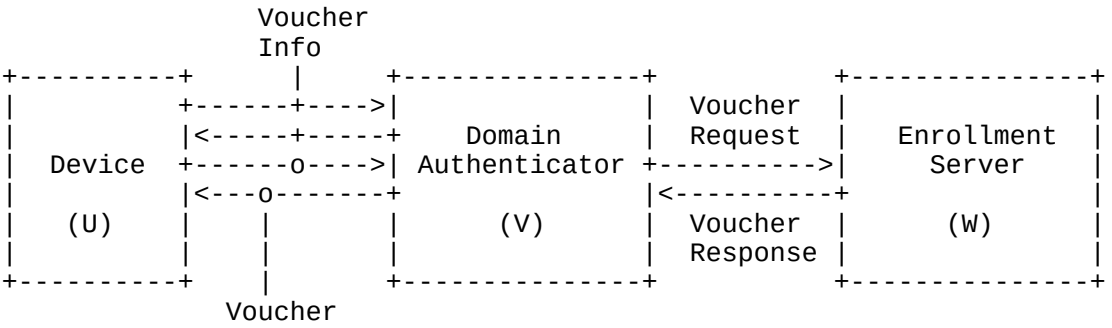


Figure 1: Overview of the message flow. EDHOC is used on the constrained link between U and V. Voucher Info and Voucher are sent in EDHOC External Authorization Data (EAD). The link between V and W is not constrained.

3. Assumptions

The protocol is based on the following pre-existing relations between the device (U), the domain authenticator (V) and the enrollment server (W), see Figure 2.

* U and W have an explicit relation: U is configured with a public key of W, see Section 3.1.

* V and W have an implicit relation, e.g., based on web PKI with trusted CA certificates, see Section 3.2.

* U and V need not have any previous relation. This protocol establishes a relation between U and V.

Each of the three parties can gain protected communication with the other two during the protocol.

V may be able to access credentials over non-constrained networks, but U may be limited to constrained networks. Implementations wishing to leverage the zero-touch capabilities of this protocol are expected to support transmission of credentials from V to U by value during the EDHOC exchange, which will impact the message size depending on the type of credential used.

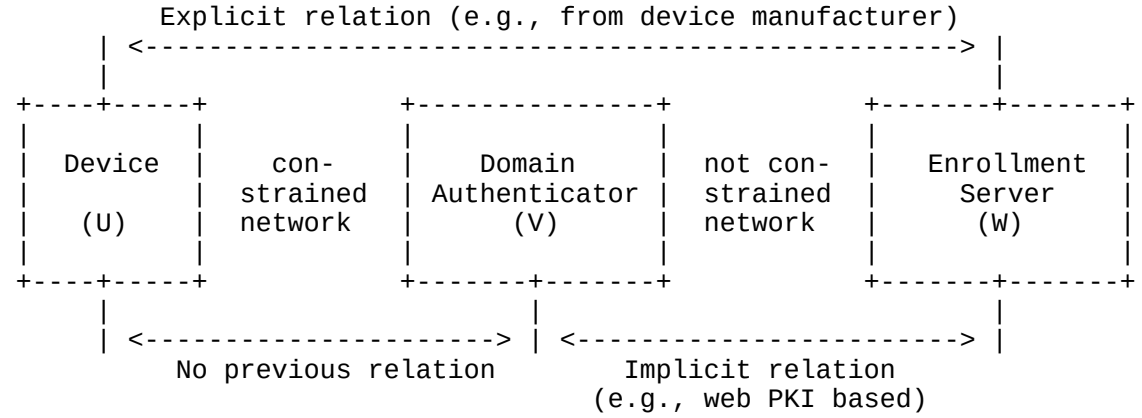


Figure 2: Overview of pre-existing relations.

3.1. Device (U)

To authenticate to V, the device (U) runs EDHOC in the role of Initiator with authentication credential CRED_U, for example, an X.509 certificate [RFC5280] or a CBOR Web Token (CWT, [RFC8392]). CRED_U may, for example, be carried by value in ID_CRED_I of EDHOC message_3 or be provisioned to V over a non-constrained network, leveraging a credential identifier in ID_CRED_I (see Figure 3).

U also needs to identify itself to W, for which it reuses ID_CRED_I. This means that the value used by U in ID_CRED_I needs to be unique enough to be understood by both V and W. W will use ID_CRED_I to determine if the device with this identifier is authorized to enroll with V.

U is also provisioned with information about W:

* A static public key of W (PK_W) used to establish secure communication with the enrollment server (see Section 4.3).

* Location information about the enrollment server (LOC_W) that can be used by V to reach W. This is typically a URI but may alternatively be only the domain name.

3.2. Domain Authenticator (V)

To authenticate to U, the domain authenticator (V) runs EDHOC in the role of Responder with an authentication credential CRED_V containing a public key of V, see Section 4.4.2.1. This proves to U the possession of the private key corresponding to the public key of CRED_V. CRED_V typically needs to be transported to U in EDHOC (using ID_CRED_R = CRED_V, see Section 3.5.2 of [RFC9528]) since there is no previous relation between U and V.

V and W need to establish a secure (confidentiality and integrity protected) connection for the Voucher Request/Response protocol. Furthermore, W needs to access the same credential CRED_V that V uses with U (to compute the Voucher), and V needs to prove to W the possession of the private key corresponding to the public key of CRED_V. It is RECOMMENDED that V authenticates to W using the same credential CRED_V as with U.

Note that the choice of protocols may affect which type of credential and methods should be used by V. For example, in case V and W select TLS for the secure channel and PoP, then CRED_V is a X.509 certificate, and the EDHOC method used by V is signature-based. Some of the possible combinations of protocols to secure the connection between V and W are listed in Table 1 below.

Secure channel between V and W	Proof-of-Possession from V to W	Type of CRED_V	EDHOC method used by V
[D]TLS 1.3 with mutual authentication, where V is the client and W is the server.	Provided by [D]TLS.	Restricted to types that are supported by both [D]TLS and EDHOC, e.g., X.509 certificates.	V MUST authenticate using a signature.
[D]TLS 1.3, where V is the client and W is the server.	Run an EDHOC session on top of the TLS-protected channel.	Any type supported by EDHOC, e.g., X.509, C509, CWT, or CCS.	Any method may be used.
EDHOC and OSCORE, where V is the initiator and W is the responder.	Already provided by EDHOC during the setup of the secure channel.	Any type supported by EDHOC.	Any method may be used.

Table 1: Examples of how to secure the connection between V and W.

Note also that the secure connection between V and W may be long-lived and reused for multiple voucher requests/responses.

Other details of proof-of-possession related to CRED_V and transport of CRED_V are out of scope of this document.

3.3. Enrollment Server (W)

The enrollment server (W) is assumed to have the private key corresponding to PK_W, which is used to establish secure communication with the device (see Section 4.3). W provides to U the authorization decision for enrollment with V in the form of a voucher (see Section 4.3.2). Authorization policies are out of scope for this document.

Authentication credentials and communication security with V is described in Section 3.2. To calculate the voucher, W needs access to the hash of EDHOC message_2 and message_1 (H_21), ID_CRED_I, and CRED_V as used in the EDHOC session between U and V, see Section 4.3.2.

* W MUST verify that CRED_V is bound to the secure connection between W and V

* W MUST verify that V is in possession of the private key corresponding to the public key of CRED_V

W needs to be available during the execution of the protocol between U and V.

4. The Protocol

4.1. Overview

The ELA protocol consists of three security sessions going on in parallel:

1. The EDHOC session between device (U) and (domain) authenticator (V)

2. Voucher Request/Response between authenticator (V) and enrollment server (W)

3. An exchange of voucher-related information, including the voucher itself, between device (U) and enrollment server (W), mediated by the authenticator (V).

Figure 3 provides an overview of the message flow detailed in this section. An outline of EDHOC is given in Section 2 of [RFC9528].

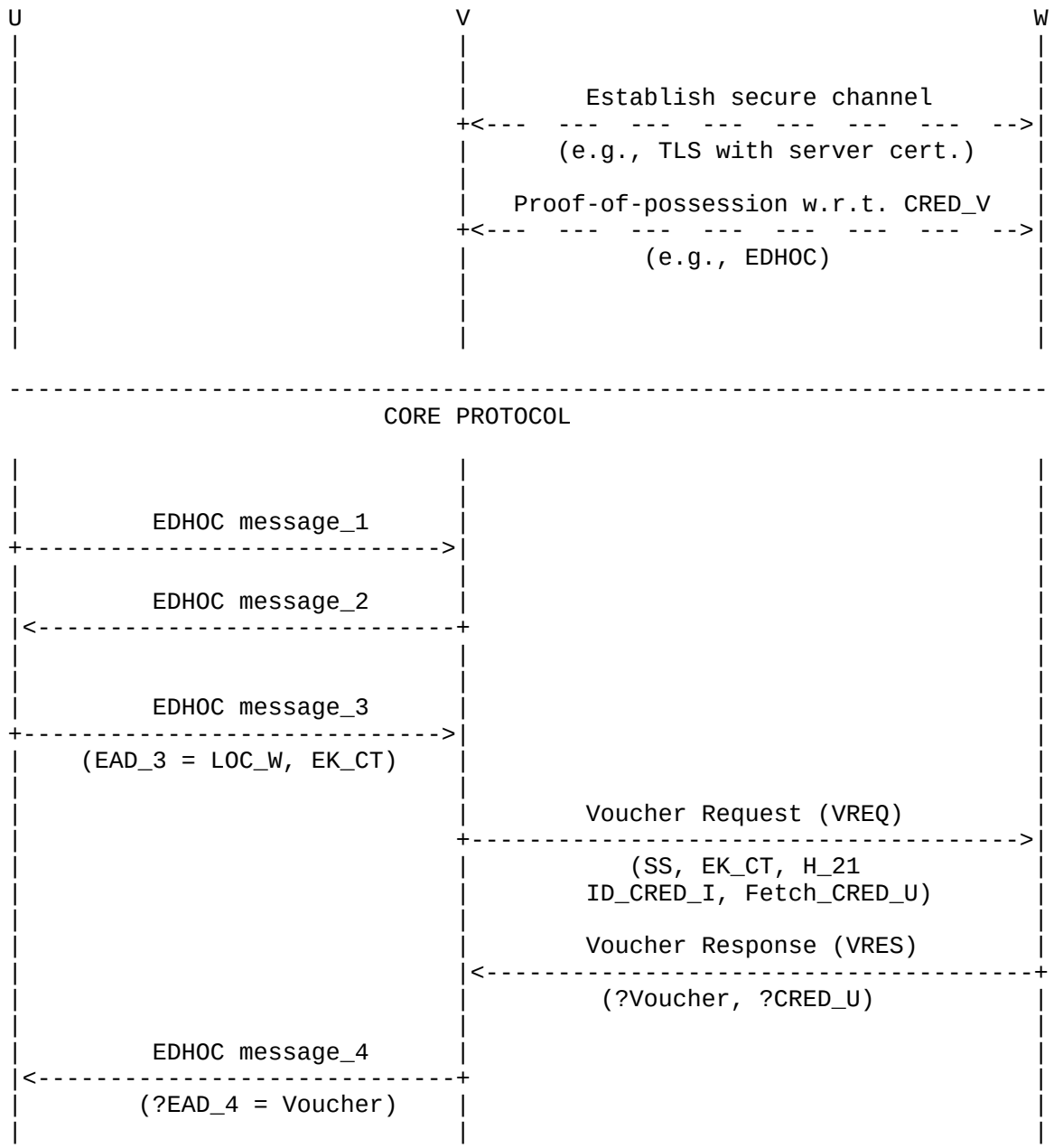

U
V
W
Establish secure channel
(e.g., TLS with server cert.)
Proof-of-possession w.r.t. CRED_V
(e.g., EDHOC)
CORE PROTOCOL
EDHOC message_1
EDHOC message_2
EDHOC message_3
(EAD_3 = LOC_W, EK_CT)
Voucher Request (VREQ)
(SS, EK_CT, H_21
ID_CRED_I, Fetch_CRED_U)
Voucher Response (VRES)
(?Voucher, ?CRED_U)
EDHOC message_4
(?EAD_4 = Voucher)

Figure 3: Overview of ELA: W-assisted authorization of U and V to each other: EDHOC between U and V, and Voucher Request/Response between V and W. Before the protocol, V and W are assumed to have established a secure channel and performed proof-of-possession of relevant keys. Credential lookup of CRED_U may involve W or other credential database.

4.2. Reuse of EDHOC

The ELA protocol illustrated in Figure 3 reuses several components of EDHOC:

* SUITES_I includes the cipher suite for EDHOC selected by U, and also defines the algorithms used between U and W (see Section 3.6 of [RFC9528]):
  - EDHOC AEAD algorithm: used to generate voucher
  - EDHOC hash algorithm: used for key derivation
  - EDHOC key exchange algorithm: used to calculate the shared secret between U and W
* EAD_3, EAD_4 are the External Authorization Data message fields of message_3 and message_4, respectively, see Section 3.8 of [RFC9528]. In case U acts as Responder (see Section 4.7), EAD_2 and EAD_3 are used in message_2 and message_3, respectively. This document specifies two new EAD items, with ead_label = TBD1 and TBD2, see Section 7.1.
* ID_CRED_I and ID_CRED_R are used to identify the authentication credentials CRED_U and CRED_V, respectively. As shown at the bottom of Figure 3, V may use W to obtain CRED_U. By default, CRED_V is transported in ID_CRED_R in message_2, see Section 4.4.2.1. In case U is Responder, CRED_V is transported in ID_CRED_I in message_3.

The protocol also reuses the EDHOC_Extract and EDHOC_Expand key derivation from EDHOC (see Section 4 of [RFC9528]).

The intermediate pseudo-random key PRK is derived using EDHOC_Extract():

* PRK = EDHOC_Extract(salt, IKM)
  - where salt = 0x (the zero-length byte string)
  - Computation of IKM depends on the EDHOC method in use.

- If the method is based on Diffie-Hellman, IKM is computed as an ECDH cofactor Diffie-Hellman shared secret from the public key of W, PK_W, and the private key corresponding to G_U (or v.v.), see Section 5.7.1.2 of [NIST-800-56A] and Section 4.3.

- If the method is based on a Key Encapsulation Mechanism (KEM), IKM is the shared secret resulting from encapsulating PK_W, see Section 2.2 of [NIST-800-227]. For example, the use of ML-KEM in COSE is currently being specified at [I-D.ietf-jose-pqc-kem].

The output keying material OKM is derived from PRK using EDHOC_Expand(), which is defined in terms of the EDHOC hash algorithm of the selected cipher suite SS, see Section 4.1.2 of [RFC9528]:

* OKM = EDHOC_Expand(PRK, info, length)

  where

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

Finally, since the ELA authorization flow happens in EDHOC message_3 and message_4, the ELA is also compatible with EDHOC application profiles, as defined in [I-D.ietf-lake-app-profiles] where advertisement of supported profiles happens in message_1 and message_2. This can be used, for example, to enable U or V to learn about each other’s capability for executing the ELA protocol.

4.3. Device <-> Enrollment Server (U <-> W)

The protocol between U and W is carried between U and V in message_3 and message_4 (Section 4.4), and between V and W in the Voucher Request/Response (Section 4.5). The data is protected between the endpoints using secret keys derived from a shared secret (see Section 4.2) as further detailed in this section.

4.3.1. Voucher Info

The external authorization data EAD_3 contains a critical EAD item with ead_label = -TBD1 and ead_value = Voucher_Info, which is a CBOR byte string:

```
Voucher_Info = bstr .cborseq Voucher_Info_Seq

Voucher_Info_Seq = [ ; used as a CBOR sequence, not array
    LOC_W: tstr,
    EK_CT: bstr,
]
```

where

* LOC_W is a text string used by V to locate W, e.g., a URI or a domain name.

* EK_CT is a field containing either an ephemeral Diffie-Hellman key or a KEM ciphertext, depending on the EDHOC method being used in the current session:

  - if the method is based on Diffie-Hellman, the device generates an ephemeral Diffie-Hellman key pair, where the public part G_U is placed in the EK_CT field. The private part of G_U will be used to decrypt and verify the voucher, see Section 4.3.2 and Section 4.2.

  - if the method is based on a PQC KEM, the device performs key encapsulation and places the resulting ciphertext CT in the EK_CT field. Here, the secret emitted by the encapsulation mechanism will be used to decrypt and verify the voucher, see Section 4.3.2 and Section 4.2.

4.3.2. Voucher

If W generates a Voucher, as determined by the application, the external authorization data EAD_4 contains an EAD item with ead_label = -TBD2 and ead_value = Voucher.

The voucher is an assertion to U that W has authorized V. It is encrypted using the EDHOC AEAD algorithm of the selected cipher suite SS specified in SUITE_I of EDHOC message_1. It consists of the 'ciphertext' field of a COSE_Encrypt0 object [RFC9052], which is a byte string, as defined below.

```
Voucher = bstr
```

Its corresponding plaintext value consists of an opaque field that can be used by W to convey information to U, such as a voucher scope. The authentication tag present in the ciphertext is also bound to the current EDHOC handshake, the identity of U, and the credential of V as described below.

* The encryption key K and nonce IV are derived as specified below.

* 'protected' is a byte string of size 0

* 'plaintext' and 'external_aad' as below:

```
plaintext = (
    ?OPAQUE_INFO: bstr
)

external_aad = (
    H_21:         bstr,
    ID_CRED_I:    bstr,
    CRED_V:       bstr,
)
```

where

* OPAQUE_INFO is an opaque field provided by the application. If present, it will contain application data that W may want to convey to U, e.g., a voucher scope. Note that OPAQUE_INFO is opaque when viewed as an information element in EDHOC. It is opaque to V, while the application in U and W can read its contents.

* H_21 is H(message_2, H(message_1)), used to bind the voucher to the current EDHOC session. It is computed using the EDHOC hash algorithm of the selected cipher suite SS specified in SUITE_I of EDHOC message_1. H_21 is sent to W as part of the voucher request, see Section 4.5.1.

* ID_CRED_I is the identifier of U transmitted via the voucher request.

* CRED_V is the credential used by V to authenticate to U and W, see Section 4.4.2.1 and Table 1.

The derivation of K = EDHOC_Expand(PRK, info, length) uses the following input to the info struct (see Section 4.2):

* info_label = 2

* context = h'' (the empty CBOR string)

* length is length of the key of the EDHOC AEAD algorithm in bytes

The derivation of IV = EDHOC_Expand(PRK, info, length) uses the following input to the info struct (see Section 4.2):

* info_label = 3

* context = h'' (the empty CBOR string)

* length is length of the nonce of the EDHOC AEAD algorithm in bytes

4.4. Device <-> Authenticator (U <-> V)

This section describes the processing in U and V, which includes the EDHOC protocol, see Figure 3. Normal EDHOC processing is omitted here.

4.4.1. Message 1

4.4.1.1. Processing in U

U composes EDHOC message_1 using authentication method, identifiers, etc. according to an agreed application profile, see Section 3.9 of [RFC9528]. The selected cipher suite, in this document denoted SS, applies also to the interaction with W as detailed in Section 4.2, in particular, with respect to the key agreement algorithm used between U and W. U sends EDHOC message_1 to V.

4.4.1.2. Processing in V

V processes EDHOC message_1 as specified in [RFC9528].

Note that as part of normal EDHOC processing, U and V negotiate a selected cipher suite SS, as specified in Section 6.3.1 of [RFC9528].

4.4.2. Message 2

4.4.2.1. Processing in V

V composes EDHOC message_2 as specified in Section 5.3.3 of [RFC9528].

The type of CRED_V may depend on the selected mechanism for the establishment of a secure channel between V and W, See Table 1.

In case the network between U and V is constrained, it is recommended that CRED_V be a CWT Claims Set (CCS) [RFC8392]. The CCS contains the public authentication key of V encoded as a COSE_Key in the 'cnf' claim, see Section 3.5.2 of [RFC9528]. ID_CRED_R contains the CWT Claims Set with 'kccs' as COSE header_map, see Section 10.6 of [RFC9528].

4.4.2.2. Processing in U

U receives EDHOC message_2 from V and processes it as specified in Section 5.3.3 of [RFC9528].

4.4.3. Message 3

4.4.3.1. Processing in U

U prepares EDHOC message_3 with EAD item (-TBD1, Voucher_Info) included in EAD_3, where Voucher_Info is specified in Section 4.3. The negative sign indicates that the EAD item is critical, see Section 3.8 of [RFC9528].

4.4.3.2. Processing in V

V receives EDHOC message_3 from U and processes it as specified in Section 5.4.3 of [RFC9528], with the additional step of processing the EAD item in EAD_3. Since the EAD item is critical, if V does not recognize it or it contains information that V cannot process, then V MUST abort the EDHOC session, see Section 3.8 of [RFC9528]. The ead_label = -TBD1 triggers the voucher request to W as described in Section 4.5. The exchange between V and W needs to be completed successfully for the EDHOC session to be continued.

As part of normal processing of EDHOC message_3, V must verify the credential of U. It is up to the application to decide whether to verify the credential of U before or after the voucher request to W, see pre and post -verification processing of EAD items in [I-D.ietf-lake-edhoc-impl-cons].

In case V has access to a credential database, it MAY query it with ID_CRED_I to obtain a corresponding CRED_U and verify the identity of U before making the Voucher request to W. If the verification of CRED_U fails, the EDHOC session is aborted. In this scenario, the EAD item in EAD_3 is processed as a post-verification item [I-D.ietf-lake-edhoc-impl-cons].

Alternatively, V MAY proceed and perform a voucher request and wait for a voucher response. If the response contains CRED_U, V then continues processing EDHOC message_3 where it verifies the credential of U. Even upon reception of a successful voucher response from W, if the verification of CRED_U fails, the EDHOC session is aborted. In this scenario, the EAD item in EAD_3 is processed as a pre-verification item [I-D.ietf-lake-edhoc-impl-cons].

4.4.4. Message 4

4.4.4.1. Processing in V

At this point, V has authenticated U, and received a valid voucher response from W as described in Section 4.5.

V prepares EDHOC message_4 where, if the voucher response contains a voucher, V includes the critical EAD item (-TBD2, Voucher) in EAD_4, i.e., ead_label = -TBD2 and ead_value = Voucher, as specified in Section 4.3.2.

4.4.4.2. Processing in U

U receives EDHOC message_4 from V and processes it as specified in Section 5.5.3 of [RFC9528], with the additional step of processing EAD_4.

If EAD_4 contains an EAD item with label = -TBD2, U MUST verify the Voucher using H_21, ID_CRED_I, CRED_V, and the keys derived as in Section 4.3.2. If the verification fails then U MUST abort the EDHOC session.

If U does not recognize the EAD item or the EAD item contains information that U cannot process, then U MUST abort the EDHOC session, see Section 3.8 of [RFC9528].

If OPAQUE_INFO is present, it is made available to the application.

4.5. Authenticator <-> Enrollment Server (V <-> W)

It is assumed that V and W have set up a secure connection, W has accessed the authentication credential CRED_V to be used in the EDHOC session between V and U, and that W has verified that V is in possession of the private key corresponding to CRED_V, see Section 3.2 and Section 3.3. V and W run the Voucher Request/ Response protocol over the secure connection.

4.5.1. Voucher Request

4.5.1.1. Processing in V

V sends the voucher request to W. The Voucher Request SHALL be a CBOR array as defined below:

```
Voucher_Request = [
    SS:             int,
    EK_CT:          bstr,
    H_21:           bstr,
    ID_CRED_I:      bstr,
    Fetch_CRED_U    bool,
]
```

where

* SS is the selected cipher suite used in the EDHOC session between U and V.

* EK_CT is either an ephemeral public key or a KEM ciphertext set by U, as defined in Section 4.3.

* H_21 corresponds to H(message_2, H(message_1)). It is computed as defined in Section 4.3.2.

* Fetch_CRED_U is a flag indicating whether W should try to load and return the credential CRED_U corresponding to ID_CRED_I.

4.5.1.2. Processing in W

W receives and parses the voucher request received over the secure connection with V. W extracts from Voucher_Request:

* SS - the selected cipher suite.

* EK_CT - either an ephemeral public key or a KEM ciphertext.

* H_21 - the hash of message_2 and message_1.

* ID_CRED_I - the identifier of U.

* Fetch_CRED_U - flag indicating whether V requests W to return CRED_U.

W verifies that it supports the cipher suite and parses the key or ciphertext in EK_CT.

W uses H_21 as a session identifier, and associates it to the device identifier ID_CRED_I. Note that EK_CT is unique, as the ephemeral key or the ciphertext MUST not be reused, therefore H_21 is expected to be unique.

If processing fails up until this point, the protocol SHALL be aborted with an error code signaling a generic issue with the request, see Section 5.4.1.

W uses ID_CRED_I to look up the associated authorization policies for U and enforces them. This is out of scope for the specification.

If ID_CRED_I is known by W, but authorization fails, the protocol SHALL be aborted with an error code signaling an access control issue, see Section 4.6 and Section 5.4.1.

If Fetch_CRED_U is true, W uses ID_CRED_I to retrieve the corresponding credential CRED_U. If retrieval succeeds, W MUST include CRED_U in the voucher response, see Section 4.5.2. If the retrieval fails, W sends the voucher response without CRED_U. If Fetch_CRED_U is false, W MUST NOT include CRED_U in the voucher response.

4.5.2. Voucher Response

4.5.2.1. Processing in W

In case a Voucher is needed (as determined by the application), W retrieves CRED_V associated with the secure connection with V, and constructs the Voucher for the device with identifier ID_CRED_I (see Section 4.3.2).

W generates the voucher response and sends it to V over the secure connection. The Voucher_Response SHALL be a CBOR array as defined below:

```
Voucher_Response = [
    ? Voucher:        bstr,
    ? CRED_U:       bstr,
]
```

where

* The Voucher is defined in Section 4.3.2, if present.

* CRED_U is the credential of U corresponding to ID_CRED_I passed in the voucher request. CRED_U is included only if Fetch_CRED_U was set in the voucher request and W successfully retrieved the credential.

W signals the successful processing of Voucher_Request via a status code in the REST interface, as defined in Section 5.4.1.

4.5.2.2. Processing in V

V receives the voucher response from W over the secure connection. If the voucher response is successfully received from W, then V responds to U with EDHOC message_4 as described in Section 4.4.4.1.

4.6. Error Handling

This section specifies a new EDHOC error code and how it is used in ELA.

4.6.1. EDHOC Error "Access denied"

This section specifies the new EDHOC error "Access denied", see Figure 4.

ERR_CODE	ERR_INFO Type	Description
TBD3	error_content	Access denied

Figure 4: EDHOC error code and error information for ‘Access denied’.

Error code TBD3 is used to indicate to the receiver that access control has been applied and the sender has aborted the EDHOC session. The ERR_INFO field contains error_content which is a CBOR Sequence consisting of an integer and an optional byte string.

```
error_content = (
  REJECT_TYPE : int,
  ? REJECT_INFO : bstr,
)
```

The purpose of REJECT_INFO is for the sender to provide verifiable and actionable information to the receiver about the error, so that an automated action may be taken to enable access.

REJECT_TYPE	REJECT_INFO	Description
0	-	No REJECT_INFO
1	bstr	REJECT_INFO from trusted third party

Figure 5: REJECT_TYPE and REJECT_INFO for ‘Access denied’.

4.6.2. Error handling in W, V, and U

ELA uses the EDHOC Error "Access denied" in the following way:

* W generates error_content and transfers it to V via the secure connection. If REJECT_TYPE is 1, then REJECT_INFO is encrypted from W to U using the EDHOC AEAD algorithm. W signals the error via an appropriate status code in the REST interface, as defined in Section 5.4.1.

* V receives error_content, prepares an EDHOC "Access denied" error, and sends it to U.

* U receives the error message and extracts the error_content. If REJECT_TYPE is 1, then U decrypts REJECT_INFO, based on which it may retry to gain access.

The encryption of REJECT_INFO follows a procedure analogous to the one defined in Section 4.3.2, with the following differences:

```
plaintext = (
    OPAQUE_INFO:   bstr,
 )

external_aad = (
    H_21:          bstr,
 )
```

where

* OPAQUE_INFO is an opaque field that contains actionable information about the error. It may contain, for example, a list of suggested Vs through which U should join instead.

* H_21 is the hash of EDHOC message_2 and message_1, obtained from the associated voucher request, see Section 4.5.1.

4.7. Reverse flow with U as Responder

This section presents a protocol variant in which U is the EDHOC Responder. This may allow optimizations in certain constrained network technologies. For example, one use case is having V broadcast message_1, to which U responds with a message_2 whose EAD_2 field contains Voucher_Info.

Note that this is different from the EDHOC reverse message flow defined in Appendix A.2.2 of [RFC9528], since we make no assumption about whether U or V is a CoAP server.

4.7.1. U is the Initiator

For clarity, we first present the regular flow with U as Initiator, as described in Section 4.1 and Section 4.4. Note that Voucher_Info and Voucher are carried in EDHOC message_3 and message_4, respectively.

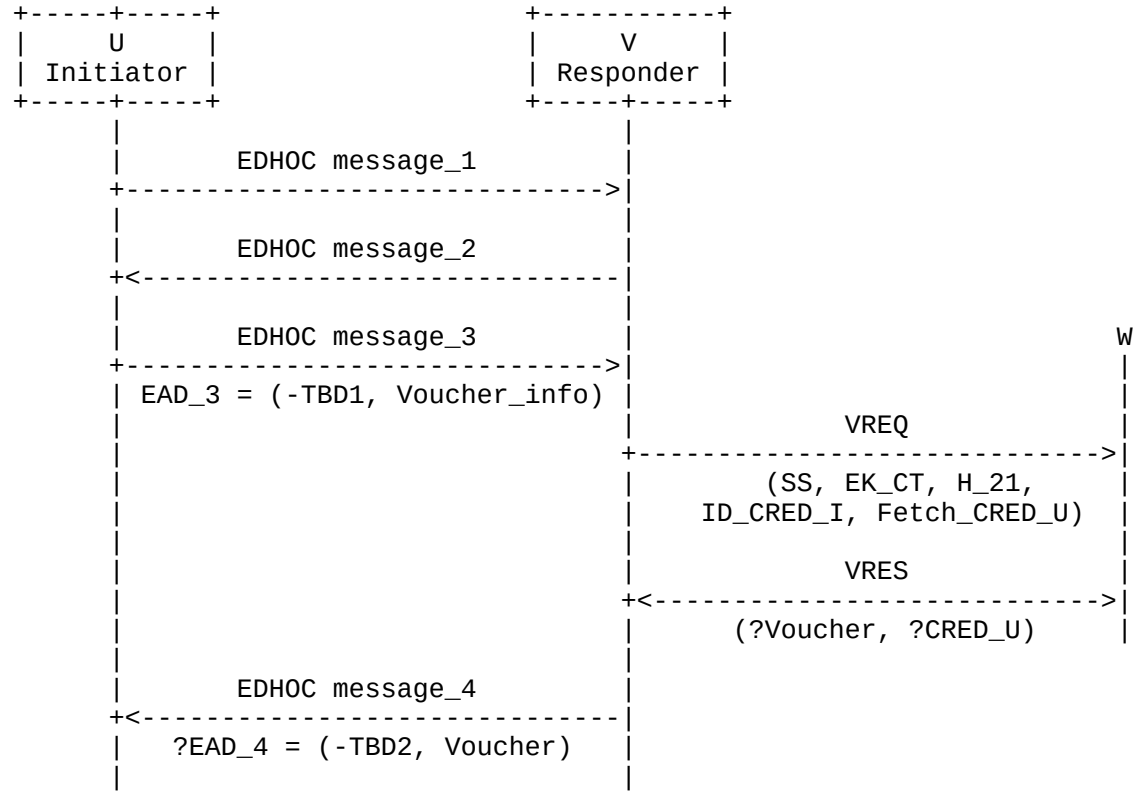


Figure 6: In ELA regular flow, U is the EDHOC Initiator.

In the ELA regular flow, once message_4 processing is finalized (including processing of EAD_4), both U and V are authorized to interact.

4.7.2. U is the Responder

ELA also works with U as the EDHOC Responder, a setup we refer to as the "ELA reverse flow", as shown in Figure 7.

We present this variant as a set of changes to the regular protocol flow. That is, here we only describe the differences in processing, when compared to the ELA regular flow.

Here is a summary of the changes needed in the ELA reverse flow:

* Voucher_Info and Voucher are transported in EDHOC message_2 and message_3, respectively (instead of message_3 and message_4).

* The EAD_2 and EAD_3 fields carry critical EAD items identified with labels -TBD1 and -TBD2, respectively.

* The VREQ / VRES protocol takes place between message_2 and message_3.

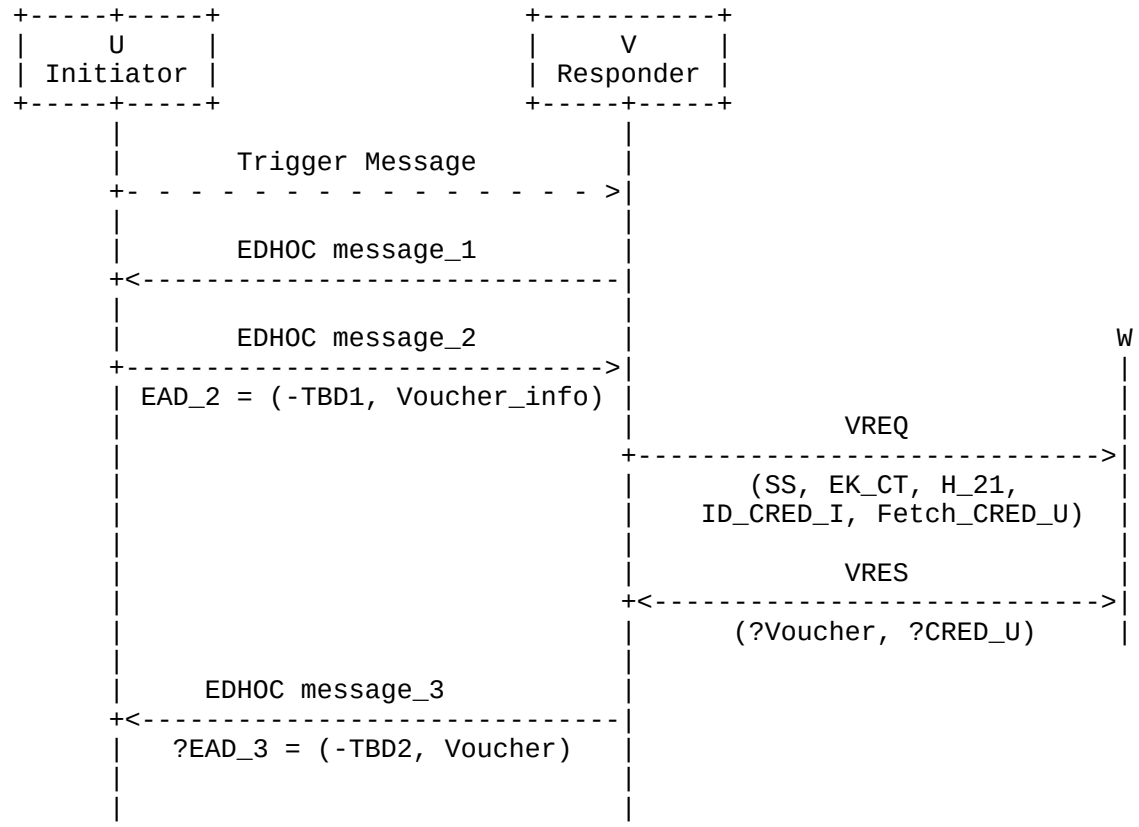


Figure 7: ELA when U is the EDHOC Responder.

The following sections detail how the processing of ELA reverse flow changes in each of the three security sessions (U-W, U-V, V-W), when compared to the baseline ELA regular flow described in sections Section 4.1 to Section 4.6.

4.7.2.1. Reverse U <-> W

The protocol between U and W is carried between U and V in message_2 and message_3, and between V and W in the Voucher Request/Response (Section 4.5).

Voucher Info:

* Voucher_Info is carried in EAD_2.

* It uses a critical EAD item with ead_label = -TBD1 and ead_value = Voucher_Info.

Voucher:

* The Voucher is carried in EAD_3.

* It uses a critical EAD item with ead_label = -TBD2 and ead_value = Voucher.

4.7.2.2. Reverse U <-> V

Message 1:

* V composes message_1 and sends it to U.

* U processes message_1 and extracts SS.

Message 2:

* U composes message_2 with a critical EAD item (-TBD1, Voucher_Info) included in EAD_2.

* U sends message_2 to V.

* V processes message_2 and the EAD item in EAD_2, extracting the Voucher_Info struct.

* V sends the voucher request to W.

Here, V can choose to validate CRED_U before or after making the voucher request, depending on application requirements and availability of CRED_U, as detailed in Section 4.4.

Message 3:

* V receives the voucher response from W.

* V sends message_3 with critical EAD item (-TBD2, Voucher) included in EAD_3.

* U processes message_3 and the EAD item in EAD_3.

The ELA reverse flow does not require sending a final EDHOC message_4.

4.7.2.3. Reverse V <-> W

Processing in V:

* The Voucher_Request fields are prepared as defined in Section 4.5.1, with the following changes:

  - EK_CT is as extracted from the EAD_2 field of message_2.

Processing in W happens as specified in Section 4.5.1.

4.7.3. Interoperability considerations

A Device (U) MUST implement one of the ELA flows, and it MAY choose to implement both.

V MUST support the regular flow and MAY support the reverse flow.

From the point of view of W, there is no difference whether U and V run as EDHOC Initiator or Responder.

4.7.4. Security implications

In the reverse flow, Voucher_Info is confidentiality and integrity protected, while Voucher is also authenticated. These properties are inherited from EDHOC message_2 and message_3. In contrast, the ELA regular flow provides confidentiality, integrity protection, and authentication to both Voucher_Info and Voucher, as inherited from EDHOC message_3 and message_4.

5. REST Interface at W

The interaction between V and W is enabled through a RESTful interface exposed by W. This RESTful interface MAY be implemented using either HTTP or CoAP. V SHOULD access the resources exposed by W through the protocol indicated by the scheme in the LOC_W URI.

5.1. Scheme "https"

In case the scheme indicates "https", V MUST perform a TLS handshake with W and access the resources defined in Section 5.4 using HTTP. If the authentication credential CRED_V can be used in a TLS handshake, e.g., an X.509 certificate of a signature public key, then V SHOULD use it to authenticate to W as a client. If the authentication credential CRED_V cannot be used in a TLS handshake, e.g., if the public key is a static key, then V SHOULD first perform a TLS handshake with W using available compatible keys. V MUST then perform an EDHOC session over the TLS connection proving to W the possession of the private key corresponding to CRED_V. Performing the EDHOC session is only necessary if V did not authenticate with CRED_V in the TLS handshake with W.

The relationship between V and W is long-lived. HTTP/1.1 and higher support persistent connections, and SHOULD be used in order to reduce overhead if a flood of new devices need to be onboarded. Support for TLS session resumptions tickets [RFC8446], Section 2.2 is appropriate for longer term associations. While a policy for renewal of the TLS connection should be applied, it is out of scope of this document.

5.2. Scheme "coaps"

In case the scheme indicates "coaps", V SHOULD perform a DTLS handshake with W and access the resources defined in Section 5.4 using CoAP. The normative requirements in Section 5.1 on performing the DTLS handshake and EDHOC session remain the same, except that TLS is replaced with DTLS. As in Section 5.1, it is RECOMMENDED to allow reuse of the DTLS session.

5.3. Scheme "coap"

In case the scheme indicates "coap", V SHOULD perform an EDHOC session with W, as specified in Appendix A of [RFC9528] and access the resources defined in Section 5.4 using OSCORE [RFC8613] and CoAP. The authentication credential in this EDHOC session MUST be CRED_V. As in Section 5.1, it is RECOMMENDED to allow reuse of the EDHOC session.

5.4. URIs

The URIs defined below are valid for both HTTP and CoAP. W MUST support the use of the path-prefix "/.well-known/", as defined in [RFC8615], and the registered name "lake-authz". A valid URI in case of HTTP thus begins with

* "https://www.example.com/.well-known/lake-authz"

In case of CoAP with DTLS:

* "coaps://example.com/.well-known/lake-authz"

In case of EDHOC and OSCORE:

* "coap://example.com/.well-known/lake-authz"

Each operation specified in the following is indicated by a path-suffix.

5.4.1. Voucher Request (/voucherrequest)

To request a voucher, V MUST issue a request such that:

* Method is POST

* Payload is the serialization of the Voucher Request object, as specified in Section 4.5.1.

* Content-Format (Content-Type) is set to "application/lake-authz-voucherrequest+cbor"

In case of successful processing at W, W MUST issue a response such that:

* Status code is 200 OK if using HTTP, or 2.04 Changed if using CoAP

* Payload is the serialized Voucher Response object, as specified in Section 4.5.2

* Content-Format (Content-Type) is set to "application/lake-authz-voucherresponse+cbor"

In case of error, two cases should be considered:

* U cannot be identified: this happens either if W fails to process the Voucher Request, or if it succeeds but ID_CRED_I is considered unknown to W. In this case, W MUST reply with 400 Bad Request if using HTTP, or 4.00 if using CoAP.

* U is identified but unauthorized: this happens if W is able to process the Voucher Request, and W recognizes ID_CRED_I as a known device, but the access policies forbid enrollment. For example, the policy could enforce enrollment within a delimited time window, via a specific V, etc. In this case, W MUST reply with a 403 Forbidden code if using HTTP, or 4.03 if using CoAP; the payload is the serialized error_content object, with Content-

Format (Content-Type) set to "application/lake-authz-vouchererror+cbor". The payload MAY be used by V to prepare an EDHOC error "Access Denied", see Section 4.6.

5.4.2. Certificate Request (/certrequest)

V requests the public key certificate of U from W through the "/certrequest" path-suffix. To request U's authentication credential, V MUST issue a request such that:

* Method is POST
* Payload is the serialization of the ID_CRED_I object, as received in EDHOC message_3.
* Content-Format (Content-Type) is set to "application/lake-authz-certrequest+cbor"

In case of a successful lookup of the authentication credential at W, W MUST issue a response such that:

* Status code is 200 OK if using HTTP, or 2.04 Changed if using CoAP
* Payload is the serialized CRED_U
* Content-Format (Content-Type) is set to "application/lake-authz-certresponse+cbor"

6. Security Considerations

This specification builds on and reuses many of the security constructions of EDHOC, e.g., shared secret calculation and key derivation. The security considerations of EDHOC [RFC9528] apply with modifications discussed here. These considerations apply to the ELA regular flow. For considerations about the ELA reverse flow, see Section 4.7.

The Voucher_Info and Voucher structs are sent over authenticated channels that are confidentiality and integrity protected between U and V, i.e., in EDHOC fields EAD_3 and EAD_4. While ELA reuses several components of EDHOC, it does not reuse keys from EDHOC (such as the ephemeral key G_X) to protect fields Voucher_Info and Voucher.

ELA is compatible with the currently standardized Diffie-Hellman shared secret derivation of EDHOC. Considering cryptographic recommendations by government agencies and the industry, ELA is also compatible with post-quantum cryptography primitives for deriving a shared secret, namely via the EK_CT field which can contain a KEM

ciphertext according to the selected cipher suite. Post-quantum cipher suites that could be used in EDHOC are currently under standardization in COSE [I-D.ietf-jose-pqc-kem].

EDHOC provides identity protection of the Initiator, here the device. In ELA, the device U will share its identity with an authenticated V, albeit before knowing (via the Voucher received from W) whether U is authorized to interact with W.

W may be used for lookup of CRED_U from ID_CRED_I, or this credential lookup function may be separate from the authorization function of W, see Section 4.4. The trust model used here is that U reveals its identity to any V, before knowing if the ELA authorization flow will be successfully completed. In onboarding use cases, U MAY choose to use an onboarding-only identity, whereas future operational communications can use a different identity under the already established secure channel.

In the ELA regular flow, EDHOC message_4 is mandatory since it carries the Voucher. Implementations MUST NOT omit message_4 when ELA is in use.

7. IANA Considerations

7.1. EDHOC External Authorization Data Registry

IANA has registered the following entries in the "EDHOC External Authorization Data" registry under the group name "Ephemeral Diffie-Hellman Over COSE (EDHOC)".

Label	Value Type	Description
TBD1	bstr	Voucher_Info structure, prepared by the Device (U).
TBD2	bstr	Voucher structure, prepared by the Enrollment Server (W).

Table 2: Addition to the EDHOC EAD registry

The ead_label = TBD1 corresponds to the ead_value = Voucher_Info, which can be carried in either EAD_2 or EAD_3, depending on whether U acts as EDHOC Initiator or Responder, see Section 4.7.

The ead_label = TBD2 corresponds to ead_value = Voucher, and can be carried in either EAD_3 or EAD_4, see Section 4.7.

Note for IANA reviewers: the preferred value range is 0-23 (Standards Action with Expert Review).

7.2. The Well-Known URI Registry

IANA has registered the following entry in "The Well-Known URI Registry", using the template from [RFC8615]:

* URI suffix: lake-authz

* Change controller: IETF

* Specification document: [[this document]]

* Status: permanent

* Related information: None

7.3. Well-Known Name Under ".arpa" Name Space

This document allocates a well-known name under the .arpa name space according to the rules given in [RFC3172] and [RFC6761]. The name "lake-authz.arpa" is requested. No subdomains are expected, and addition of any such subdomains requires the publication of an IETF Standards Track RFC. No A, AAAA, or PTR record is requested.

7.3.1. Domain Name Reservation Considerations

As required by [RFC6761], the following considerations apply to the reservation of "lake-authz.arpa":

1. Users: Are human users expected to recognize these names as special and use them differently? In what way?

No. This name is not intended for direct use or recognition by human users.

1. Application Software: Are writers of application software expected to make their software recognize these names as special and treat them differently? In what way? (For example, if a human user enters such a name, should the application software reject it with an error message?)

Yes. Applications that implement ELA and use CoAP may include "lake-authz.arpa" in the URI-Host option when the Device (U) does not yet know the address or identity of the Authenticator (V), such as during zero-touch enrollment.

1. Name Resolution APIs and Libraries: Are writers of name resolution APIs and libraries expected to make their software recognize these names as special and treat them differently? If so, how?

No.

1. Caching DNS Servers: Are developers of caching domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

No.

1. Authoritative DNS Servers: Are developers of authoritative domain name servers expected to make their implementations recognize these names as special and treat them differently? If so, how?

No.

1. DNS Server Operators: Does this reserved Special-Use Domain Name have any potential impact on DNS server operators? If they try to configure their authoritative DNS server as authoritative for this reserved name, will compliant name server software reject it as invalid? Do DNS server operators need to know about that and understand why? Even if the name server software doesn’t prevent them from using this reserved name, are there other ways that it may not work as expected, of which the DNS server operator should be aware?

No.

1. DNS Registries/Registrars: How should DNS Registries/Registrars treat requests to register this reserved domain name? Should such requests be denied? Should such requests be allowed, but only to a specially designated entity? (For example, the name "www.example.org" is reserved for documentation examples and is not available for registration; however, the name is in fact registered; and there is even a website at that name, which states circularly that the name is reserved for use in documentation and cannot be registered!)

Any requests to register this domain name should be denied.

7.4. Media Types Registry

IANA has added the media types "application/lake-authz-voucherrequest+cbor" to the "Media Types" registry.

7.4.1. application/lake-authz-voucherrequest+cbor Media Type Registration

* Type name: application
* Subtype name: lake-authz-voucherrequest+cbor
* Required parameters: N/A
* Optional parameters: N/A
* Encoding considerations: binary (CBOR)
* Security considerations: See Section 6 of this document.
* Interoperability considerations: N/A
* Published specification: [[this document]] (this document)
* Application that use this media type: To be identified
* Fragment identifier considerations: N/A
* Additional information:
  - Magic number(s): N/A
  - File extension(s): N/A
  - Macintosh file type code(s): N/A
* Person & email address to contact for further information: IETF LAKE Working Group (lake@ietf.org)
* Intended usage: COMMON
* Restrictions on usage: N/A
* Author: LAKE WG
* Change Controller: IETF

7.5. CoAP Content-Formats Registry

IANA has added the following Content-Format number in the "CoAP Content-Formats" registry under the registry group "Constrained RESTful Environments (CoRE) Parameters".

Content Type	Content Encoding	ID	Reference
application/lake-authz-voucherrequest+cbor	-	TBD3	[[this document]]

Table 3: Addition to the CoAP Content-Formats registry

Note for IANA reviewers: the preferred value range is 0-255 (Expert Review).

8. References

8.1. Normative References

[NIST-800-227]
Alagic, G., Barker, E., Chen, L., Moody, D., Robinson, A., Silberg, H., and N. Waller, "Recommendations for Key-Encapsulation Mechanisms - NIST Special Publication 800-227", September 2025, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-227.pdf>.

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

[RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-editor.org/info/rfc8366>.

[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>.

[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>.

[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>.

[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>.

8.2. Informative References

[I-D.ietf-jose-pqc-kem] Reddy.K, T., Banerjee, A., and H. Tschofenig, "Post-Quantum Key Encapsulation Mechanisms (PQ KEMs) for JOSE and COSE", Work in Progress, Internet-Draft, draft-ietf-jose-pqc-kem-05, 8 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-jose-pqc-kem-05>.

[I-D.ietf-lake-app-profiles] Tiloca, M. and R. Hglund, "Coordinating the Use of Application Profiles for Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-app-profiles-04, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-app-profiles-04>.

[I-D.ietf-lake-edhoc-impl-cons] Tiloca, M., "Implementation Considerations for Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-impl-cons-06, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-impl-cons-06>.

[I-D.ietf-lake-reqs] Vuini, M., Selander, G., Mattsson, J. P., and D. Garcia-Carillo, "Requirements for a Lightweight AKE for OSCORE", Work in Progress, Internet-Draft, draft-ietf-lake-reqs-04, 8 June 2020, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-reqs-04>.

[IEEE802.15.4]
IEEE standard for Information Technology, "IEEE Std 802.15.4 Standard for Low-Rate Wireless Networks", n.d..

[IEEE802.1X]
IEEE standard for Information Technology, "IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control", February 2010.

[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>.

[RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <https://www.rfc-editor.org/info/rfc3172>.

[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>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <https://www.rfc-editor.org/info/rfc6761>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC7593] Wierenga, K., Winter, S., and T. Wolniewicz, "The eduroam Architecture for Network Roaming", RFC 7593, DOI 10.17487/RFC7593, September 2015, <https://www.rfc-editor.org/info/rfc7593>.

[RFC8137] Kivinen, T. and P. Kinney, "IEEE 802.15.4 Information Element for the IETF", RFC 8137, DOI 10.17487/RFC8137, May 2017, <https://www.rfc-editor.org/info/rfc8137>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

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

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/info/rfc8995>.

[RFC9031] Vuini, M., Ed., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", RFC 9031, DOI 10.17487/RFC9031, May 2021, <https://www.rfc-editor.org/info/rfc9031>.

Appendix A. Optimization Strategies

When ELA is used for zero-touch enrollment of IoT devices, U may have little to no knowledge about V's available in its vicinity. This may lead to situations where U retries several times at different V's until it finds one that works. This section presents two optimization strategies and different approaches to implement it. These strategies were developed to address scenarios where V's are radio gateways to which U wants to enroll, but may also be applicable to other use cases.

A.1. Using EDHOC Application Profiles

This section provides an example of how support for ELA can be advertised using EDHOC application profiles.

Application Profiles for EDHOC [I-D.ietf-lake-app-profiles] defines mechanisms to describe and distribute parameters regarding the supported capabilities of EDHOC peers. Specifically, Section 5 of [I-D.ietf-lake-app-profiles] describes how EDHOC peers can advertise supported profiles by using fields EAD_1 and EAD_2 in EDHOC message_1 and message_2, respectively.

Since the ELA regular flow uses message_3 and message_4, peers can advertise support for ELA, and refrain from attempting the protocol in case one does not support it. In the case of the reverse flow, V can advertise a profile in message_1, while the protocol happens in message_2 and message_3.

For example, to advertise support for ELA in the regular flow, U prepares EAD_1 containing an item where ead_label = TBD_EAD_LABEL and ead_value encodes a profile_id_with_eads, containing the following value:

```
<<
   true,                                  / reply_flag /
   [e'APP-PROF-MINIMAL-CS-2', TBD1, TBD2]  / profile_id_with_eads /
>>
```

Where:

* true indicates that U expects an application profile response from V in EAD_2, see Section 5.1 of [I-D.ietf-lake-app-profiles]

* APP-PROF-MINIMAL-CS-2 corresponds to a well-known EDHOC application profile (Method 3; Cipher Suite 2; CCS; kid), see Section 8 of [I-D.ietf-lake-app-profiles]

* TBD1 and TBD2 correspond to the EAD labels registered in this document, see Section 7

In the reply, V prepares EAD_2 containing an item where ead_label = TBD_EAD_LABEL and ead_value encodes a profile_id_with_eads, containing the following value:

```
<<
   [e'APP-PROF-EXTENSIVE', TBD1, TBD2]  / profile_id_with_eads /
>>
```

Where APP-PROF-EXTENSIVE corresponds to a well-known EDHOC application profile, see Section 8 of [I-D.ietf-lake-app-profiles].

A.2. V advertises support for ELA

In this strategy, V shares some information (V_INFO) with a potential U, that can help it decide whether to try to enroll with that V.

The exact contents of the V_INFO structure, as well as the mechanism used to transport it, will depend on the underlying communication technology and also on application needs. For example, V_INFO may state that:

* V implements ELA -- similarly to how EAPOL [IEEE802.1X] frames state support for IEEE 802.1X.

* V is part of a certain domain -- similarly to how Eduroam [RFC7593] is used in the SSID field of IEEE 802.11 packets

V_INFO can be sent over a network beacon (see Appendix A.2.1), which may require technology specific profiling, e.g., the IEEE 802.15.4 enhanced beacon may be extended according to [RFC8137]. Alternatively, V_INFO can be sent as part of an EAD field, as shown in Appendix A.2.2.

As a guideline for implementers, we define the following field that can be included in a V_INFO structure:

DOMAIN_ID: bstr

The DOMAIN_ID field identifies the domain to which V belongs to, for example an URL or UUID.

Below are three examples of how the advertisement strategy may be applied according to different application needs. The examples include sending V_INFO in network beacons, as part of EAD_1 in reverse message flow, or as part of a periodic CoAP multicast packet. The advantages, costs, and security impacts of each approach are also discussed.

A.2.1. V_INFO in network beacons

This approach allows carrying V_INFO in beacons sent over the network layer, as shown in Figure 8. It requires that the network layer offers a mechanism to configure its beacon packets. Depending on the network type, a solicitation packet may also be needed, as is the case of non-beaconed IEEE 802.15.4 and BLE with GATT.

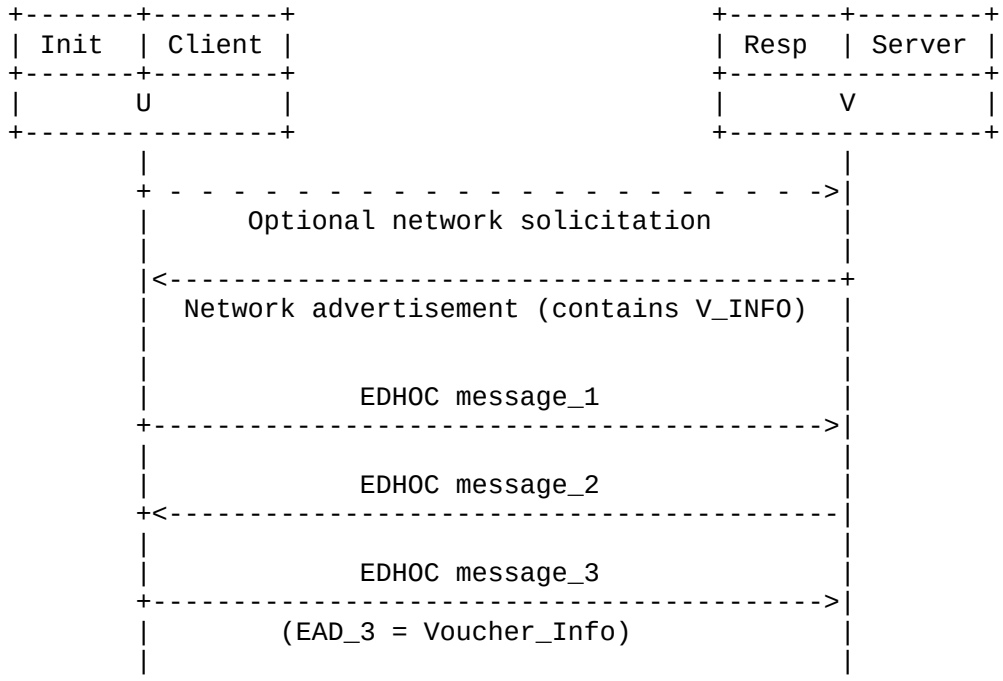


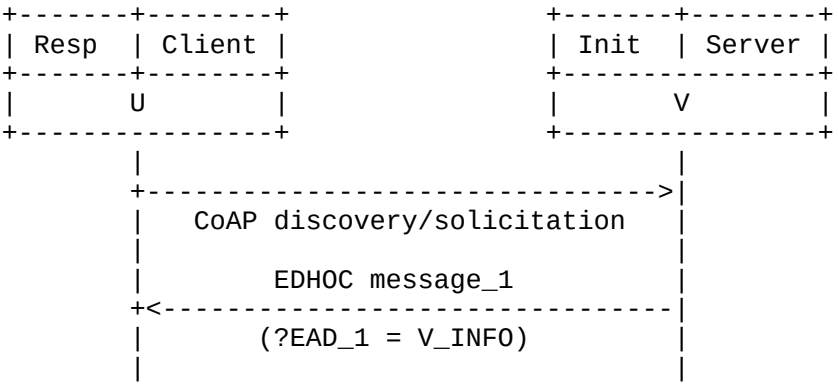
Figure 8: Advertising ELA using V_INFO in network-layer beacons.

This strategy can be used, for example, in IEEE 802.15.4, where an Enhanced Beacon [IEEE802.15.4] can be used to transmit V_INFO. Specifically, a new information element for carrying V_INFO can be defined according to [RFC8137].

This approach has the advantage of requiring minimal changes to the regular protocol as presented in Section 4.1, i.e., it does not require using the ELA reverse flow. It requires, however, some profiling of the lower layer beacons.

A.2.2. V_INFO in EAD_1

The ELA reverse flow (see Section 4.7) allows implementing advertising where U first sends a trigger packet, in the format of a CoAP request that is broadcasted to the newtork. When a suitable V receives the solicitation, if it implements ELA, it should respond with an EDHOC message_1 whose EAD_1 has label -TBD4 and value V_INFO (see Appendix A.2).



(... reverse flow continues normally ...)

Figure 9: Advertising ELA using V_INFO in EAD_1, employing the EDHOC reverse flow with U as responder.

Note that V will only reply if it supports ELA. V_INFO can be structured to contain only an optional domain identifier:

```
V_INFO = (
  ?DOMAIN_ID: bstr,
)
```

This approach enables a simple filtering mechanism, where only V's that support ELA will reply. In addition, it may not require layer-two profiling (in case the network allows transporting data before authorization).

A.2.3. V_INFO in a CoAP Multicast Packet

In this approach, V periodically multicasts a CoAP packet containing V_INFO, see Figure 10. Upon receiving one or more CoAP messages and processing V_INFO, U can decide whether or not to initiate the ELA protocol with a given V. Next, the application can either keep U acting as a server, and thus employ the EDHOC reverse flow, or implement a CoAP client and use the forward flow.

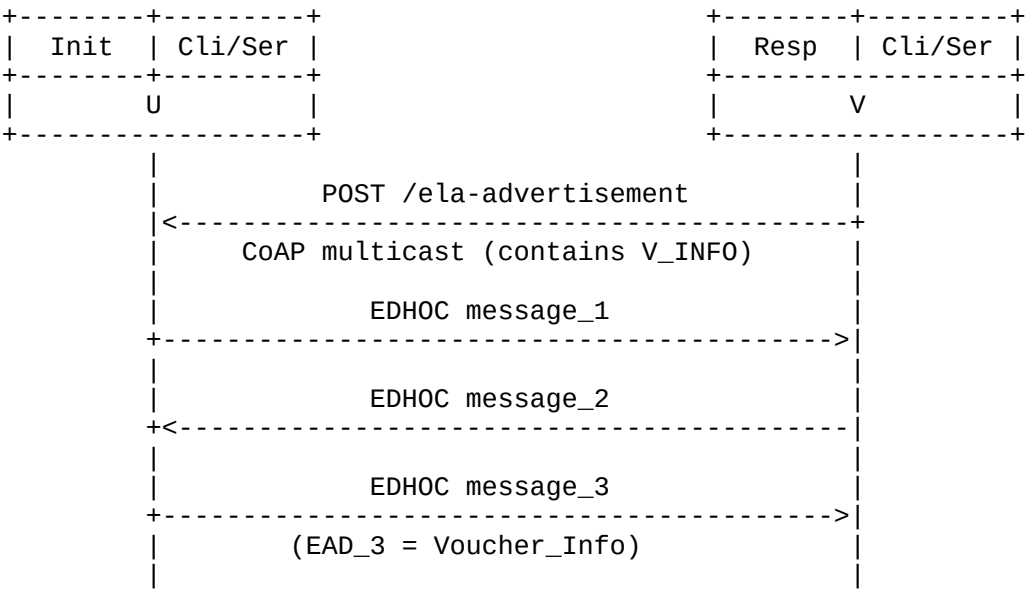


Figure 10: Advertising ELA using the network layer.

The V_INFO structure is sent as part of the CoAP payload. It is encoded as a CBOR sequence:

```
V_INFO = (
  ?DOMAIN_ID: bstr,
)
```

In this approach, the periodic multicast may have resource usage impacts in the network.

Appendix B. Examples of protocol execution

This section presents high level examples of the protocol execution.

Note: the examples below include samples of access policies used by W. These are provided for the sake of completeness only, since the authorization mechanism used by W is out of scope in this document.

B.1. Minimal

This is a simple example that demonstrates a successful execution of ELA.

Premises:

* device u1 has identity ID_CRED_I = key id = 14

* the access policy in W specifies, via a list of identities, that device u1 can enroll via any domain authenticator, i.e., the list contains ID_CRED_I = 14. In this case, the policy only specifies a restriction in terms of U, effectively allowing enrollment via any V.

Execution:

1. device u1 discovers a gateway (v1) and tries to enroll

2. gateway v1 identifies the zero-touch join attempt by checking that the label of the EAD item in EAD_3 is -TBD1, and prepares a Voucher Request using the information contained in the value of the EAD item

3. upon receiving the request, W obtains ID_CRED_I = 14, authorizes the access, and replies with Voucher Response

B.2. Wrong gateway

In this example, a device u1 tries to enroll a domain via gateway v1, but W denies the request because the pairing (u1, v1) is not configured in its access policies.

This example also illustrates how the REJECT_INFO field of the EDHOC error Access Denied could be used, in this case to suggest that the device should select another gateway for the join procedure.

Premises:

* devices and gateways communicate via Bluetooth Low Energy (BLE), therefore their network identifiers are MAC addresses (EUI-48)

* device u1 has ID_CRED_I = key id = 14

* there are 3 gateways in the radio range of u1:

  - v1 with MAC address = A2-A1-88-EE-97-75

  - v2 with MAC address = 28-0F-70-84-51-E4

  - v3 with MAC address = 39-63-C9-D0-5C-62

* the access policy in W specifies, via a mapping of shape (ID_CRED_I; MAC1, MAC2, ...) that device u1 can only join via gateway v3, i.e., the mapping is: (14; 39-63-C9-D0-5C-62)

* W is able to map the PoP key of the gateways to their respective MAC addresses

Execution:

1. device u1 tries to join via gateway v1, which forwards the request to W

2. W determines that MAC address A2-A1-88-EE-97-75 is not in the access policy mapping, and replies with an error. The error_content has REJECT_TYPE = 1, and the plaintext OPAQUE_INFO (used to compute the encrypted REJECT_INFO) specifies a list of suggested gateways = [h'3963C9D05C62']. The single element in the list is the 6-byte MAC address of v3, serialized as a bstr.

3. gateway v1 assembles an EDHOC error "Access Denied" with error_content, and sends it to u1

4. device u1 processes the error, decrypts REJECT_INFO, and retries the protocol via gateway v3

Acknowledgments

The authors sincerely thank Aurelio Schellenbaum for his contribution in the initial phase of this work, and Marco Tiloca for extensively reviewing the document. We also thank Christian Amsss for his active participation in discussions that led to improvements in the document.

Work on this document has in part been supported by the Horizon Europe Framework Programme project OpenSwarm (grant agreement No. 101093046).

Authors' Addresses

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

John Preu Mattsson
Ericsson AB
Sweden

Email: john.mattsson@ericsson.com

Malia Vuini
INRIA
France
Email: malisa.vucinic@inria.fr

Geovane Fedrecheski
INRIA
France
Email: geovane.fedrecheski@inria.fr

Michael Richardson
Sandelman Software Works
Canada
Email: mcr+ietf@sandelman.ca