Lightweight Authenticated Key Exchange
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

Y. Song
Inria
G. Selander
Ericsson AB

2 March 2026

Remote attestation over EDHOC
draft-ietf-lake-ra-04

Abstract

This document specifies how to perform remote attestation as part of the lightweight authenticated Diffie-Hellman key exchange protocol EDHOC (Ephemeral Diffie-Hellman Over COSE), based on the Remote ATtestation procedureS (RATS) architecture.

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/ra/draft-ietf-lake-ra.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-lake-ra/.

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/ra.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Remote attestation is a security process which verifies and confirms the integrity and trustworthiness of a remote target (e.g., device, system, group of devices) in the network. This process helps establish a level of trust in the remote system before allowing the device to e.g. join the network or get access to some sensitive information and resources. The use cases that require remote attestation include secure boot and firmware management, cloud computing, network access control, etc.

The IETF working group Remote ATtestation procedureS (RATS) has defined an architecture [RFC9334] for remote attestation. The three main roles in the RATS architecture are the Attester, the Verifier and the Relying Party. The Attester generates evidence (a set of claims) concerning its identity and integrity, which must be appraised by the Verifier for its validity. Then, the Verifier produces the attestation result, which is consequently used by the Relying Party for the purposes of reliably applying application-specific actions.

One type of interaction model defined in the RATS architecture is called the background-check model. It resembles the procedure of how employers perform background checks to determine the prospective employee's trustworthiness, by contacting the respective organization

that issues a report. In this case, the employer acts as the Relying Party, the employee acts as the Attester and the organization acts as the Verifier. The Attester conveys evidence directly to the Relying Party and the Relying Party forwards the evidence to the Verifier for appraisal. Once the attestation result is computed by the Verifier, it is sent back to the Relying Party to decide what action to take based on the attestation result. Another model is called passport model, where the Attester communicates directly with the Verifier. The Attester presents the evidence to the Verifier and gets an attestation result from the Verifier. Then the Attester conveys the attestation result to the Relying Party. This specification employs both the RATS background-check model and the passport model.

This document specifies the protocol between the Attester and the Relying Party. The details of the protocol between the Relying Party and the Verifier in the background-check model, and the protocol between the Attester and the Verifier in the passport model are out of the scope. This communication may be secured through protocols such as EDHOC, TLS or other security protocols that support the secure transmission to and from the Verifier.

One way of conveying attestation evidence or the attestation result is the Entity Attestation Token (EAT) [RFC9711]. It provides an attested claims set which can be used to determine a level of trustworthiness. This specification relies on the EAT as the format for attestation evidence and the attestation result.

Ephemeral Diffie-Hellman over COSE (EDHOC) [RFC9528] is a lightweight authenticated key exchange protocol for highly constrained networks. In EDHOC, the two parties involved in the key exchange are referred to as the Initiator (I) and the Responder (R). EDHOC supports the transport of external authorization data, through the dedicated EAD fields. This specification delivers EAT through EDHOC. Specifically, EAT is transported as an EAD item. This specification also defines new EAD items needed to perform remote attesation over EDHOC in Section 5.3 and Section 5.4.

For the generation of evidence, the Attester incorporates an internal attestation service, including a specific trusted element known as the "root of trust". Root of trust serves as the starting point for establishing and validating the trustworthiness appraisals of other components on the system. The measurements signed by the attestation service are referred to as the Evidence. The signing is requested through an attestation API. How the components are separated between the secure and non-secure worlds on a target is out of scope of this specification.

2. Conventions and Definitions

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

The reader is assumed to be familiar with the terms and concepts defined in EDHOC [RFC9528] and RATS architecture [RFC9334].

3. Overview

EDHOC provides the benefit of minimal message overhead and reduced round trips for lightweight authentication between an Initiator and a Responder. However, authentication ensures only identity-level security, and additional integrity assurance may be required through remote attestation. This specification describes how to perform remote attestation over the EDHOC protocol, following the RATS architecture. More importantly, by integrating remote attestation with EDHOC, attestation can be run in parallel with authentication, improving the efficiency and maintaining lightweight properties.

Remote attestation protocol elements are carried within EDHOC’s External Authorization Data (EAD) fields. EDHOC [RFC9528] supports one or more EAD items in each EAD field.

The Attester can act as either the EDHOC Initiator or the EDHOC Responder, depending on the attesting target. In the background-check model, the Attester (EDHOC Initiator/IoT device) exchanges evidence with the Relying Party (EDHOC Responder/Network service) during the EDHOC session. In the passport model, the Attester (EDHOC Responder/Network servie) exhcnages the attestation result with the Relying Party (EDHOC Initiator/IoT device) during the EDHOC session. Section Section 5 defines three independent dimensions for performing remote attestation over EDHOC:

1. Target (see Section 5.1, Section 5.2) defining the entity that undergoes the attestation process (IoT device or network service).

2. Model (see Section 5.3, Section 5.4) defining the attestation model in use based on the RATS architecture (background-check model or passport model).

3. Message Flow (see Section 5.5, Section 5.6) defining the EDHOC message flow in use (forward message flow or reverse message flow).

This document specifies the cases that are suited for constrained IoT environments. See this document [I-D.ietf-iotops-7228bis] as a reference for classification of IoT devices.

The remote attestation operation defined in this document preserves the properties of EDHOC:

1. The EDHOC protocol is not modified, the remote attestation elements are carried within EDHOC EAD fields.

2. The attestation protocol is in parallel but does not interfere with the authentication flow.

3. The privacy and security properties of EDHOC are not changed.

4. Assumptions

In the background-check model, the Verifier is assumed to support verification of at least one evidence format provided by the Attester. The Verifier is assumed to be provisioned with the Attester’s attestation public key and the reference values required for evidence validation prior to the attestation procedure. It is assumed that the Relying Party also has knowledge about the Attester, so it can narrow down the evidence type selection and send to the Attester only one format of the evidence type.

In the passport model, the credential identity of the Verifier is assumed to be stored at the Attester and the Relying Party, which means the Verifier is trusted by the Attester and the Relying Party to obtain the attestation result. If timestamps are used to ensure freshness in the passport model, synchronized time between the Attester and Relying Party is assumed. For detailed time considerations, refer to Appendix A of [RFC9334]. If nonce is used to ensure freshness in the passport model, the IoT device is assumed to be able to generate a random byte string that is not predictable.

5. Remote Attestation in EDHOC

This section specifies three independent dimensions that characterize the remote attestation process over EDHOC.

1. Target: Defines the entity that undergoes the attestation process.

2. Model: Defines the attestation models based on RATS architecture. This specification supports both the RATS background-check model (see Section 5.3) and the passport model (see Section 5.4). The corresponding EAD items for background-check model and the passport model are independent of each other.

3. EDHOC message flow: The EDHOC protocol defines two possible message flows, namely the EDHOC forward message flow and the EDHOC reverse message flow (see Appendix A.2.2 of [RFC9528]). In this specification, both flows can be used to perform remote attestation.

5.1. Target: IoT Device Attestation (IoT)

The IoT device acts as the Attester.

5.2. Target: Network Service Attestation (Net)

The network service acts as the Attester.

This attestation pattern applies when constrained devices need to establish trust with a network gateway. For example, before transmitting sensitive data to a network gateway, a constrained device requires attestation of the network gateway.

5.3. Model: Background-check Model (BG)

In the background-check model, the Attester sends the evidence to the Relying Party. Evidence contains a set of claims about the current status of the Attester, including configurations, health or construction that have security relevance (see Section 8.1 of [RFC9334]). The Relying Party transfers the evidence to the Verifier and gets back the attestation result from the Verifier.

An EDHOC session is established between the Attester and the Relying Party. A negotiation of the evidence type is required before the Attester sends the evidence. All three parties must agree on a selected evidence type.

The Attester first sends a list of the proposed evidence types to the Relying Party. The list is formatted as an Attestation proposal in an EDHOC EAD item. The Relying Party relays the list to the Verifier and receives at least one supported evidence type from the Verifier in return. If the Relying Party receives more than one evidence type, a single evidence type should be selected by the Relying Party based on its knowledge of the Attester. The Relying Party then sends it back within the Attestation request to the Attester.

A nonce, at least 8-bytes long [RFC9711]), guarantees the freshness of the attestation session. The nonce is generated by the Verifier and sent to the Relying Party. The Relying Party puts the nonce and the selected evidence type together in a tuple to generate an Attestation request.

Once the Attester receives the Attestation request, it can call its attestation service to generate the evidence, with the nonce value as one of the inputs.

The ead_label for Attestation_proposal, Attestation_request and Evidence is the same (TBD1) because these EAD items appear in distinct and fixed positions within the EDHOC message sequence. Specifically, they are conveyed in EAD_1, EAD_2, and EAD_3 of EDHOC messaage_1, EDHOC message_2, and EDHOC message_3, respectively. As their positions identify each item, separate labels are not required.

5.3.1. Attestation_proposal

To propose a list of provided evidence types in background-check model, the Attester transports the Proposed_EvidenceType object. It signals to the Relying Party the proposal to do remote attestation, as well as which types of the attestation claims the Attester supports. The Proposed_EvidenceType is encoded in CBOR in the form of a sequence.

The EAD item for an attestation proposal is:

* ead_label = TBD1

* ead_value = Attestation_proposal, which is a CBOR byte string:

```
Attestation_proposal = bstr .cbor Proposed_EvidenceType

Proposed_EvidenceType = [ + content-format ]

content-format = uint
```

where

* Proposed_EvidenceType is an array that contains all the supported evidence types by the Attester.

* There MUST be at least one item in the array.

* content-format is an indicator of the format type (e.g., application/eat+cwt with an appropriate eat_profile parameter set), from [IANA-CoAP-Content-Formats].

The sign of ead_label TBD1 MUST be negative to indicate that the EAD item is critical. If the receiver (EDHOC Responder/Relying Party) cannot recognize the critical EAD item, or cannot process the information in the critical EAD item, then the receiver MUST abort the EDHOC session, as defined in Section 3.8 of [RFC9528].

5.3.2. Attestation_request

As a response to the attestation proposal, the Relying Party signals to the Attester the supported and requested evidence type. In case none of the evidence types is supported, the Relying Party rejects the first message_1 with an error indicating support for another evidence type.

The EAD item for an attestation request is:

* ead_label = TBD1

* ead_value = Attestation_request, which is a CBOR byte string:

```
Attestation_request = bstr .cbor Selected_EvidenceType

Selected_EvidenceType = (
  content-format: uint,
  nonce: bstr .size (8..64)
)
```

where

* content-format is the selected evidence type by the Relying Party and supported by the Verifier.

* nonce is generated by the Verifier and forwarded by the Relying Party.

The sign of ead_label TBD1 MUST be negative to indicate that the EAD item is critical. If the receiver (EDHOC Initiator/Attester) cannot recognize the critical EAD item, or cannot process the information in the critical EAD item, then the receiver MUST abort the EDHOC session, as defined in Section 3.8 of [RFC9528].

5.3.3. Evidence

The Attester calls its local attestation service to generate and return a serialized Entity Attestation Token (EAT) [RFC9711] as Evidence. The Evidence is sent in EAD_3 within EDHOC message_3.

The EAD item is:

* ead_label = TBD1

* ead_value is a serialized EAT in COSE_Sign1 structure.

For remote attestation over EDHOC, The EAT MUST be formatted as a CBOR Web Token (CWT) containing attestation-oriented claims. The complete set of attestation claims for the EAT is specified in [RFC9711]. An example is provided in Section Appendix C.

A minimal claims set is defined as the payload of COSE_Sign1 when the Attester operates under constrained message size requirements and/or limited computational resources:

```
{
/eat-nonce/          10: bstr .size 8
/ueid/               256: bstr .size (7..33)
/measurements/       273: measurements-type
}

measurements-type = [+ measurements-format]
measurements-format = [
      content-format: uint,
      content: bytes
]
```

where

* The "measurements" claim can be a CoSWID [RFC9393]. When CoSWID [RFC9393] is used, the claim MUST be an evidence CoSWID rather than a payload CoSWID. Formats other than CoSWID are permitted and MUST be identified by CoAP Content Format.

In the forward EDHOC message flow, Evidence is sent in EDHOC message_3. The signature over the Evidence MUST include an attestation binder, which is defined as a cryptographic hash of the first two EDHOC messages.

```
attestation_binder = H(message_1, message_2)
```

where

* H() is the EDHOC hash algorithm of the selected cipher suite.

The inclusion of the attestation binder is mandatory to cryptographically bind the attestation to the authentication, and to ensure that the attester is the authenticated peer. The attestation binder prevents relay attacks whereby an attacker relays Evidence generated in a different session.

The signature in COSE_Sign1 is computed over a Sig_structure containing protected header, externally supplied data (external_aad) and payload using a private attestation key. The message to be signed is:

["Signature1", body_protected, external_aad, payload]

where

* body_protected is the same CBOR byte string as the protected header in COSE_Sign1

* external_aad = attestation_binder

* payolad is the same CBOR byte string as the payload in COSE_Sign1

In the reverse EDHOC message flow, Evidence is sent in EDHOC message_4. The signature over the Evidence MUST include an attestation binder, which is derived using EDHOC_Exporter defined in [RFC9528].

attestation_binder = EDHOC_Exporter (exporter_label, context, length)

where

* exporter_label = 2

* context = "attestation_binder"

* length = 32

5.4. Model: Passport Model (PP)

In the passport model, the Attester sends the evidence to the Verifier. After the Attester receives the attestation result from the Verifier, the Attester sends the attestation result to the Relying Party. The attestation result may carry a boolean value indicating compliance or non-compliance with a Verifier’s apprasal policy, or many carry a set of claims to indicate the results in different aspects (Section 8.4 of [RFC9334]).

An EDHOC session is established between the Attester (EDHOC Responder) and the Relying Party (EDHOC Initiator). The Attester and the Relying Party should decide from which Verifier the Attester obtains the attestation result and transfers it to the Relying Party. The Attester first sends a list of the Verifier identities that it can get the attestation result. The Relying Party selects one trusted Verifier identity and sends it back as a Result request.

Regarding the freshness in passport model, the Attester could either establish a real-time connection with the selected Verifier, or use a pre-stored attestation result from the selected Verifier. If the attestation result is not obtained via a real-time connection, it MUST include a time stamp and/or expiry time to indicate its validity. Time synchronization is out of scope of this specification.

Once the Attester obtains the attestation result from the selected Verifier, it sends the attestation result to the Relying Party.

The ead_label for Result_proposal, Result_request and Result is the same (TBD2) because these EAD items appear in distinct and fixed positions within the EDHOC message sequence. Specifically, they are conveyed in EAD_2, EAD_3, and EAD_4 of EDHOC messaage_2, EDHOC message_3, and EDHOC message_4, respectively. As their positions identify each item, separate labels are not required.

5.4.1. Result_proposal

An attestation result proposal contains the identification of the credentials of the Verifiers to indicate Verifiers' indentities. The identification of credentials relies on COSE header parameters [IANA-COSE-Header-Parameters], with a header label and credential value.

The EAD item for the attestation result proposal is:

* ead_label = TBD2

* ead_value = Result_proposal, which is a CBOR byte string:

```
Result_proposal = bstr .cbor Proposed_VerifierIdentity
Proposed_VerifierIdentity = [ + VerifierIdentity ]

VerifierIdentity = {
  label => values
}
```

where

* Proposed_VerifierIdentity is defined as a list of one or more VerifierIdentity elements.

* Each VerifierIdentity within the list is a map defined in [IANA-COSE-Header-Parameters] that:

  - label = int / tstr

- values = any

5.4.2. Result_request

As a response to the attestation result proposal, the Relying Party signals to the Attester the trusted Verifier. In case none of the Verifiers can be trusted by the Relying Party, the session is aborted. Relying Party generates a nonce to ensure the freshness of the attestation result from the Verifier.

The EAD item for an attestation result request is:

* ead_label = TBD2

* ead_value = Result_request, which is a CBOR byte string:

Result_request = bstr .cbor Request_structure

```
Request_structure = {
<<<<<<< version04
  selected_verifier: VerfierIdentity,
  ? nonce: bstr .size (8..64)
=======
  selected_verifier: VerifierIdentity,
  ? nonce: bstr .size 8..64
>>>>>>> main
}
```

5.4.3. Result

The attestation result is generated and signed by the Verifier as a serialized EAT [RFC9711]. The Relying Party can decide what action to take with regards to the Attester based on the information elements in attetation result.

The EAD item is:

* ead_label = TBD2

* ead_value is a serialized EAT.

5.4.4. trigger_pp

The EAD item trigger_pp is used when the sender (EDHOC Initiator/ Relying Party) triggers the receiver (EDHOC Responder/Attester) to start a remote attestation in the passport model. The receiver MUST reply with an EAD item correspondign to the passport model, either a result proposal in Section 5.4.1 or a result in Section 5.4.3. The ead_value MUST not be present, as the ead_label serves as the trigger.

The EAD item is:

* ead_label = TBD3

* ead_value = null

5.5. EDHOC forward Message Flow (Fwd)

In forward message flow, EDHOC may run with the Initiator as a CoAP/ HTTP client. Remote attestation over EDHOC starts with a POST requests to the Uri-Path: "/.well-known/lake-ra".

5.6. EDHOC reverse Message Flow (Rev)

In the reverse message flow, the CoAP/HTTP client is the Responder and the server is the Initiator. A new EDHOC session begins with an empty POST request to the server’s resource for EDHOC.

6. Instantiation of Remote Attestation Protocol

We use the format (Target, Model, Message Flow) to denote instantiations. For example, (IoT, BG, Fwd) represents IoT device attestation using the background-check model with forward EDHOC message flow.

There are 8 possible instantiations combining IoT/Net, BG/PP, and Fwd/Rev configurations. Considering practical feasibility, IoT device attestation in the passport model and Network Service attestation in the background model are less viable. These instantiations require IoT devices to maintain multiple network connections and perform complex computations, which exceed the capabilities of typical low-power, resource-constrained IoT devices.

In this document, only the most relevant instantiations for constrained IoT environments are specified.

6.1. (IoT, BG, Fwd): IoT Device Attestation

A common use case for (IoT, BG, Fwd) is to attest an IoT device to a network server. For example, a simple illustratice case is doing remote attestation to verify that the latest version of firmware is running on the IoT device before the network server allows it to join the network (see Appendix C).

An overview of doing IoT device attestation in background-check model and EDHOC forward message flow is established in Figure 1. EDHOC Initiator plays the role of the RATS Attester (A). EDHOC Responder plays the role of the RATS Relying Party (RP). The Attester and the Relying Party communicate by transporting messages within EDHOC's External Authorization Data (EAD) fields. An external entity, out of scope of this specification, plays the role of the RATS Verifier (V). The EAD items specific to the background-check model are defined in Section 5.3.

The Attester starts the attestation by sending an Attestation proposal in EDHOC message_1. The Relying Party generates EAD_2 with the received evidence type and nonce from the Verifier, and sends it to the Attester. The Attester sends the Evidence to the Relying Party in EAD_3. The Verifier evaluates the Evidence and sends the Attestation result to the Relying Party.

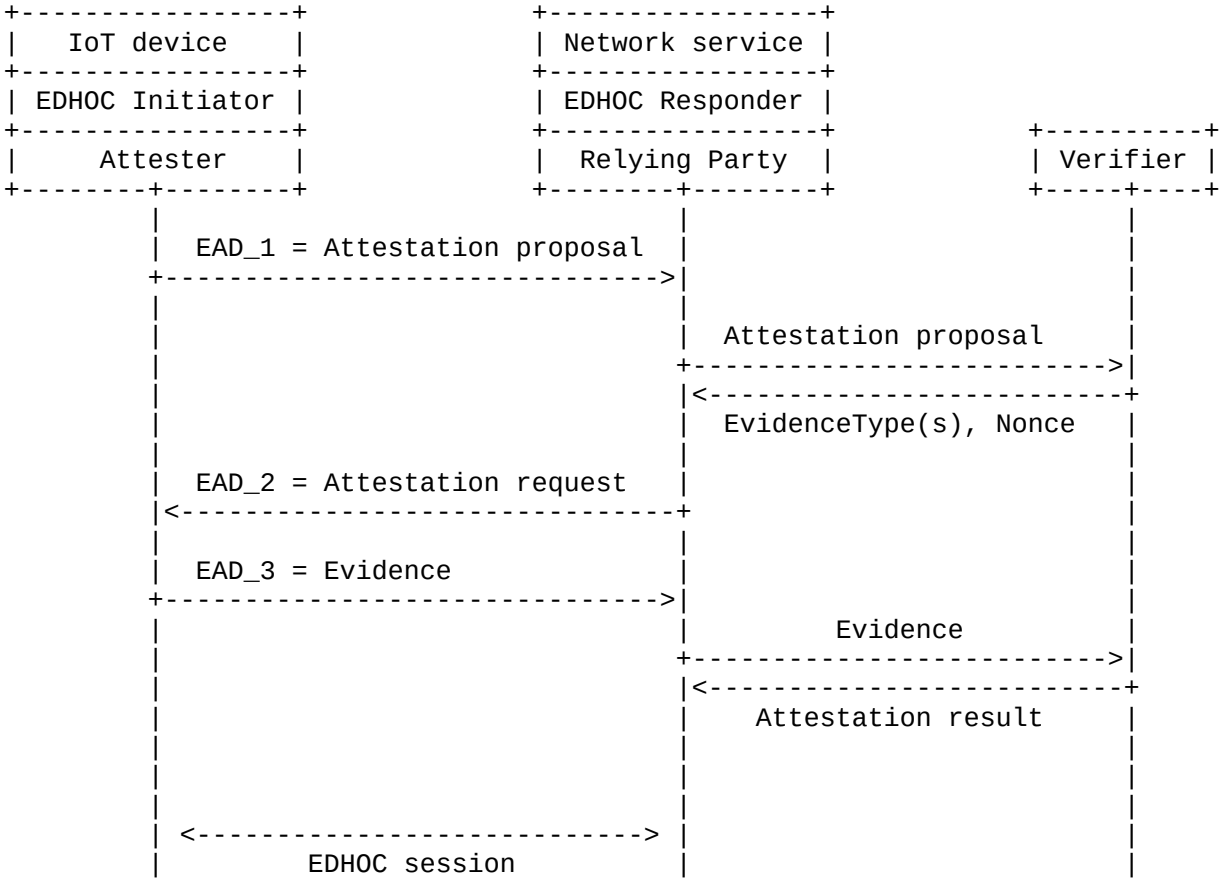


Figure 1: Overview of IoT device attestation in background-check model and EDHOC forward message flow. EDHOC is used between A and RP.

6.2. (Net, PP, Fwd): Network Service Attestation

One use case for (Net, PP, Fwd) is when a network server needs to attest itself to a client (e.g., an IoT device). For example, the client needs to send some sensitive data to the network server, which requires the network server to be attested first. In (Net, PP, Fwd), the network server acts as an Attester and the client acts as a Relying Party.

An overview of the message flow is illustrated in Figure 2. EDHOC Initiator plays the role of the RATS Relying Party. EDHOC Responder plays the role of the RATS Attester. An external entity, out of scope of this specification, plays the role of the RATS Verifier. The EAD items specific to the passport model are defined in Section 5.4.

The Relying Party asks the Attester to do a remote attestation by sending a trigger_pp (see Section 5.4.4) in EDHOC message_1. The Attester replies to the Relying Party with a Result proposal in EAD_2. Then the Relying Party selects a trusted Verifier identity and sends it as a Result request. How the Attester negotiates with the selected Verifier to get the attestation result is out of scope of this specification. A fourth EDHOC message is required to send the Result from the Attester to the Relying Party.

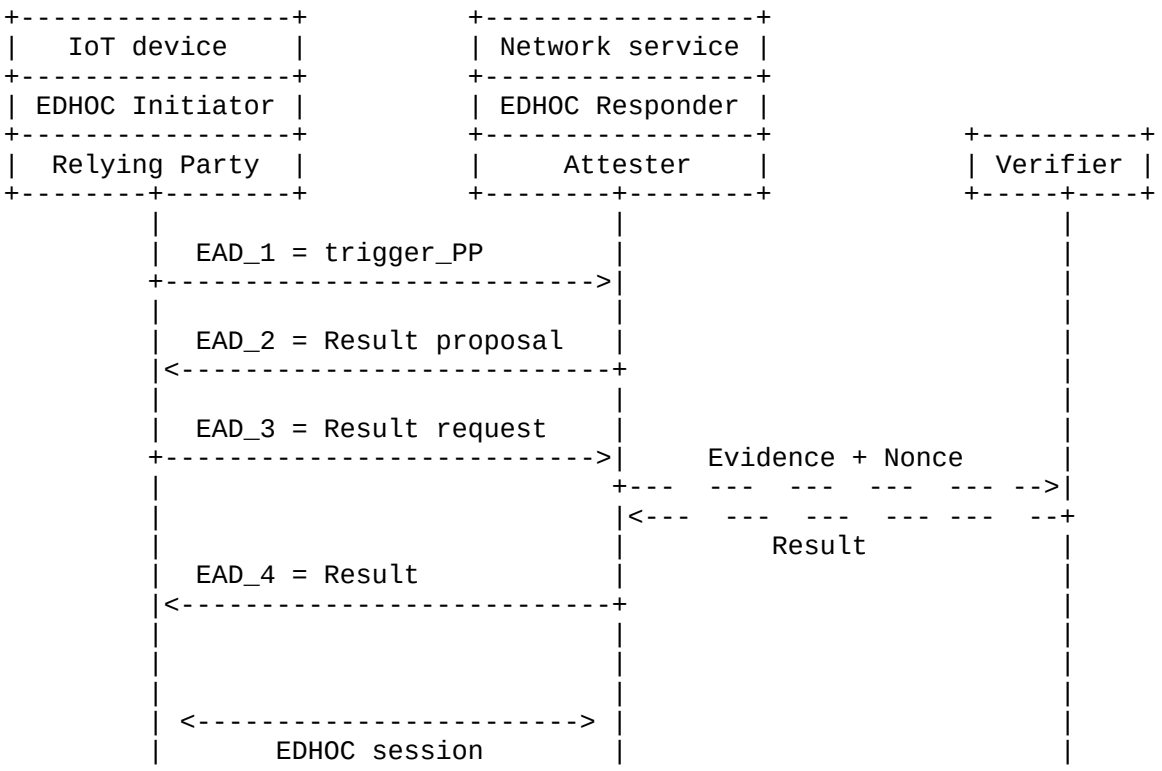


Figure 2: Overview of network service attestation in passport model and EDHOC forward message flow. EDHOC is used between RP and A. The dashed line illustrates a logical connection that does not need to occur in real time.

7. Mutual Attestation in EDHOC

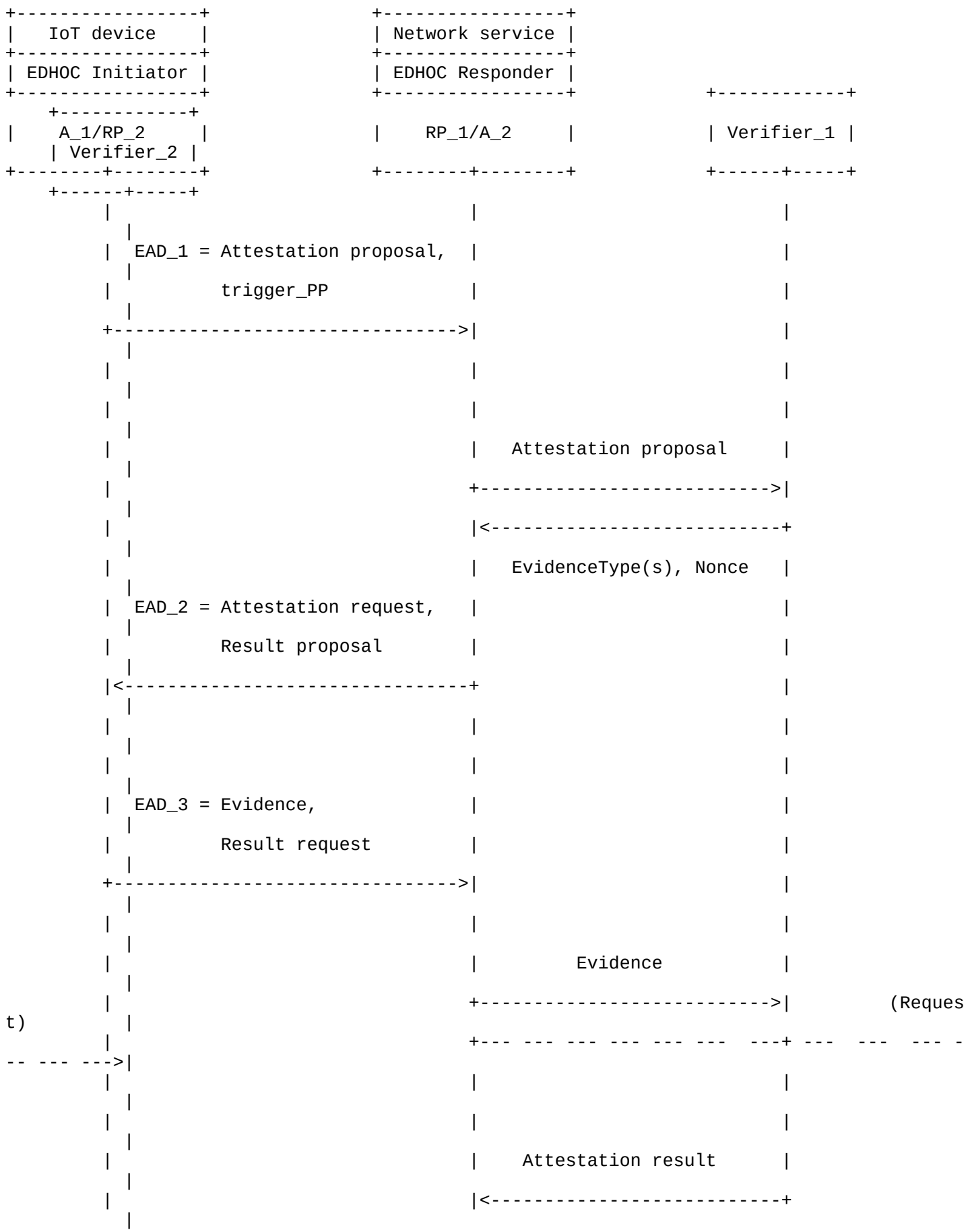
Mutual attestation over EDHOC combines the cases where one of the EDHOC parties uses the IoT device attestation and the other uses the Network service attestation. Performing mutual attestation to a single EDHOC message flow acheives a lightweight use with reduced message overhead. Note that the message flow for the two parties in mutual attestation needs to be the same.

In this specification, we list the most relevant mutual attestation example for constrained IoT environments.

7.1. (IoT, BG, Fwd) - (Net, PP, Fwd)

In this example, the mutual attestation is performed in EDHOC forward message flow, by one IoT device attestation in background-check model and another network service attestation in passport model. The process is illustrated in Figure 3. How the Network service connects with the Verifier_1 and potential Verifier_2 is out of scope in this specification.

The first remote attestation is initiated by the IoT device (A_1) in background-check model. In parallel, the IoT device (A_1) requests the network service (A_2) to perform a remote attestation in passport model. EAD_2 carries the EAD items Attestation request and Result proposal. EAD_3 carries the EAD items Evidence and Result request. EAD_4 carries the EAD item Result for the passport model.


IoT device
EDHOC Initiator
Network service
EDHOC Responder
A_1/RP_2
Verifier_2
RP_1/A_2
Verifier_1
EAD_1 = Attestation proposal,
trigger_PP
Attestation proposal
EvidenceType(s), Nonce
EAD_2 = Attestation request,
Result proposal
EAD_3 = Evidence,
Result request
Evidence
(Request)
Attestation result

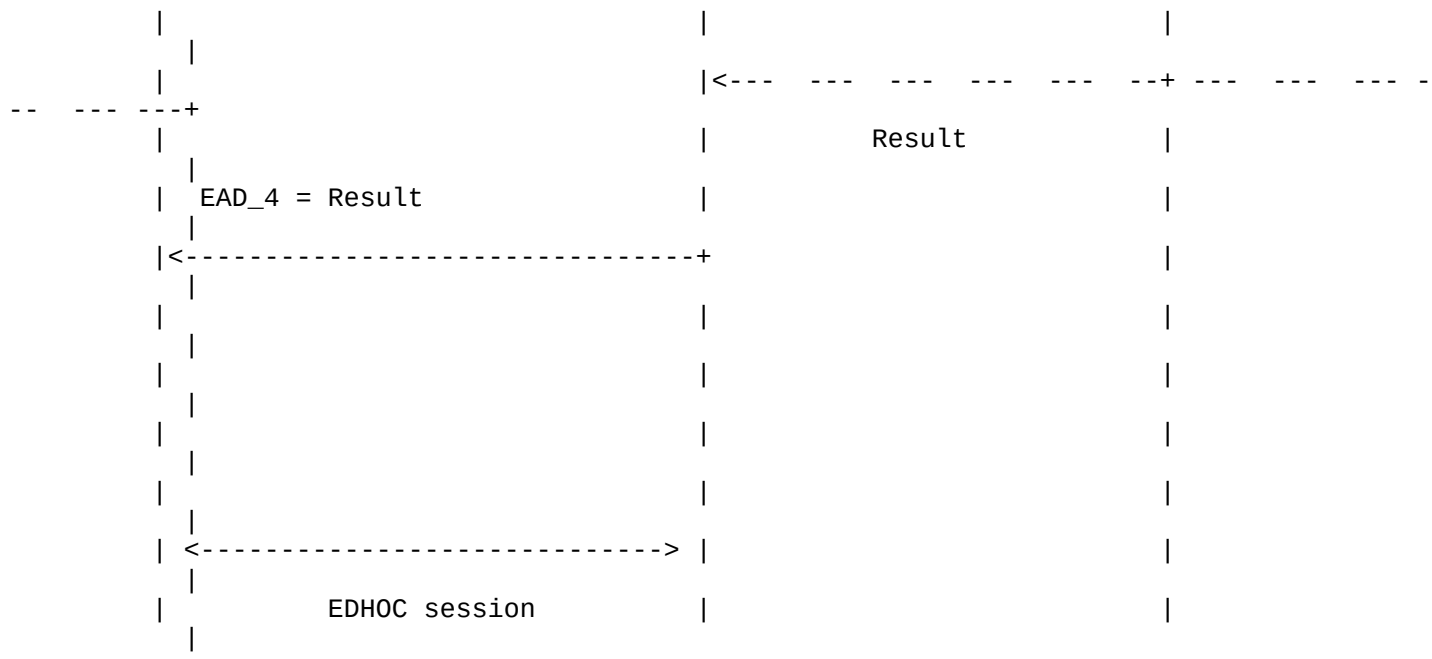

Result
EAD_4 = Result
EDHOC session

Figure 3: Overview of mutual attestation of (IoT, BG, Fwd) - (Net, PP, Fwd). EDHOC is used between A and RP. The dashed line illustrates a logical connection that does not need to occur in real time.

8. Verifier

The Verifier maintains an explicit trust relationship with the Attester, whereby the Verifier is provisioned with the Attester's attestation public key prior to the remote attestation process. This explicit relationship may be established through various means, such as manufacturer provisioning, trusted certification authorities, or direct configuration. Reference values used for comparison against received evidence should also be provided to the Verifier before the attesation. The evaluation policy employed by the Verifier varies according to specific use cases and should be determined prior to the attestation; such policy definition is out of scope in this specification.

The Verifier maintains an implicit trust relationship with the Relying Party, established through mechanisms such as web PKI with trusted Certificate Authority (CA) certificates, enabling the Relying Party to trust the attestation result that is generated by the Verifier.

8.1. Processing in the Background-check Model

The Verifier is connected with the Relying Party and is responsible for evaluating evidence forwarded by the Relying Party. After the Relying Party receives EDHOC message_1 from the Attester, it extracts and transmits the Attestation proposal to the Verifier. The Verifier must support at least one evidence type for evaluation, otherwise it returns an empty list. Alongside the selected evidence type, the Verifier generates a random nonce and sends both elements to the Relying Party.

When the Relying Party obtains EDHOC message_3, it forwards the evidence and the attestation binder (hash of the first two EDHOC messages) to the Verifier for evaluation.

The evidence evaluation process SHOULD include the signature verification, nonce validation, and comparison of measurement values against trusted reference values. An example evaluation procedure for evidence formatted as an Entity Attestation Token (EAT) within a COSE_Sign1 structure is as follows:

1. Decode the COSE_Sign1 structure and extract constituent components: headers, payload, signature.

2. Verify the signature using the Attester’s attestation public key. The Verifier reconstructs the Sig_Structure, with the attestation binder carried in the external_add field.

3. Verify that the nonce exists in the Verifier’s local nonce list. If the nonce is found, validation passes and the nonce is removed from the list to prevent replay attacks.

4. Compare the received evidence measurement values against the reference value. The attestation result is returned to the Relying Party, with result generation conforming to the attestation token format defined in [RFC9711].

8.2. Processing in the Passport Model

When the Attester utilizes a cached attestation result previously generated by the Verifier, real-time re-evaluation by the Verifier is not required. If the Attester receives result_request from the Relying Party and performs real-time attestation with the Verifier, the Verifier then generates the attestation result formatted as an Entity Attestation Token (EAT). The token uses the "Software Measurement Results (measres)" claim as defined in [RFC9711], and incorporates the nonce generated by the Relying Party as an input parameter.

9. Security Considerations

This specification is performed over EDHOC [RFC9528] by using EDHOC’s EAD fields. The privacy considerations of EADs in EDHOC apply to this specification.

EAD_1 is not resistant to either active attackers or passive attackers, because neither the Initiator nor the Responder has been authenticated.

Although EAD_2 is encrypted, the Initiator has not been authenticated, rendering EAD_2 vulnerable against the active attackers.

The ead items in EAD_1 and EAD_2 MAY be very specific and potentially reveal sensitive information about the device. The leaking of the data in EAD_1 and/or EAD_2 MAY risk to be used by the attackers for malicious purposes. Data in EAD_3 and EAD_4 are protected between the Initiator and the Responder in EDHOC.

Mutual attestation carries a lower risk for EAD items when the Responder is the Attester. For the mutual attestation at the EDHOC Responder, only the Attestation_proposal/Result_proposal in EAD_2 is

not protected to active attackers. Both the Attestation_request/ Result_request in EAD_3 and the Evidence/Result in EAD_4 are protected.

The privacy considerations of remote attestation refer to Section 11 of [RFC9334].

10. IANA Considerations

10.1. EDHOC External Authorization Data Registry

IANA is requested to register the following entry in the "EDHOC External Authorization Data" registry under the group name "Ephemeral Diffie-Hellman Over Cose (EDHOC)".

* The ead_label = TBD1 corresponds to the ead_value Attestation_proposal in Section 5.3.1, Attestation_request in Section 5.3.2 and Evidence in Section 5.3.3.

* The ead_label = TBD2 corresponds to the ead_value Result_proposal in Section 5.4.1, Result_request in Section 5.4.2 and the Result in Section 5.4.3.

* The ead_label = TBD3 corresponds to the EAT item trigger_pp as specified in Section 5.4.4.

Name	Label	Description	Reference
TBD	TBD1	BG model related information	Section 5.2.1.1
TBD	TBD2	PP model related information	Section 5.2.2.1
TBD	TBD3	trigger to start attestation in PP	Section 5.2.2.1

Figure 4: EAD labels.

11. References

11.1. Normative References

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

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

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

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <https://www.rfc-editor.org/rfc/rfc9711>.

11.2. Informative References

[I-D.ietf-iotops-7228bis] Bormann, C., Ersue, M., Kernen, A., and C. Gomez, "Terminology for Constrained-Node Networks", Work in Progress, Internet-Draft, draft-ietf-iotops-7228bis-03, 4 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-iotops-7228bis-03>.

[I-D.ietf-lake-authz] Selander, G., Mattsson, J. P., Vuini, M., Fedrecheski, G., and M. Richardson, "Lightweight Authorization using Ephemeral Diffie-Hellman Over COSE (ELA)", Work in Progress, Internet-Draft, draft-ietf-lake-authz-06, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-authz-06>.

[IANA-CoAP-Content-Formats] "CoAP Content-Formats", n.d., <https://www.iana.org/assignments/core-parameters>.

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

[IANA.CWT.Claims] IANA, "CBOR Web Token (CWT) Claims", <https://www.iana.org/assignments/cwt>.

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

[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/rfc/rfc8613>.

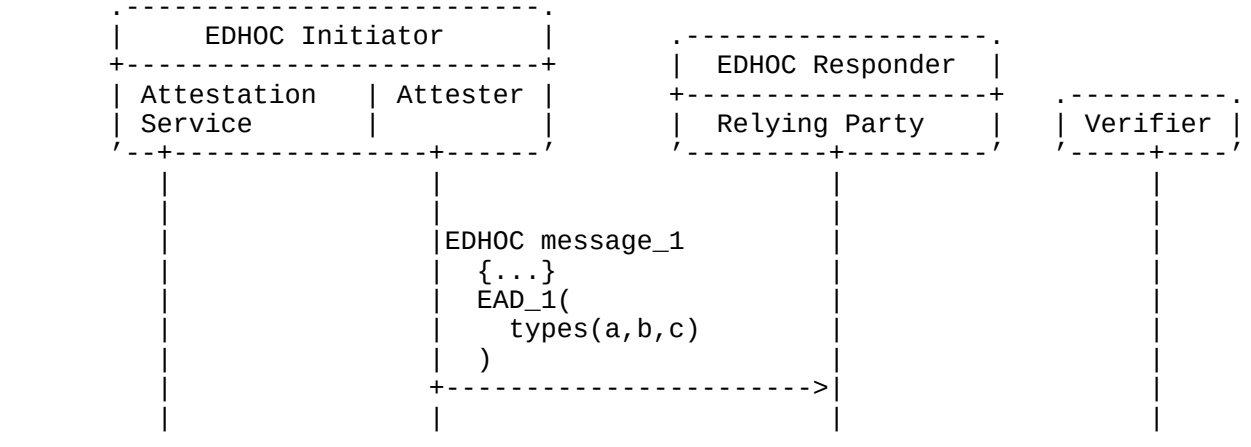
[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

[RFC9393] Birkholz, H., Fitzgerald-McKay, J., Schmidt, C., and D. Waltermire, "Concise Software Identification Tags", RFC 9393, DOI 10.17487/RFC9393, June 2023, <https://www.rfc-editor.org/rfc/rfc9393>.

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

Appendix A. Example: Remote Attestation Flow



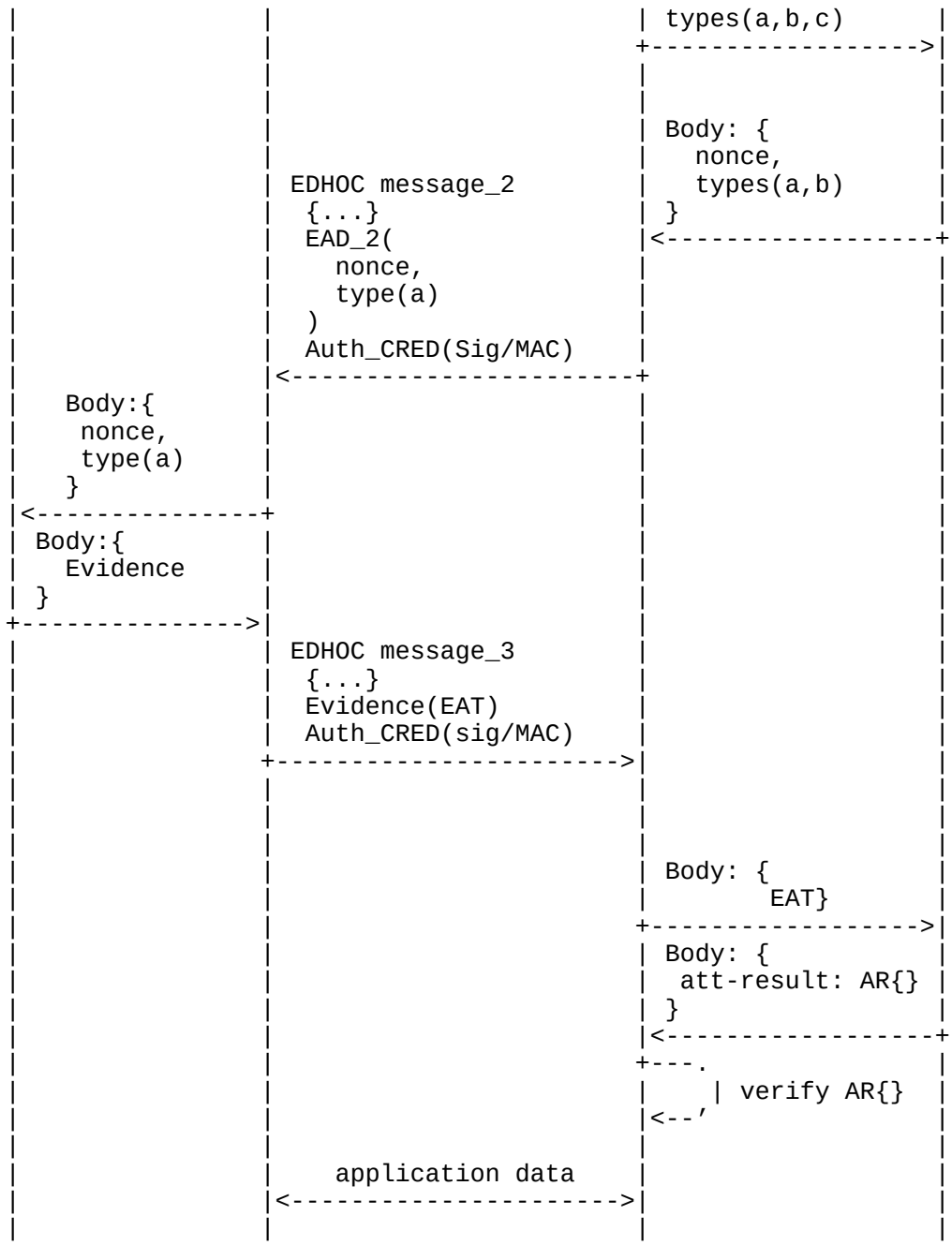


Figure 5: Example of remote attestation.

Appendix B. Remote attestation in parallel with enrollment authorization

This section discusses the possibility of doing remote attestation in parallel with the enrollment authorization procedure defined in [I-D.ietf-lake-authz]. In this case, the message count is much decreased.

The detailed procedure is TBD.

Appendix C. Example: Firmware Version

The goal in this example is to verify that the firmware running on the device is the latest version, and is neither tampered or compromised. A device acts as the Attester, currently in an untrusted state. The Attester needs to generate the evidence to attest itself. A gateway that can communicate with the Attester and can control its access to the network acts as the Relying Party. The gateway will finally decide whether the device can join the network or not depending on the attestation result. The attestation result is produced by the Verifier, which is a web server that can be seen as the manufacturer of the device. Therefore it can appraise the evidence that is sent by the Attester. The remote attestation session starts with the Attester sending EAD_1 in EDHOC message 1.

An example of the EAD_1 in EDHOC message_1 could be:

[60,61,258]

If the Verifier and the Relying Party can support at least one evidence type that is proposed by the Attester, the Relying Party will include in the EAD_2 field the same evidence type, alongside a nonce for message freshness.

(258, h’a29f62a4c6cdaae5’)

The Evidence in EAD_3 field is an Entity Attestation Token (EAT) [RFC9711], with the measurements claim formatted in CoSWID[RFC9393]. The Evidence is in COSE_Sign1 structure, where the payload of COSE_Sign1 contains the following claims:

```
{
/eat-nonce/                10: h'a29f62a4c6cdaae5',
/ueid/                     256: 'aaabbcc',
/measurements/             273: [
  /CoAP Content-Format ID/       [ 258,
  /evidence in CoSWID/             {
                                     0: 'tagID'             /tag-id/
                                     12: 0                  /tag-version/
                                     1: "DotBot firmware"   /software-name/
                                     2: {                   /entity/
                                         31: "Attester"     /entity-name/
                                         33: 1              /role, must be "tag-creator"
 which is 1/
                                         },
                                     3: {                   /evidence/
                                         17: [              /file/
                                              {
                                               24: "partition0-nrf52840dk.bin",   /fs-na
me/
                                               7: [                               /hash
of file/
                                                   1,                             /alg S
HA-256/
                                                   h'06294f6806b9c685eea795048579cfd02a0
c025bc8b5abca42a19ea0ec23e81a'
                                                   ]                              /hash
value/
                                              }
                                             ]
                                         }
                                   }
                                 ]
                               ]
}
```

The infomation above serves as the payload of the COSE object. The Sig_structure to compute the signature of COSE_Sign1 is:

```
Sig_structure = [
    "Signature1",
    h'a10127',      /ED25519 algorithm, same as the protected header in COSE_Sign1/
    h'7b4c94f32a0e6db86d915a444f76525fc32912b2e07dd481a96f627ee98a110c',      /hash of th
e first two EDHOC messages/
    h'A30A48A29F62A4C6CDAAE5190100476161616262636319011181821901 02A50045746
    16749440C00016F446F74426F74206669726D7761726502A2181F684174746573746572
    18210103A11181A2181878197061727469746F6E302D6E72663532383430646B2E626
    96E078201582006294F6806B9C685EEA795048579CFD02A0C025BC8B5ABCA42A19EA0EC
    23E81A',                                                                  /same as t
he payload in COSE_Sign1/
]
```

The complete resulting COSE object is:

```
18([
    /*protected header*/
    h'a10127',

    /*unprotected header*/
    {},

    /*payload*/
    h'A30A48A29F62A4C6CDAAE5190100476161616262636319011181821901 02A50045746
    16749440C00016F446F74426F74206669726D7761726502A2181F684174746573746572
    18210103A11181A2181878197061727469746F6E302D6E72663532383430646B2E626
    96E078201582006294F6806B9C685EEA795048579CFD02A0C025BC8B5ABCA42A19EA0EC
    23E81A',

    /*signature*/
    h'd4100901f4c3e51312c3110c6ddc8dcf7f68d8f5d3791c19133f2f0ac158c1f5ee6ed
    afe9d7c3d6eb3d2d197f82e733d375fdda9fb258b304961dfc38558950d'
])
```

which has the following base16 encoding:

```
D28443A10127A05890A30A48A29F62A4C6CDAAE519010047616161626263631901118 18
2190102A5004574616749440C00016F446F74426F74206669726D7761726502A2181F68
41747465737465721821 0103A11181A2181878197061727469746F6E302D6E7266353
2383430646B2E62696E078201582006294F6806B9C685EEA795048579CFD02A0C025BC8
B5ABCA42A19EA0EC23E81A5840D4100901F4C3E51312C3110C6DDC8DCF7F68D8F5D3791
C19133F2F0AC158C1F5EE6EDAFE9D7C3D6EB3D2D197F82E733D375FDDA9FB258B304961
DFC38558950D
```

The Relying Party (co-located with the gateway) then treats the Evidence as opaque and sends it together with the hash value of the first two EDHOC messages to the Verifier. Once the Verifier sends back the Attestation Result, the Relying Party can be assured on the version of the firmware that the device is running.

Appendix D. Post-handshake Attestation over OSCORE

Beyond the intra-handshake attestation defined in this document, remote attestation after an EDHOC session can be performed via post-handshake attestation over Object Security for Constrained RESTful Environments (OSCORE) [RFC8613]. OSCORE provides application-layer protection for the Constrained Application Protocol (CoAP) using CBOR Object Signing and Encryption (COSE). As specified in [RFC9668], EDHOC can run over CoAP to establish a Security Context for OSCORE.

Post-handshake attestation decouples attestation process from the initial handshake, enabling continuous attestation throughout the session lifetime and guaranteeing the runtime integrity.

D.1. Mapping Attestation to OSCORE

This section outlines how remote attestation is performed in the background-check model over OSCORE.

EDITOR NOTE: put a figure

D.1.1. Flight 1

The CoAP client (Attester) initiates attestation with a CoAP request:

* The request method is POST.

* The Uri-path is set to ".well-known/attest".

* The payload is the Attestation_proposal CBOR sequence, where Attestation_proposal is constructed as defined in Section 5.3.1.

D.1.2. Flight 2

The CoAP server (Relying Party) receives the request and processes it as described in Section 5.3 then prepares Attestation_request as defined in Section 5.3.2. The CoAP server maps the message to a CoAP response:

* The response code is 2.04 Changed.

* The payload is the Attestation_request CBOR sequence, as defined in Section 5.3.2.

D.1.3. Flight 3

The CoAP client (Attester) receives the response and processes it as described in Section 5.3 then generates the evidence. The CoAP client maps the message to a CoAP request:

* The request method is POST.

* The Uri-path is set to ".well-known/attest".

* The payload is the Evidence CBOR sequence, as defined in Section 5.3.3.

D.2. Differences between Intra-handshake and Post-handshake Attestation

Intra-handshake attestation embeds evidence exchange into the authentication handshake, cryptographically tying identity to device state and blocking unauthenticated and untrusted devices from completing the handshake. Post-handshake attestation separates attestation from the authentication handshake, providing modularity that allows attestation mechanisms to be integrated independently of the handshake protocol. This section compares the two different types of remote attestation methods in terms of performance and security properties.

D.2.1. Performance properties

Intra-handshake attestation provides round-trip efficiency as attestation occurs within the handshake, requiring no additional round trips. The intra-handshake attestation defined in this document does not modify the EDHOC protocol itself, though other intra-handshake designs may require changes. Post-handshake has higher modularity which the attestation is integrated independently but it requires additional round trips.

D.2.2. Security properties

Remote attestation provides security properties including evidence integrity, freshness guarantees, and replay protection. Besides, intra-handshake attestation is cryptographically bound to the authentication process, and the trust is established before the session begins. Post-handshake attestation gurantees the runtime integrity which can obtain dynamic mesurements during device execution, and can be repeatedly performed throughout the session which has the continuous assurance.

Acknowledgments

The author would like to thank Thomas Fossati, Malisa Vucinic, Ionut Mihalcea, Muhammad Usama Sardar, Michael Richardson, Geovane Fedrecheski and John Mattsson for the provided ideas and feedback.

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

Authors' Addresses

Yuxuan Song
Inria
Email: yuxuan.song@inria.fr

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