Workgroup: ACE Working Group
Internet-Draft:
draft-ietf-ace-edhoc-oscore-profile-01
Published: 10 March 2023
Intended Status: Standards Track
Expires: 11 September 2023
Authors: G. Selander   J. Preuß Mattsson   M. Tiloca
Ericsson   Ericsson   RISE
R. Höglund
RISE

Ephemeral Diffie-Hellman Over COSE (EDHOC) and Object Security for Constrained Environments (OSCORE) Profile for Authentication and Authorization for Constrained Environments (ACE)

Abstract

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework. It utilizes Ephemeral Diffie-Hellman Over COSE (EDHOC) for achieving mutual authentication between an OAuth 2.0 Client and Resource Server, and it binds an authentication credential of the Client to an OAuth 2.0 access token. EDHOC also establishes an Object Security for Constrained RESTful Environments (OSCORE) Security Context, which is used to secure communications when accessing protected resources according to the authorization information indicated in the access token. A resource-constrained server can use this profile to delegate management of authorization information to a trusted host with less severe limitations regarding processing power and memory.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines the "coap_edhoc_oscore" profile of the ACE framework [RFC9200]. This profile addresses a "zero-touch" constrained setting where trusted operations can be performed with low overhead without endpoint specific configurations.

Like in the "coap_oscore" profile [RFC9203], also in this profile a client (C) and a resource server (RS) use the Constrained Application Protocol (CoAP) [RFC7252] to communicate, and Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] to protect their communications. Also, the processing of requests for specific protected resources is identical to what is defined in the "coap_oscore" profile.

When using this profile, C accesses protected resources hosted at RS with the use of an access token issued by a trusted authorization server (AS) and bound to an authentication credential of C. This differs from the "coap_oscore" profile, where the access token is bound to a symmetric key used to derive OSCORE keying material. As recommended in [RFC9200], this document recommends the use of CBOR Web Tokens (CWTs) [RFC8392] as access tokens.

The authentication and authorization processing requires C and RS to have access to each other's authentication credentials. C can obtain the authentication credential of RS from AS together with the access token. RS can obtain the authentication credential of C together with the associated access token in different ways. If RS successfully verifies the access token, then C and RS run the Ephemeral Diffie-Hellman Over COSE (EDHOC) protocol [I-D.ietf-lake-edhoc] using the authentication credentials.

Once the EDHOC execution is completed, C and RS are mutually authenticated and can derive an OSCORE Security Context to protect subsequent communications.

An authentication credential can be a raw public key, e.g., encoded as a CWT Claims Set (CCS, [RFC8392]); or a public key certificate, e.g., encoded as an X.509 certificate or as a CBOR encoded X.509

certificate (C509, [I-D.ietf-cose-cbor-encoded-cert]); or a different type of data structure containing the public key of the peer in question.

The ACE protocol establishes what those authentication credentials are, and may transport the actual authentication credentials by value or uniquely refer to them. If an authentication credential is pre-provisioned or can be obtained over less constrained links, then it suffices that ACE provides a unique reference such as a certificate hash (e.g., by using the COSE header parameter "x5t", see [RFC9360]). This is in the same spirit as EDHOC, where the authentication credentials may be transported or referenced in the ID_CRED_x message fields (see Section 3.5.3 of [I-D.ietf-lake-edhoc]).

In general, AS and RS are likely to have trusted access to each other's authentication credentials, since AS acts on behalf of RS as per the trust model of ACE. Also, AS needs to have some information about C, including the relevant authentication credential, in order to identify C when it requests an access token and to determine what access rights it can be granted. However, the authentication credential of C may potentially be conveyed (or uniquely referred to) within the request for access which C makes to AS.

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.

Certain security-related terms such as "authentication", "authorization", "confidentiality", "(data) integrity", "Message Authentication Code (MAC)", "Hash-based Message Authentication Code (HMAC)", and "verify" are taken from [RFC4949].

RESTful terminology follows HTTP [RFC9110].

Readers are expected to be familiar with the terms and concepts defined in CoAP [RFC7252], OSCORE [RFC8613] and EDHOC [I-D.ietf-lake-edhoc].

Readers are also expected to be familiar with the terms and concepts of the ACE framework described in [RFC9200] and in [RFC9201].

Terminology for entities in the architecture is defined in OAuth 2.0 [RFC6749], such as the client (C), the resource server (RS), and the authorization server (AS). It is assumed in this document that a given resource on a specific RS is associated with a unique AS.

Note that the term "endpoint" is used here, as in [RFC9200], following its OAuth definition, which is to denote resources such as /token and /introspect at AS and /authz-info at RS. The CoAP [RFC7252] definition, which is "An entity participating in the CoAP protocol" is not used in this document.

The authorization information (authz-info) resource refers to the authorization information endpoint as specified in [RFC9200]. The term "claim" is used in this document with the same semantics as in [RFC9200], i.e., it denotes information carried in the access token or returned from introspection.

This document defines "token series" as a series of access tokens sorted in chronological order as they are released, characterized by the following properties:

* issued by the same AS

* issued to the same C and for the same RS

* issued together with the same authentication credential of RS

* associated with the same authentication credential of C

When an access token becomes invalid (e.g., due to its expiration or revocation), the token series it belongs to ends.

Concise Binary Object Representation (CBOR) [RFC8949][RFC8742] and Concise Data Definition Language (CDDL) [RFC8610] are used in this document. CDDL predefined type names, especially bstr for CBOR byte strings and tstr for CBOR text strings, are used extensively in this document.

Examples throughout this document are expressed in CBOR diagnostic notation without the tag and value abbreviations.

2. Protocol Overview

This section gives an overview of how to use the ACE framework [RFC9200] together with the authenticated key establishment protocol EDHOC [I-D.ietf-lake-edhoc]. By doing so, a client (C) and a resource server (RS) generate an OSCORE Security Context [RFC8613] associated with authorization information, and use that Security Context to protect their communications. The parameters needed by C to negotiate the use of this profile with the authorization server (AS), as well as the OSCORE setup process, are described in detail in the following sections.

RS maintains a collection of authentication credentials. These are related to OSCORE Security Contexts associated with authorization information for all the clients that RS is communicating with. The authorization information is maintained as policy that is used as input to the processing of requests from those clients.

This profile specifies how C requests an access token from AS for the resources it wants to access on an RS, by sending an access token request to the /token endpoint, as specified in Section 5.8 of [RFC9200]. The access token request and response MUST be confidentiality protected and ensure authenticity. The use of EDHOC and OSCORE between C and AS is RECOMMENDED in this profile, in order to reduce the number of libraries that C has to support. However, other protocols fulfilling the security requirements defined in Section 5 of [RFC9200] MAY alternatively be used, such as TLS [RFC8446] or DTLS [RFC9147].

If C has retrieved an access token, there are two different options for C to upload it to RS, as further detailed in this document.

1. C posts the access token to the /authz-info endpoint by using the mechanisms specified in Section 5.10 of [RFC9200]. If the access token is valid, RS responds to the request with a 2.01 (Created) response, after which C initiates the EDHOC protocol by sending EDHOC message_1 to RS. The communication with the /authz-info endpoint is not protected, except for the update of access rights.

2. C initiates the EDHOC protocol by sending EDHOC message_1 to RS, specifying the access token as External Authorization Data (EAD) in the EAD_1 field of EDHOC message_1 (see Section 3.8 of [I-D.ietf-lake-edhoc]). If the access token is valid and the processing of EDHOC message_1 is successful, RS responds with EDHOC message_2, thus continuing the EDHOC protocol. This alternative cannot be used for the update of access rights.

When running the EDHOC protocol, C uses the authentication credential of RS specified by AS together with the access token, while RS uses the authentication credential of C bound to and specified within the access token. If C and RS complete the EDHOC execution successfully, they are mutually authenticated and they derive an OSCORE Security Context as per Appendix A.1 of [I-D.ietf-lake-edhoc]. Also, RS associates the two used authentication credentials and the completed EDHOC execution with the derived Security Context. The latter is in turn associated with the access token and the access rights of C specified therein.

From then on, C effectively gains authorized and secure access to protected resources on RS, for as long as the access token is valid. Until then, C can communicate with RS by sending a request protected with the established OSCORE Security Context above. The Security

Context is discarded when an access token (whether the same or a different one) is used to successfully derive a new Security Context for C, either by exchanging nonces and using the EDHOC-KeyUpdate function (see Section 5), or by re-running EDHOC. In particular, when supporting this profile, both C and RS MUST support the EDHOC-KeyUpdate function, and they MUST use it instead of re-running EDHOC if the outcome of their previously completed EDHOC execution is still valid.

After the whole procedure has completed and while the access token is valid, C can contact AS to request an update of its access rights, by sending a similar request to the /token endpoint. This request also includes an identifier, which allows AS to find the data it has previously shared with C. This specific identifier, encoded as a byte string, is assigned by AS to a "token series" (see Section 1.1). Upon a successful update of access rights, the new issued access token becomes the latest in its token series. When the latest access token of a token series becomes invalid (e.g., when it expires or gets revoked), that token series ends.

An overview of the profile flow for the "coap_edhoc_oscore" profile is given in Figure 1. The names of messages coincide with those of [RFC9200] when applicable.

```
   C                            RS                       AS
   |                            |                         |
   | <==== Mutual authentication and secure channel ====> |
   |                            |                         |
   | ------- POST /token  ------------------------------> |
   |                            |                         |
   | <-------------------------------- Access Token ----- |
   |                               + Access Information   |
   |                            |                         |
   | ---- POST /authz-info ---> |                         |
   |       (access_token)       |                         |
   |                            |                         |
   | <----- 2.01 Created ------ |                         |
   |                            |                         |
   | <========= EDHOC ========> |                         |
   |  Mutual authentication     |                         |
   |  and derivation of an      |                         |
   |  OSCORE Security Context   |                         |
   |                            |                         |
   |                /Proof-of-possession and              |
   |                Security Context storage/             |
   |                            |                         |
   | ---- OSCORE Request -----> |                         |
   |                            |                         |
   | <--- OSCORE Response ----- |                         |
   |                            |                         |
/Proof-of-possession            |                         |
and Security Context            |                         |
storage (latest)/               |                         |
   |                            |                         |
   | ---- OSCORE Request -----> |                         |
   |                            |                         |
   | <--- OSCORE Response ----- |                         |
   |                            |                         |
   |           ...              |                         |
```

Figure 1: Protocol Overview

3. Client-AS Communication

The following subsections describe the details of the POST request and response to the /token endpoint between C and AS.

In this exchange, AS provides C with the access token, together with a set of parameters that enable C to run EDHOC with RS. In particular, these include information about the authorization credential of RS, AUTH_CRED_RS, transported by value or uniquely referred to.

The access token is securely associated with the authentication credential of C, AUTH_CRED_C, by including it or uniquely referring to it in the access token.

AUTH_CRED_C is specified in the "req_cnf" parameter defined in [RFC9201] of the POST request to the /token endpoint from C to AS, either transported by value or uniquely referred to.

The request to the /token endpoint and the corresponding response can include EDHOC_Information, which is a CBOR map object defined in Section 3.3. This object is transported in the "edhoc_info" parameter registered in Section 10.2 and Section 10.3.

3.1. C-to-AS: POST to /token endpoint

The client-to-AS request is specified in Section 5.8.1 of [RFC9200].

The client must send this POST request to the /token endpoint over a secure channel that guarantees authentication, message integrity and confidentiality (see Section 6).

Editor's note: This formulation overlaps with 3rd para in Section 2, which has normative language. Preferable to keep normative language here.

An example of such a request is shown in Figure 2. In this example, C specifies its own authentication credential by reference, as the hash of an X.509 certificate carried in the "x5t" field of the "req_cnf" parameter. In fact, it is expected that C can typically specify its own authentication credential by reference, since AS is expected to obtain the actual authentication credential during an early client registration process or during a previous secure association establishment with C.

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
{
  "audience" : "tempSensor4711",
  "scope" : "read",
  "req_cnf" : {
    "x5t" : h'822E4879F2A41B510C1F9B'
  }
}
```

Figure 2: Example of C-to-AS POST /token request for an access token.

If C wants to update its access rights without changing an existing OSCORE Security Context, it MUST include EDHOC_Information in its POST request to the /token endpoint. In turn, EDHOC_Information MUST include the "id" field, carrying a CBOR byte string containing the identifier of the token series to which the current, still valid access token shared with RS belongs to. This POST request MUST omit the "req_cnf" parameter.

This identifier is assigned by AS as discussed in Section 3.2, and, together with other information such as audience (see Section 5.8.1 of [RFC9200]), can be used by AS to determine the token series to which the new requested access token has to be added. Therefore, the identifier MUST identify the pair (AUTH_CRED_C, AUTH_CRED_RS) associated with a still valid access token previously issued for C and RS by AS.

AS MUST verify that the received value identifies a token series to which a still valid access token issued for C and RS belongs to. If that is not the case, the Client-to-AS request MUST be declined with the error code "invalid_request" as defined in Section 5.8.3 of [RFC9200].

An example of such a request is shown in Figure 3.

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
{
  "audience" : "tempSensor4711",
  "scope" : "write",
  "edhoc_info" : {
     "id" : h'01'
  }
}
```

Figure 3: Example of C-to-AS POST /token request for updating access rights to an access token.

3.2. AS-to-C: Access Token Response

After verifying the POST request to the /token endpoint and that C is authorized to obtain an access token corresponding to its access token request, AS responds as defined in Section 5.8.2 of [RFC9200]. If the request from C was invalid, or not authorized, AS returns an error response as described in Section 5.8.3 of [RFC9200].

AS can signal that the use of EDHOC and OSCORE as per this profile is REQUIRED for a specific access token, by including the "ace_profile" parameter with the value "coap_edhoc_oscore" in the access token response. This means that C MUST use EDHOC with RS and derive an OSCORE Security Context, as specified in Section 4.3. After that, C MUST use the established OSCORE Security Context to protect communications with RS, when accessing protected resources at RS according to the authorization information indicated in the access token. Usually, it is assumed that constrained devices will be pre-configured with the necessary profile, so that this kind of profile signaling can be omitted.

When issuing any access token of a token series, AS MUST send the following data in the response to C.

* The identifier of the token series to which the issued access token belongs to. This is specified in the "id" field of EDHOC_Information.

  All the access tokens belonging to the same token series are associated with the same identifier, which does not change throughout the series lifetime. A token series ends when the latest issued access token in the series becomes invalid (e.g., when it expires or gets revoked).

  AS assigns an identifier to a token series when issuing the first access token T of that series. When assigning the identifier, AS MUST ensure that this was never used in a previous series of access tokens such that: i) they were issued for the same RS for which the access token T is being issued; and ii) they were bound to the same authentication credential AUTH_CRED_C of the requesting client to which the access token T is being issued (irrespectively of the exact way AUTH_CRED_C is specified in such access tokens).

When issuing the first access token of a token series, AS MUST send the following data in the response to C.

* The authentication credential of RS, namely AUTH_CRED_RS. This is specified in the "rs_cnf" parameter defined in [RFC9201]. AUTH_CRED_RS can be transported by value or referred to by means of an appropriate identifier.

  When issuing the first access token ever to a pair (C, RS) using a pair of corresponding authentication credentials (AUTH_CRED_C, AUTH_CRED_RS), it is typically expected that the response to C specifies AUTH_CRED_RS by value.

When later issuing further access tokens to the same pair (C, RS) using the same AUTH_CRED_RS, it is typically expected that the response to C specifies AUTH_CRED_RS by reference.

When issuing the first access token of a token series, AS MAY send the following data in the response to C. If present, this data MUST be included in the corresponding fields of EDHOC_Information. Some of this information takes advantage of the knowledge that AS may have about C and RS since a previous registration process, with particular reference to what they support as EDHOC peers.

* The EDHOC methods supported by both C and RS (see Section 3.2 of [I-D.ietf-lake-edhoc]). This is specified in the "methods" field of EDHOC_Information.

* The EDHOC cipher suite (see Section 3.6 of [I-D.ietf-lake-edhoc]) to be used by C and RS as selected cipher suite when running EDHOC. This is specified in the "cipher_suites" field of EDHOC_Information. If present, this MUST specify the EDHOC cipher suite which is most preferred by C and at the same time supported by both C and RS.

* Whether RS supports or not EDHOC message_4 (see Section 5.5 of [I-D.ietf-lake-edhoc]). This is specified in the "message_4" field of EDHOC_Information.

* Whether RS supports or not the combined EDHOC + OSCORE request defined in [I-D.ietf-core-oscore-edhoc]. This is specified in the "comb_req" field of EDHOC_Information.

* The path component of the URI of the EDHOC resource at RS, where C is expected to send EDHOC messages as CoAP requests. This is specified in the "uri_path" field of EDHOC_Information. If not specified, the URI path "/.well-known/edhoc" defined in Section 9.7 of [I-D.ietf-lake-edhoc]) is assumed.

* The size in bytes of the OSCORE Master Secret to derive after the EDHOC execution (see Appendix A.1 of [I-D.ietf-lake-edhoc]) and to use for establishing an OSCORE Security Context. This is specified in the "osc_ms_len" field of EDHOC_Information. If not specified, the default value from Appendix A.1 of [I-D.ietf-lake-edhoc] is assumed.

* The size in bytes of the OSCORE Master Salt to derive after the EDHOC execution (see Appendix A.1 of [I-D.ietf-lake-edhoc]) and to use for establishing an OSCORE Security Context. This is specified in the "osc_salt_len" field of EDHOC_Information. If not specified, the default value from Appendix A.1 of [I-D.ietf-lake-edhoc] is assumed.

* The OSCORE version to use (see Section 5.4 of [RFC8613]). This is specified in the "osc_version" field of EDHOC_Information. If specified, AS MUST indicate the highest OSCORE version supported by both C and RS. If not specified, the default value of 1 (see Section 5.4 of [RFC8613]) is assumed.

When issuing any access token of a token series, AS MUST specify the following data in the claims associated with the access token.

* The identifier of the token series, specified in the "id" field of EDHOC_Information, and with the same value specified in the response to C from the /token endpoint.

* The same authentication credential of C that C specified in its POST request to the /token endpoint (see Section 3.1), namely AUTH_CRED_C. If the access token is a CWT, this information MUST be specified in the "cnf" claim.

  In the access token, AUTH_CRED_C can be transported by value or referred to by means of an appropriate identifier, regardless of how C specified it in the request to the /token endpoint. Thus, the specific field carried in the access token claim and specifying AUTH_CRED_C depends on the specific way used by AS.

  When issuing the first access token ever to a pair (C, RS) using a pair of corresponding authentication credentials (AUTH_CRED_C, AUTH_CRED_RS), it is typically expected that AUTH_CRED_C is specified by value.

  When later issuing further access tokens to the same pair (C, RS) using the same AUTH_CRED_C, it is typically expected that AUTH_CRED_C is specified by reference.

When issuing the first access token of a token series, AS MAY specify the following data in the claims associated with the access token. If these data are specified in the response to C from the /token endpoint, they MUST be included in the access token and specify the same values that they have in the response from the /token endpoint.

* The size in bytes of the OSCORE Master Secret to derive after the EDHOC execution and to use for establishing an OSCORE Security Context. If it is included, it is specified in the "osc_ms_len" field of EDHOC_Information, and it has the same value that the "osc_ms_len" field has in the response to C. If it is not included, the default value from Appendix A.1 of [I-D.ietf-lake-edhoc] is assumed.

* The size in bytes of the OSCORE Master Salt to derive after the EDHOC execution (see Appendix A.1 of [I-D.ietf-lake-edhoc]) and

to use for establishing an OSCORE Security Context. If it is included, it is specified in the "osc_salt_len" field of EDHOC_Information, and it has the same value that the "osc_salt_len" field has in the response to C. If it is not included, the default value from Appendix A.1 of [I-D.ietf-lake-edhoc] is assumed.

* The OSCORE version to use (see Section 5.4 of [RFC8613]). This is specified in the "osc_version" field of the "edhoc_info" parameter. If it is included, it is specified in the "osc_version" field of EDHOC_Information, and it has the same value that the "osc_version" field has in the response to C. If it is not included, the default value of 1 (see Section 5.4 of [RFC8613]) is assumed.

When issuing the first access token of a token series, AS can take either of the two possible options.

* AS provides the access token to C, by specifying it in the "access_token" parameter of the access token response. In such a case, the access token response MAY include the parameter "token_uploaded", which MUST encode the CBOR simple value "false" (0xf4).

* AS does not provide the access token to C. Rather, AS uploads the access token to the /authz-info endpoint at RS, exactly like C would do, and as defined in Section 4.1 and Section 4.2. Then, when replying to C with the access token response as defined above, the response MUST NOT include the parameter "access_token", and MUST include the parameter "token_uploaded" encoding the CBOR simple value "true" (0xf5). This is shown by the example in Appendix A.3.

  Note that, in case C and RS have already completed an EDHOC execution leveraging a previous access token series, using this approach implies that C and RS have to re-run the EDHOC protocol. That is, they cannot more efficiently make use of the EDHOC-KeyUpdate function, as defined in Section 5, see Section 4.

  Also note that this approach is not applicable when issuing access tokens following the first one in the same token series, i.e., when updating access rights.

When CWTs are used as access tokens, EDHOC_Information MUST be transported in the "edhoc_info" claim, defined in Section 10.5.

Since the access token does not contain secret information, only its integrity and source authentication are strictly necessary to ensure. Therefore, AS can protect the access token with either of the means discussed in Section 6.1 of [RFC9200]. Nevertheless, when using this profile, it is RECOMMENDED that the access token is a CBOR web token (CWT) protected with COSE_Encrypt/COSE_Encrypt0 as specified in [RFC8392].

Figure 4 shows an example of an AS response. The "rs_cnf" parameter specifies the authentication credential of RS, as an X.509 certificate transported by value in the "x5chain" field. The access token and the authentication credential of RS have been truncated for readability.

```
Header: Created (Code=2.01)
   Content-Type: "application/ace+cbor"
   Payload:
   {
     "access_token" : h'8343a1010aa2044c53 ...
      (remainder of access token (CWT) omitted for brevity)',
     "ace_profile" : "coap_edhoc_oscore",
     "expires_in" : "3600",
     "rs_cnf" : {
       "x5chain" : h'3081ee3081a1a00302 ...'
       (remainder of the access credential omitted for brevity)'
     }
     "edhoc_info" : {
       "id" : h'01',
       "methods" : [0, 1, 2, 3],
       "cipher_suites": 0
     }
   }
```

Figure 4: Example of AS-to-C Access Token response with EDHOC and OSCORE profile.

Figure 5 shows an example CWT Claims Set, including the relevant EDHOC parameters in the "edhoc_info" claim. The "cnf" claim specifies the authentication credential of C, as an X.509 certificate transported by value in the "x5chain" field. The authentication credential of C has been truncated for readability.

```
{
 "aud" : "tempSensorInLivingRoom",
 "iat" : "1360189224",
 "exp" : "1360289224",
 "scope" :  "temperature_g firmware_p",
 "cnf" : {
   "x5chain" : h'3081ee3081a1a00302 ...'
 }
 "edhoc_info" : {
   "id" : h'01',
   "methods" : [0, 1, 2, 3],
   "cipher_suites": 0
 }
}
```

Figure 5: Example of CWT Claims Set with EDHOC parameters.

If C has requested an update to its access rights using the same OSCORE Security Context, which is valid and authorized, then:

* The response MUST NOT include the "rs_cnf" parameter.

* The EDHOC_Information in the response MUST include only the "id" field, specifying the identifier of the token series.

* The EDHOC_Information in the access token MUST include only the "id" field, specifying the identifier of the token series. In particular, if the access token is a CWT, the "edhoc_info" claim MUST include only the "id" field.

This identifier of the token series needs to be included in the new access token in order for RS to identify the old access token to supersede, as well as the OSCORE Security Context already shared between C and RS and to be associated with the new access token.

3.3. The EDHOC_Information

An EDHOC_Information is an object including information that guides two peers towards executing the EDHOC protocol. In particular, the EDHOC_Information is defined to be serialized and transported between nodes, as specified by this document, but it can also be used by other specifications if needed.

The EDHOC_Information can either be encoded as a JSON object or as a CBOR map. The set of common fields that can appear in an EDHOC_Information can be found in the IANA "EDHOC Information" registry (see Section 10.9), defined for extensibility, and the initial set of parameters defined in this document is specified below. All parameters are optional.

Figure 6 provides a summary of the EDHOC_Information parameters
defined in this section.

Name	CBOR Type	CBOR value	Registry	Description
id	0	bstr		Identifier of EDHOC execution
methods	1	int / array of int	EDHOC Method Type Registry	Set of supported EDHOC methods
cipher_suites	2	int / array of int	EDHOC Cipher Suites Registry	Set of supported EDHOC cipher suites
key_update	3	simple value "true" / simple value "false"		Support for the EDHOC-KeyUpdate function
message_4	4	simple value "true" / simple value "false"		Support for EDHOC message_4
comb_req	5	simple value "true" / simple value "false"		Support for the EDHOC + OSCORE combined request
uri_path	6	tstr		URI-path of the EDHOC resource
osc_ms_len	7	uint		Length in bytes of the OSCORE Master Secret to derive
osc_salt_len	8	uint		Length in bytes of the OSCORE Master Salt to derive
osc_version	9	uint		OSCORE version number to use

Figure 6: EDHOC_Information Parameters

*id: This parameter identifies an EDHOC execution and is encoded as a byte string. In JSON, the "id" value is a Base64 encoded byte string. In CBOR, the "id" type is a byte string, and has label 0.

*methods: This parameter specifies a set of supported EDHOC methods (see Section 3.2 of [I-D.ietf-lake-edhoc]). If the set is composed of a single EDHOC method, this is encoded as an integer. Otherwise, the set is encoded as an array of integers, where each array element encodes one EDHOC method. In JSON, the "methods" value is an integer or an array of integers. In CBOR, the "methods" is an integer or an array of integers, and has label 1.

*cipher_suites: This parameter specifies a set of supported EDHOC cipher suites (see Section 3.6 of [I-D.ietf-lake-edhoc]). If the set is composed of a single EDHOC cipher suite, this is encoded as an integer. Otherwise, the set is encoded as an array of integers, where each array element encodes one EDHOC cipher suite. In JSON, the "cipher_suites" value is an integer or an array of integers. In CBOR, the "cipher_suites" is an integer or an array of integers, and has label 2.

*key_update: This parameter indicates whether the EDHOC-KeyUpdate function (see Appendix I of [I-D.ietf-lake-edhoc]) is supported. In JSON, the "key_update" value is a boolean. In CBOR, "key_update" is the simple value "true" or "false", and has label 3.

*message_4: This parameter indicates whether the EDHOC message_4 (see Section 5.5 of [I-D.ietf-lake-edhoc]) is supported. In JSON, the "message_4" value is a boolean. In CBOR, "message_4" is the simple value "true" or "false", and has label 4.

*comb_req: This parameter indicates whether the combined EDHOC + OSCORE request defined in [I-D.ietf-core-oscore-edhoc]) is supported. In JSON, the "comb_req" value is a boolean. In CBOR, "comb_req" is the simple value "true" or "false", and has label 5.

*uri_path: This parameter specifies the path component of the URI of the EDHOC resource where EDHOC messages have to be sent as requests. In JSON, the "uri_path" value is a string. In CBOR, "uri_path" is text string, and has label 6.

*osc_ms_len: This parameter specifies the size in bytes of the OSCORE Master Secret to derive after the EDHOC execution, as per Appendix A.1 of [I-D.ietf-lake-edhoc]. In JSON, the "osc_ms_len" value is an integer. In CBOR, the "osc_ms_len" type is unsigned integer, and has label 7.

* osc_salt_len: This parameter specifies the size in bytes of the OSCORE Master Salt to derive after the EDHOC execution, as per Appendix A.1 of [I-D.ietf-lake-edhoc]. In JSON, the "osc_salt_len" value is an integer. In CBOR, the "osc_salt_len" type is unsigned integer, and has label 8.

* osc_version: This parameter specifies the OSCORE Version number that the two EDHOC peers have to use when using OSCORE. For more information about this parameter, see Section 5.4 of [RFC8613]. In JSON, the "osc_version" value is an integer. In CBOR, the "osc_version" type is unsigned integer, and has label 9.

An example of JSON EDHOC_Information is given in Figure 7.

```
   "edhoc_info" : {
       "id" : b64'AQ==',
       "methods" : 1,
       "cipher_suites" : 0
   }
```

Figure 7: Example of JSON EDHOC_Information

The CDDL grammar describing the CBOR EDHOC_Information is:

```
EDHOC_Information = {
   ? 0 => bstr,                ; id
   ? 1 => int / array,         ; methods
   ? 2 => int / array,         ; cipher_suites
   ? 3 => true / false,        ; key_update
   ? 4 => true / false,        ; message_4
   ? 5 => true / false,        ; comb_req
   ? 6 => tstr,                ; uri_path
   ? 7 => uint,                ; osc_ms_len
   ? 8 => uint,                ; osc_salt_len
   ? 9 => uint,                ; osc_version
   * int / tstr => any
}
```

4. Client-RS Communication

The following subsections describe the exchanges between C and RS, which comprise the token uploading to RS, and the execution of the EDHOC protocol. Note that, as defined in Section 3.2, AS may not have provided C with the access token, and have rather uploaded the access token to the /authz-info endpoint at RS on behalf of C.

In order to upload the access token to RS, C can send a POST request to the /authz-info endpoint at RS. This is detailed in Section 4.1 and Section 4.2, and it is shown by the example in Appendix A.1.

Alternatively, C can upload the access token while executing the EDHOC protocol, by transporting the access token in the EAD_1 field of the first EDHOC message sent to RS. This is further discussed in Section 4.3, and it is shown by the example in Appendix A.2.

In either case, following the uploading of the access token, C and RS run the EDHOC protocol to completion, by exchanging POST requests and related responses to a dedicated EDHOC resource at RS (see Section 4.3). Once completed the EDHOC execution, C and RS have agreed on a common secret key PRK_out (see Section 4.1.3 of [I-D.ietf-lake-edhoc]), from which they establish an OSCORE Security Context (see Section 4.3). After that, C and RS use the established OSCORE Security Context to protect their communications when accessing protected resources at RS, as per the access rights specified in the access token (see Section 4.4).

Note that, by means of the respective authentication credentials, C and RS are mutually authenticated once they have successfully completed the execution of the EDHOC protocol.

As to proof-of-possession, RS always gains knowledge that C has PRK_out at the end of the successful EDHOC execution. Conversely, C gains knowledge that RS has PRK_out either when receiving and successfully verifying the optional EDHOC message_4 from RS, or when successfully verifying a response from RS protected with the generated OSCORE Security Context.

4.1. C-to-RS: POST to /authz-info endpoint

The access token can be uploaded to RS by using the /authz-info endpoint at RS. To this end, C MUST use CoAP [RFC7252] and the Authorization Information endpoint described in Section 5.10.1 of [RFC9200] in order to transport the access token.

That is, C sends a POST request to the /authz-info endpoint at RS, with the request payload conveying the access token without any CBOR wrapping. As per Section 5.10.1 of [RFC9200], the Content-Format of the POST request has to reflect the format of the transported access token. In particular, if the access token is a CWT, the content-format MUST be "application/cwt".

The communication with the /authz-info endpoint does not have to be protected, except for the update of access rights case described below.

In case of initial access token provisioning to RS for this Client, or in other cases without a valid EDHOC session, the uploading of the access token is followed by the execution of the EDHOC protocol (or combined using EAD as described in Section 4.3) and by the derivation of an OSCORE Security Context, as detailed later in this section.

In the following, we outline some alternative processing, which occur when the provisioned access token or the established OSCORE Security Context for various reasons is no longer available or sufficient. In the cases below, it is assumed that the EDHOC session is still valid, otherwise the processing essentially falls back to the case discussed above.

1. C may be required to re-POST the access token, since RS may have deleted a stored access token (and associated OSCORE Security Context) at any time, for example due to all storage space being consumed. This situation can be detected by C when it receives a 4.01 (Unauthorized) response from RS, especially as an "AS Request Creation Hints" message (see Section 5.3 of [RFC9200].

2. C may be posting the first access token in a new series, e.g., because the old access token expired and thus its series terminated.

3. C may need to update the OSCORE Security Context, e.g., due to a policy limiting its use in terms of time or amount of processed data, or to the imminent exhaustion of the OSCORE Sender Sequence Number space. The OSCORE Security Context can be updated by:

   a. using the KUDOS key update protocol specified in [I-D.ietf-core-oscore-key-update], if supported by both C and RS; or

   b. re-posting the access token using the same procedure as in case 1 above.

In cases 1, 2 and 3b, C and RS rely on a protocol similar to the coap_oscore profile [RFC9203], but leveraging the EDHOC-KeyUpdate function (see Section 5) before deriving a new OSCORE Security Context.

Case 3a provides a lightweight alternative independent of ACE, and does not require the posting of an access token.

In either case, C and RS establish a new OSCORE Security Context that replaces the old one and will be used for protecting their communications from then on. In particular, RS MUST associate the new OSCORE Security Context with the current (potentially re-posted) access token. Note that, unless C and RS re-run the EDHOC protocol, they preserve their same OSCORE identifiers, i.e., their OSCORE Sender/Recipient IDs.

If C has already posted a valid access token, has already established an OSCORE Security Context with RS, and wants to update its access rights, then C can do so by posting a new access token to the /authz-info endpoint. The new access token contains the updated access rights for C to access protected resources at RS, and C has to obtain it from AS as a new access token in the same token series of the current one (see Section 3.1 and Section 3.2). When posting the new access token to the /authz-info endpoint, C MUST protect the POST request using the current OSCORE Security Context shared with RS. After successful verification (see Section 4.2), RS will replace the old access token with the new one, while preserving the same OSCORE Security Context. In particular, C and RS do not re-run the EDHOC protocol and they do not establish a new OSCORE Security Context.

4.2. RS-to-C: 2.01 (Created)

Upon receiving an access token from C, RS MUST follow the procedures defined in Section 5.10.1 of [RFC9200]. That is, RS must verify the validity of the access token. RS may make an introspection request (see Section 5.9.1 of [RFC9200]) to validate the access token.

If the access token is valid, RS proceeds as follows.

RS checks whether it is already storing the authentication credential of C, namely AUTH_CRED_C, specified as PoP-key in the access token by value or reference. In such a case, RS stores the access token and MUST reply to the POST request with a 2.01 (Created) response.

Otherwise, RS retrieves AUTH_CRED_C, e.g., from the access token if the authentication credential is specified therein by value, or from a further trusted source pointed to by the AUTH_CRED_C identifier included in the access token. After that, RS validates the actual AUTH_CRED_C. In case of successful validation, RS stores AUTH_CRED_C as a valid authentication credential. Then, RS stores the access token and MUST reply to the POST request with a 2.01 (Created) response.

If RS does not find an already stored AUTH_CRED_C, or fails to retrieve it or to validate it, then RS MUST respond with an error response code equivalent to the CoAP code 4.00 (Bad Request). RS may provide additional information in the payload of the error response, in order to clarify what went wrong.

Instead, if the access token is valid but it is associated with claims that RS cannot process (e.g., an unknown scope), or if any of the expected parameters is missing (e.g., any of the mandatory parameters from AS or the identifier "id"), or if any parameters received in the EDHOC_Information is unrecognized, then RS MUST respond with an error response code equivalent to the CoAP code 4.00 (Bad Request). In the latter two cases, RS may provide additional information in the payload of the error response, in order to clarify what went wrong.

When an access token becomes invalid (e.g., due to its expiration or revocation), RS MUST delete the access token and the associated OSCORE Security Context, and MUST notify C with an error response with code 4.01 (Unauthorized) for any long running request, as specified in Section 5.8.3 of [RFC9200].

If RS receives an access token in an OSCORE protected request, it means that C is requesting an update of access rights. In such a case, RS MUST check that both the following conditions hold.

* RS checks whether it stores an access token T_OLD, such that the "id" field of EDHOC_Identifier matches the "id" field of EDHOC_Identifier in the new access token T_NEW.

* RS checks whether the OSCORE Security Context CTX used to protect the request matches the OSCORE Security Context associated with the stored access token T_OLD.

If both the conditions above hold, RS MUST replace the old access token T_OLD with the new access token T_NEW, and associate T_NEW with the OSCORE Security Context CTX. Then, RS MUST respond with a 2.01 (Created) response protected with the same OSCORE Security Context, with no payload.

Otherwise, RS MUST respond with a 4.01 (Unauthorized) error response. RS may provide additional information in the payload of the error response, in order to clarify what went wrong.

As specified in Section 5.10.1 of [RFC9200], when receiving an updated access token with updated authorization information from C (see Section 4.1), it is recommended that RS overwrites the previous access token. That is, only the latest authorization information in the access token received by RS is valid. This simplifies the process needed by RS to keep track of authorization information for a given client.

4.3. EDHOC Execution and Setup of OSCORE Security Context

In order to mutually authenticate and establish a long-term secret key PRK_out with forward secrecy, C and RS run the EDHOC protocol

[I-D.ietf-lake-edhoc]. In particular, C acts as EDHOC Initiator thus sending EDHOC message_1, while RS acts as EDHOC Responder.

As per Appendix A.2 of [I-D.ietf-lake-edhoc], C sends EDHOC message_1 and EDHOC message_3 to an EDHOC resource at RS, as CoAP POST requests. Also RS sends EDHOC message_2 and (optionally) EDHOC message_4 as 2.04 (Changed) CoAP responses. If, in the access token response received from AS (see Section 3.1), the "uri_path" field of the EDHOC_Information was included, then C MUST target the EDHOC resource at RS with the URI path specified in the "uri_path" field.

In order to seamlessly run EDHOC, a client does not have to first upload to RS an access token whose scope explicitly indicates authorized access to the EDHOC resource. At the same time, RS has to ensure that attackers cannot perform requests on the EDHOC resource, other than sending EDHOC messages. Specifically, it SHOULD NOT be possible to perform anything else than POST on an EDHOC resource.

When preparing EDHOC message_1, C performs the following steps, in additions to those defined in Section 5.2.1 of [I-D.ietf-lake-edhoc].

* If, in the access token response received from AS (see Section 3.1), the "methods" field of the EDHOC_Information was included, then C MUST specify one of those EDHOC methods in the METHOD field of EDHOC message_1. That is, one of the EDHOC methods specified in the "methods" field of EDHOC_Information MUST be the EDHOC method used when running EDHOC with RS.

* If, in the access token response received from AS (see Section 3.1), the "cipher_suites" field of the EDHOC_Information was included, then C MUST specify the EDHOC cipher suite therein in the SUITES_I field of EDHOC message_1. That is, the EDHOC cipher suite specified in the "cipher_suites" field of EDHOC_Information MUST be the selected cipher suite when running EDHOC with RS.

* Rather than first uploading the access token to the /authz-info endpoint at RS as described in Section 4.1, C MAY include the access token in the EAD_1 field of EDHOC message_1 (see Section 3.8 of [I-D.ietf-lake-edhoc]). This is shown by the example in Appendix A.2.

  In such a case, as per Section 3.8 of [I-D.ietf-lake-edhoc], C adds the EAD item EAD_ACCESS_TOKEN = (ead_label, ead_value) to the EAD_1 field. In particular, ead_label is the integer value TBD registered in Section 10.8 of this document, while ead_value is a CBOR byte string with value the access token. That is, the

CBOR byte string is equal to the value of the "access_token" field of the access token response from AS (see Section 3.2).

If EDHOC message_1 includes the EAD item EAD_ACCESS_TOKEN within the field EAD_1, then RS MUST process the access token carried out in ead_value as specified in Section 4.2. If such a process fails, RS MUST reply to C with an EDHOC error message with ERR_CODE 1 (see Section 6 of [I-D.ietf-lake-edhoc]), and it MUST discontinue the EDHOC protocol. RS MUST have successfully completed the processing of the access token before continuing the EDHOC execution by sending EDHOC message_2.

Note that the EAD_1 field of EDHOC message_1 cannot carry an access token for the update of access rights, but rather only an access token issued as the first of a token series.

In EDHOC message_2, the authentication credential CRED_R indicated by the message field ID_CRED_R is the authentication credential of RS, namely AUTH_CRED_RS, that C obtained from AS. The processing of EDHOC message_2 is defined in detail in Section 5.3 of [I-D.ietf-lake-edhoc].

In EDHOC message_3, the authentication credential CRED_I indicated by the message field ID_CRED_I is the authentication credential of C, namely AUTH_CRED_C, i.e., the PoP key bound to the access token and specified therein. The processing of EDHOC message_3 is defined in detail in Section 5.4 of [I-D.ietf-lake-edhoc].

Once successfully completed the EDHOC execution, C and RS have both derived the long-term secret key PRK_out (see Section 4.1.3 of [I-D.ietf-lake-edhoc]), from which they both derive the key PRK_Exporter (see Section 4.2.1 of [I-D.ietf-lake-edhoc]). Then, C and RS derive an OSCORE Security Context, as defined in Appendix A.1 of [I-D.ietf-lake-edhoc]. In addition, the following applies.

* If, in the access token response received from AS (see Section 3.1) and in the access token, the "osc_ms_size" field of the EDHOC_Information was included, then C and RS MUST use the value specified in the "osc_ms_size" field as length in bytes of the OSCORE Master Secret. That is, the value of the "osc_ms_size" field MUST be used as value for the oscore_key_length parameter of the EDHOC-Exporter function when deriving the OSCORE Master Secret (see Appendix A.1 of [I-D.ietf-lake-edhoc]).

* If, in the access token response received from AS (see Section 3.1) and in the access token, the "osc_salt_size" field of the EDHOC_Information was included, then C and RS MUST use the value specified in the "osc_salt_size" field as length in bytes of the OSCORE Master Salt. That is, the value of the

"osc_salt_size" field MUST be used as value for the oscore_salt_length parameter of the EDHOC-Exporter function when deriving the OSCORE Master Salt (see Appendix A.1 of [I-D.ietf-lake-edhoc]).

* If, in the access token response received from AS (see Section 3.1) and in the access token, the "osc_version" field of the EDHOC_Information was included, then C and RS MUST derive the OSCORE Security Context, and later use it to protect their communications, consistently with the OSCORE version specified in the "osc_version" field.

* Given AUTH_CRED_C the authentication credential of C used as CRED_I in the completed EDHOC execution, RS associates the derived OSCORE Security Context with the stored access token bound to AUTH_CRED_C as PoP-key (regardless of whether AUTH_CRED_C is specified by value or by reference in the access token claims).

If C supports it, C MAY use the EDHOC + OSCORE combined request defined in [I-D.ietf-core-oscore-edhoc], as also shown by the example in Appendix A.2. In such a case, both EDHOC message_3 and the first OSCORE-protected application request to a protected resource are sent to RS as combined together in a single OSCORE-protected CoAP request, thus saving one round trip. This requires C to derive the OSCORE Security Context with RS already after having successfully processed the received EDHOC message_2. If, in the access token response received from AS (see Section 3.1), the "comb_req" field of the EDHOC_Information was included and specified the CBOR simple value "false" (0xf4), then C MUST NOT use the EDHOC + OSCORE combined request with RS.

4.4. Access Rights Verification

RS MUST follow the procedures defined in Section 5.10.2 of [RFC9200]. That is, if RS receives an OSCORE-protected request targeting a protected resource from C, then RS processes the request according to [RFC8613], when Version 1 of OSCORE is used. Future specifications may define new versions of OSCORE, that AS can indicate C and RS to use by means of the "osc_version" field of EDHOC_Information (see Section 3).

If OSCORE verification succeeds and the target resource requires authorization, RS retrieves the authorization information using the access token associated with the OSCORE Security Context. Then, RS must verify that the authorization information covers the target resource and the action intended by C on it.

5. Use of EDHOC-KeyUpdate

Once successfully completed an EDHOC execution, C and RS are expected to preserve the EDHOC state of such an execution, as long as the authentication credentials of both C and RS, namely AUTH_CRED_C and AUTH_CRED_RS are valid. This especially consists in preserving the secret key PRK_out attained at the end of the EDHOC execution.

In case C has to establish a new OSCORE Security Context with RS, and as long as the outcome of their previously completed EDHOC execution is still valid, C and RS MUST rely on the EDHOC-KeyUpdate function defined in Appendix I of [I-D.ietf-lake-edhoc] as further specified in the rest of this section, rather than re-running the EDHOC protocol. When supporting this profile, both C and RS MUST support the EDHOC-KeyUpdate function. The procedure is sketched in Figure 8.

```
   C                            RS
   |                            |
   |                            |
   | ---- POST /authz-info ---> |
   |     (access_token, N1)     |
   |                            |
   | <--- 2.01 Created (N2) --- |
   |                            |

  /Apply EDHOC-KeyUpdate with
   concatenated nonces as input,
   derive OSCORE Security Context/

   |                            |
   | ---- OSCORE Request -----> |
   |                            |
   |        /Proof-of-possession and
   |         Security Context storage/
   |                            |
   | <--- OSCORE Response ----- |
   |                            |
/Proof-of-possession and        |
 Security Context storage/      |
   |                            |
   | ---- OSCORE Request -----> |
   |                            |
   | <--- OSCORE Response ----- |
   |                            |
   |           ...              |
```

Figure 8: Updated OSCORE Security Context using EDHOC-KeyUpdate

Establishing a new OSCORE Security Context by leveraging the EDHOC-KeyUpdate function is possible in the following cases.

* C has to upload to RS the newly obtained, first access token of a new token series, as an unprotected POST request to the /authz-info endpoint at RS. This is the case after the latest access token of the previous token series has become invalid (e.g., it expired or got revoked), and thus RS has deleted it together with the associated OSCORE Security Context (see Section 7).

* C re-uploads to RS the current access token shared with RS, i.e., the latest access token in the current token series, as an unprotected POST request to the /authz-info endpoint at RS. Like in Section 4.1, this is the case when C simply wants to replace the current OSCORE Security Context with a new one, and associate it with the same current, re-uploaded access token.

In either case, C and RS have to establish a new OSCORE Security Context and to associate it with the (re-)uploaded access token.

When using this approach, C and RS perform the following actions.

C MUST generate a nonce value N1 very unlikely to have been used with the same pair of authentication credentials (AUTH_CRED_C, AUTH_CRED_RS). When using this profile, it is RECOMMENDED to use a 64-bit long random number as the nonce's value. C MUST store the nonce N1 as long as the response from RS is not received and the access token related to it is still valid (to the best of C's knowledge).

C MUST use CoAP [RFC7252] and the Authorization Information resource as described in Section 5.10.1 of [RFC9200] to transport the access token and N1 to RS.

Note that, unlike what is defined in Section 4.1, the use of the payload and of the Content-Format is different from what is described in Section 5.10.1 of [RFC9200], where only the access token is transported and without any CBOR wrapping. That is, C MUST wrap the access token and N1 in a CBOR map, and MUST use the Content-Format "application/ace+cbor" defined in Section 8.16 of [RFC9200]. The CBOR map MUST specify the access token using the "access_token" parameter, and N1 using the "nonce1" parameter defined in Section 4.1.1 of [RFC9203].

The communication with the /authz-info endpoint at RS MUST NOT be protected. This approach is not applicable when C uploads to RS for the first time an access token to update access rights (which rather requires protected communication and does not result in deriving a new OSCORE Security Context).

Figure 9 shows an example of the POST request sent by C to RS. The access token has been truncated for readability.

```
Header: POST (Code=0.02)
Uri-Host: "rs.example.com"
Uri-Path: "authz-info"
Content-Format: "application/ace+cbor"
Payload:
  {
    "access_token": h'8343a1010aa2044c53 ...
     (remainder of access token (CWT) omitted for brevity)',
    "nonce1": h'018a278f7faab55a'
  }
```

Figure 9: Example of C-to-RS POST /authz-info request using CWT and EDHOC-KeyUpdate

Upon receiving the POST request from C, RS MUST follow the procedures defined in Section 5.10.1 of [RFC9200]. That is, RS must verify the validity of the access token. RS may make an introspection request (see Section 5.9.1 of [RFC9200]) to validate the access token.

If the access token is valid, RS proceeds as follows.

RS checks whether it is already storing the authentication credential of C, namely AUTH_CRED_C, specified as PoP-key in the access token by value or reference.

If RS does not find AUTH_CRED_C among the stored authentication credentials, RS retrieves AUTH_CRED_C, e.g., from the access token if the authentication credential is specified therein by value, or from a further trusted source pointed to by the AUTH_CRED_C identifier included in the access token. After that, RS validates the actual AUTH_CRED_C. In case of successful validation, RS stores AUTH_CRED_C as a valid authentication credential.

If RS does not find an already stored AUTH_CRED_C, or fails to retrieve it or to validate it, then RS MUST respond with an error response code equivalent to the CoAP code 4.00 (Bad Request). RS may provide additional information in the payload of the error response, in order to clarify what went wrong.

If the access token is valid but it is associated with claims that RS cannot process (e.g., an unknown scope), or if any of the expected parameters is missing (e.g., any of the mandatory parameters from AS or the identifier "id"), or if any parameters received in the EDHOC_Information is unrecognized, then RS MUST respond with an error response code equivalent to the CoAP code 4.00 (Bad Request). In the latter two cases, RS may provide additional information in the payload of the error response, in order to clarify what went wrong.

If RS does not find the stored state of a completed EDHOC execution where the authentication credential AUTH_CRED_C was used as CRED_I, then RS MUST respond with an error response code equivalent to the CoAP code 4.00 (Bad Request). RS may provide additional information in the payload of the error response, in order to clarify what went wrong.

Otherwise, if all the steps above are successful, RS stores the access token and MUST generate a nonce N2 very unlikely to have been previously used with the same pair of authentication credentials (AUTH_CRED_C, AUTH_CRED_RS). When using this profile, it is RECOMMENDED to use a 64-bit long random number as the nonce's value. Then, RS MUST reply to the POST request with a 2.01 (Created) response, for which RS MUST use the Content-Format "application/ace+cbor" defined in Section 8.16 of [RFC9200]. The payload of the response MUST be a CBOR map, which MUST specify N2 using the "nonce2" parameter defined in Section 4.2.1 of [RFC9203].

Figure 10 shows an example of the response sent by RS to C.

```
Header: Created (Code=2.01)
Content-Format: "application/ace+cbor"
Payload:
  {
    "nonce2": h'25a8991cd700ac01'
  }
```

Figure 10: Example of RS-to-C 2.01 (Created) response using EDHOC-KeyUpdate

Once they have exchanged N1 and N2, both C and RS build a CBOR byte string EXTENDED_NONCE as follows.

First, RAW_STRING is prepared as the concatenation of N1 and N2, in this order: RAW_STRING = N1 | N2, where | denotes byte string concatenation, and N1 and N2 are the two nonces encoded as CBOR byte strings. Then, the resulting RAW_STRING is wrapped into the CBOR byte string EXTENDED_NONCE.

An example is given in Figure 11, with reference to the values of N1 and N2 shown in Figure 9 and Figure 10.

```
N1 and N2 expressed in CBOR diagnostic notation
N1 = h'018a278f7faab55a'
N2 = h'25a8991cd700ac01'

N1 and N2 as CBOR encoded byte strings
N1 = 0x48018a278f7faab55a
N2 = 0x4825a8991cd700ac01

RAW_STRING = 0x48 018a278f7faab55a 48 25a8991cd700ac01

EXTENDED_NONCE expressed in CBOR diagnostic notation
EXTENDED_NONCE = h'48018a278f7faab55a4825a8991cd700ac01'

EXTENDED_NONCE as CBOR encoded byte string
EXTENDED_NONCE = 0x52 48018a278f7faab55a4825a8991cd700ac01
```

Figure 11: Example of EXTENDED_NONCE construction, with N1 and N2 encoded in CBOR

If JSON is used instead of CBOR, then RAW_STRING is the Base64 encoding of the concatenation of the same parameters, each of them prefixed by their size encoded in 1 byte. When using JSON, the nonces have a maximum size of 255 bytes. An example is given in Figure 12, where the nonces and RAW_STRING are encoded in Base64.

```
N1 and N2 values
N1 = 0x018a278f7faab55a (8 bytes)
N2 = 0x25a8991cd700ac01 (8 bytes)

Input to Base64 encoding:
0x08 018a278f7faab55a 08 25a8991cd700ac01

RAW_STRING = b64'CAGKJ49/qrVaCCWomRzXAKwB'

EXTENDED_NONCE expressed in CBOR diagnostic notation
EXTENDED_NONCE = h'08018a278f7faab55a0825a8991cd700ac01'

EXTENDED_NONCE as CBOR encoded byte string
EXTENDED_NONCE = 0x52 08018a278f7faab55a0825a8991cd700ac01
```

Figure 12: Example of EXTENDED_NONCE construction, with N1 and N2 encoded in JSON

Once computed the CBOR byte string EXTENDED_NONCE, both C and RS perform the following steps.

1. C refers to the stored state of the completed EDHOC execution where: i) the authentication credential AUTH_CRED_C was used as CRED_I; and ii) the authentication credential that was used as

CRED_R is the AUTH_CRED_RS that AS indicated in the access token response (see Section 3.2), when providing C with the first access token of the token series comprising the access token just uploaded to RS.

RS refers to the stored state of a completed EDHOC execution where the authentication credential AUTH_CRED_C was used as CRED_I. In case of multiple matching EDHOC executions, RS considers the state of the EDHOC execution that, among the matching ones, has completed latest.

2. With reference to the EDHOC state determined at the previous step, C and RS invoke the EDHOC-KeyUpdate function (see Appendix I of [I-D.ietf-lake-edhoc]), specifying the CBOR byte string EXTENDED_NONCE as "context" argument. This results in updating the secret key PRK_out to be considered from here on for this EDHOC state.

3. With reference to the same EDHOC state as above, C and RS update the secret key PRK_Exporter as per Section 4.2.1 of [I-D.ietf-lake-edhoc]. In particular, the key PRK_out derived at step 2 is specified as "PRK_out" argument. This results in updating the secret key PRK_Exporter to be considered from here on for this EDHOC state.

4. C and RS establish a new OSCORE Security Context as defined in Section 4.3, just like if they had completed an EDHOC execution. Note that, since C and RS have not re-run the EDHOC protocol, they preserve their same OSCORE identifiers, i.e., their OSCORE Sender/Recipient IDs.

5. RS associates the posted access token with the OSCORE Security Context established at step 4. In case C has in fact re-posted a still valid access token, RS also discards the old OSCORE Security Context previously associated with that access token.

6. Hereafter, C and RS use the OSCORE Security Context established at step 4 to protect their communications.

6. Secure Communication with AS

As specified in the ACE framework (see Sections 5.8 and 5.9 of [RFC9200]), the requesting entity (RS and/or C) and AS communicates via the /token or /introspect endpoint. When using this profile, the use of CoAP [RFC7252] and OSCORE [RFC8613] for this communication is RECOMMENDED. Other protocols fulfilling the security requirements defined in Section 5 of [RFC9200] (such as HTTP and DTLS or TLS) MAY be used instead.

If OSCORE is used, the requesting entity and AS need to have a OSCORE Security Context in place. While this can be pre-installed, the requesting entity and AS can establish such an OSCORE Security Context, for example, by running the EDHOC protocol, as shown between C and AS by the examples in Appendix A.1, Appendix A.2 and Appendix A.3. The requesting entity and AS communicate through the /token endpoint as specified in Section 5.8 of [RFC9200] and through the /introspect endpoint as specified in Section 5.9 of [RFC9200].

Furthermore, as defined in Section 3.2 and shown by the example in Appendix A.3, AS may upload the access token to the /authz-info endpoint at RS, on behalf of C. In such a case, that exchange between AS and RS is not protected, just like when C uploads the access token to RS by itself.

7. Discarding the Security Context

There are a number of cases where C or RS have to discard the OSCORE Security Context, and possibly establish a new one.

C MUST discard the current OSCORE Security Context shared with RS when any of the following occurs.

* The OSCORE Sender Sequence Number space of C gets exhausted.

* The access token associated with the OSCORE Security Context becomes invalid, for example due to expiration or revocation.

* C receives a number of 4.01 (Unauthorized) responses to OSCORE-protected requests sent to RS and protected using the same OSCORE Security Context. The exact number of such received responses needs to be specified by the application.

* C receives a nonce N2 in the 2.01 (Created) response to an unprotected POST request to the /authz-info endpoint at RS, when re-posting a still valid access token associated with the existing OSCORE Security context together with a nonce N1, in order to trigger the use of the EDHOC-KeyUpdate function (see Section 5).

* The authentication credential of C (of RS) becomes invalid (e.g., due to expiration or revocation), and it was used as CRED_I (CRED_R) in the EDHOC execution whose PRK_out was used to establish the OSCORE Security Context.

RS MUST discard the current OSCORE Security Context shared with C when any of the following occurs:

* The OSCORE Sender Sequence Number space of RS gets exhausted.

* The access token associated with the OSCORE Security Context becomes invalid, for example due to expiration or revocation.

* The current OSCORE Security Context shared with C has been successfully replaced with a newer one, following an unprotected POST request to the /authz-info endpoint at RS that re-posted a still valid access token together with a nonce N1, in order to trigger the use of the EDHOC-KeyUpdate function (see Section 5).

* The authentication credential of C (of RS) becomes invalid (e.g., due to expiration or revocation), and it was used as CRED_I (CRED_R) in the EDHOC execution whose PRK_out was used to establish the OSCORE Security Context.

After a new access token is successfully uploaded to RS, and a new OSCORE Security Context is established between C and RS, messages still in transit that were protected with the previous OSCORE Security Context might not be successfully verified by the recipient, since the old OSCORE Security Context might have been discarded. This means that messages sent shortly before C has uploaded the new access token to RS might not be successfully accepted by the recipient.

Furthermore, implementations may want to cancel CoAP observations at RS, if registered before the new OSCORE Security Context has been established. Alternatively, applications need to implement a mechanism to ensure that, from then on, messages exchanged within those observations are going to be protected with the newly derived OSCORE Security Context.

8. Security Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [RFC9200]. Thus, the general security considerations from the ACE framework also apply to this profile.

Furthermore, the security considerations from OSCORE [RFC8613] and from EDHOC [I-D.ietf-lake-edhoc] also apply to this specific use of the OSCORE and EDHOC protocols.

As previously stated, once completed the EDHOC execution, C and RS are mutually authenticated through their respective authentication credentials, whose retrieval has been facilitated by AS. Also once completed the EDHOC execution, C and RS have established a long-term secret key PRK_out enjoying forward secrecy. This is in turn used by C and RS to establish an OSCORE Security Context.

Furthermore, RS achieves confirmation that C has PRK_out (proof-of-possession) when completing the EDHOC execution. Rather, C achieves confirmation that RS has PRK_out (proof-of-possession) either when receiving the optional EDHOC message_4 from RS, or when successfully verifying a response from RS protected with the established OSCORE Security Context.

OSCORE is designed to secure point-to-point communication, providing a secure binding between a request and the corresponding response(s). Thus, the basic OSCORE protocol is not intended for use in point-to-multipoint communication (e.g., enforced via multicast or a publish-subscribe model). Implementers of this profile should make sure that their use case of OSCORE corresponds to the expected one, in order to prevent weakening the security assurances provided by OSCORE.

As defined in Section 5, C can (re-)post an access token to RS and contextually exchange two nonces N1 and N2, in order to efficiently use the EDHOC-KeyUpdate function rather than re-running the EDHOC protocol with RS. The use of nonces guarantees uniqueness of the new PRK_out derived by running EDHOC_KeyUpdate. Consequently, it ensures uniqueness of the AEAD (nonce, key) pairs later used by C and RS, when protecting their communications with the OSCORE Security Context established after updating PRK_out. Thus, it is REQUIRED that the exchanged nonces are not reused with the same pair of authentication credentials (AUTH_CRED_C, AUTH_CRED_RS), even in case of reboot. When using this profile, it is RECOMMENDED to use a 64-bit long random numbers as a nonce's value. Considering the birthday paradox, the average collision for each nonce will happen after 2^32 messages, which amounts to considerably more issued access token than it would be expected for intended applications. If applications use something else, such as a counter, they need to guarantee that reboot and loss of state on either node does not yield reuse of nonces. If that is not guaranteed, nodes are susceptible to reuse of AEAD (nonce, key) pairs, especially since an on-path attacker can cause the use of a previously exchanged nonce N1 by replaying the corresponding C-to-RS message.

When using this profile, it is RECOMMENDED that RS stores only one access token per client. The use of multiple access tokens for a single client increases the strain on RS, since it must consider every access token associated with the client and calculate the actual permissions that client has. Also, access tokens indicating different or disjoint permissions from each other may lead RS to enforce wrong permissions. If one of the access tokens expires earlier than others, the resulting permissions may offer insufficient protection. Developers SHOULD avoid using multiple access tokens for a same client. Furthermore, RS MUST NOT store more than one access token per client per PoP-key (i.e., per client's authentication credential).

9. Privacy Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [RFC9200]. Thus, the general privacy considerations from the ACE framework also apply to this profile.

Furthermore, the privacy considerations from OSCORE [RFC8613] and from EDHOC [I-D.ietf-lake-edhoc] also apply to this specific use of the OSCORE and EDHOC protocols.

An unprotected response to an unauthorized request may disclose information about RS and/or its existing relationship with C. It is advisable to include as little information as possible in an unencrypted response. When an OSCORE Security Context already exists between C and RS, more detailed information may be included.

Except for the case where C attempts to update its access rights, the (encrypted) access token is sent in an unprotected POST request to the /authz-info endpoint at RS. Thus, if C uses the same single access token from multiple locations, it can risk being tracked by the access token's value even when the access token is encrypted.

As defined in Section 5, C can (re-)post an access token to RS and contextually exchange two nonces N1 and N2, in order to efficiently use the EDHOC-KeyUpdate function rather than re-running the EDHOC protocol with RS. Since the exchanged nonces are also sent in the clear, using random nonces is best for privacy, as opposed to, e.g., a counter that might leak some information about C.

The identifiers used in OSCORE, i.e., the OSCORE Sender/Recipient IDs, are negotiated by C and RS during the EDHOC execution. That is, the EDHOC Connection Identifier C_I of C is going to be the OSCORE Recipient ID of C (the OSCORE Sender ID of RS). Conversely, the EDHOC Connection Identifier C_R of RS is going to be the OSCORE Recipient ID of RS (the OSCORE Sender ID of C). These OSCORE identifiers are privacy sensitive (see Section 12.8 of [RFC8613]). In particular, they could reveal information about C, or may be used for correlating different requests from C, e.g., across different networks that C has joined and left over time. This can be mitigated if C and RS dynamically update their OSCORE identifiers, e.g., by using the method defined in [I-D.ietf-core-oscore-key-update].

10. IANA Considerations

This document has the following actions for IANA.

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

10.1. ACE OAuth Profile Registry

IANA is asked to add the following entry to the "ACE OAuth Profile" Registry following the procedure specified in [RFC9200].

- *Profile name: coap_edhoc_oscore
- *Profile Description: Profile for delegating client authentication and authorization in a constrained environment by establishing an OSCORE Security Context [RFC8613] between resource-constrained nodes, through the execution of the authenticated key establishment protocol EDHOC [I-D.ietf-lake-edhoc].
- *Profile ID: TBD (value between 1 and 255)
- *Change Controller: IESG
- *Reference: [RFC-XXXX]

10.2. OAuth Parameters Registry

IANA is asked to add the following entries to the "OAuth Parameters" registry.

- *Name: "edhoc_info"
- *Parameter Usage Location: token request, token response
- *Change Controller: IESG
- *Specification Document(s): [RFC-XXXX]
- *Name: "token_uploaded"
- *Parameter Usage Location: token response
- *Change Controller: IESG
- *Specification Document(s): [RFC-XXXX]

10.3. OAuth Parameters CBOR Mappings Registry

IANA is asked to add the following entries to the "OAuth Parameters CBOR Mappings" following the procedure specified in [RFC9200].

- *Name: "edhoc_info"
- *CBOR Key: TBD
- *Value Type: map

* Specification Document(s): [RFC-XXXX]

* Name: "token_uploaded"

* CBOR Key: TBD

* Value Type: simple value "true" / simple type "false"

* Specification Document(s): [RFC-XXXX]

10.4. JSON Web Token Claims Registry

IANA is asked to add the following entries to the "JSON Web Token Claims" registry following the procedure specified in [RFC7519].

* Claim Name: "edhoc_info"

* Claim Description: Information for EDHOC execution

* Change Controller: IETF

* Reference: [RFC-XXXX]

10.5. CBOR Web Token Claims Registry

IANA is asked to add the following entries to the "CBOR Web Token Claims" registry following the procedure specified in [RFC8392].

* Claim Name: "edhoc_info"

* Claim Description: Information for EDHOC execution

* JWT Claim Name: "edhoc_info"

* Claim Key: TBD

* Claim Value Type(s): map

* Change Controller: IESG

* Specification Document(s): [RFC-XXXX]

10.6. JWT Confirmation Methods Registry

IANA is asked to add the following entries to the "JWT Confirmation Methods" registry following the procedure specified in [RFC7800].

* Confirmation Method Value: "x5bag"

* Confirmation Method Description: An unordered bag of X.509 certificates

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Value: "x5chain"

*Confirmation Method Description: An ordered chain of X.509
 certificates

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Value: "x5t"

*Confirmation Method Description: Hash of an X.509 certificate

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Value: "x5u"

*Confirmation Method Description: URI pointing to an X.509
 certificate

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Value: "c5b"

*Confirmation Method Description: An unordered bag of C509
 certificates

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Value: "c5c"

*Confirmation Method Description: An ordered chain of C509
 certificates

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Value: "c5t"

*Confirmation Method Description: Hash of an C509 certificate

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Value: "c5u"

*Confirmation Method Description: URI pointing to a COSE_C509 containing an ordered chain of certificates

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Value: "kcwt"

*Confirmation Method Description: A CBOR Web Token (CWT) containing a COSE_Key in a 'cnf' claim

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Value: "kccs"

*Confirmation Method Description: A CWT Claims Set (CCS) containing a COSE_Key in a 'cnf' claim

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

10.7. CWT Confirmation Methods Registry

IANA is asked to add the following entries to the "CWT Confirmation Methods" registry following the procedure specified in [RFC8747].

*Confirmation Method Name: x5bag

*Confirmation Method Description: An unordered bag of X.509 certificates

*JWT Confirmation Method Name: "x5bag"

*Confirmation Key: TBD

*Confirmation Value Type(s): COSE_X509

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Name: x5chain

*Confirmation Method Description: An ordered chain of X.509
 certificates

*JWT Confirmation Method Name: "x5chain"

*Confirmation Key: TBD

*Confirmation Value Type(s): COSE_X509

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Name: x5t

*Confirmation Method Description: Hash of an X.509 certificate

*JWT Confirmation Method Name: "x5t"

*Confirmation Key: TBD

*Confirmation Value Type(s): COSE_CertHash

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Name: x5u

*Confirmation Method Description: URI pointing to an X.509
 certificate

*JWT Confirmation Method Name: "x5u"

*Confirmation Key: TBD

*Confirmation Value Type(s): uri

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Name: c5b

*Confirmation Method Description: An unordered bag of C509
 certificates

*JWT Confirmation Method Name: "c5b"

*Confirmation Key: TBD

*Confirmation Value Type(s): COSE_C509

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Name: c5c

*Confirmation Method Description: An ordered chain of C509 certificates

*JWT Confirmation Method Name: "c5c"

*Confirmation Key: TBD

*Confirmation Value Type(s): COSE_C509

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Name: c5t

*Confirmation Method Description: Hash of an C509 certificate

*JWT Confirmation Method Name: "c5t"

*Confirmation Key: TBD

*Confirmation Value Type(s): COSE_CertHash

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Name: c5u

*Confirmation Method Description: URI pointing to a COSE_C509 containing an ordered chain of certificates

*JWT Confirmation Method Name: "c5u"

*Confirmation Key: TBD

*Confirmation Value Type(s): uri

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Name: kcwt

*Confirmation Method Description: A CBOR Web Token (CWT) containing a COSE_Key in a 'cnf' claim

*JWT Confirmation Method Name: "kcwt"

*Confirmation Key: TBD

*Confirmation Value Type(s): COSE_Messages

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

*Confirmation Method Name: kccs

*Confirmation Method Description: A CWT Claims Set (CCS) containing a COSE_Key in a 'cnf' claim

*JWT Confirmation Method Name: "kccs"

*Confirmation Key: TBD

*Confirmation Value Type(s): map / #6(map)

*Change Controller: IESG

*Specification Document(s): [RFC-XXXX]

10.8. EDHOC External Authorization Data Registry

IANA is asked to add the following entry to the "EDHOC External Authorization Data" registry defined in Section 9.5 of [I-D.ietf-lake-edhoc].

*Label: TBD

*Message: EDHOC message_1

*Description: "ead_value" specifies an access token

*Reference: [RFC-XXXX]

10.9. EDHOC Information Registry

It is requested that IANA create a new registry entitled "EDHOC Information" registry. The registry is to be created with registration policy Expert Review [RFC8126]. Guidelines for the experts are provided in Section 10.10. It should be noted that in addition to the expert review, some portions of the registry require a specification, potentially on Standards Track, be supplied as well.

The columns of the registry are:

* Name: A descriptive name that enables easier reference to this item. Because a core goal of this document is for the resulting representations to be compact, it is RECOMMENDED that the name be short.

  This name is case sensitive. Names may not match other registered names in a case-insensitive manner unless the Designated Experts determine that there is a compelling reason to allow an exception. The name is not used in the CBOR encoding.

* CBOR Value: The value to be used as CBOR abbreviation of the item.

  The value MUST be unique. The value can be a positive integer, a negative integer or a string. Integer values between -256 and 255 and strings of length 1 are to be registered by Standards Track documents (Standards Action). Integer values from -65536 to -257 and from 256 to 65535 and strings of maximum length 2 are to be registered by public specifications (Specification Required). Integer values greater than 65535 and strings of length greater than 2 are subject to the Expert Review policy. Integer values less than -65536 are marked as private use.

* CBOR Type: The CBOR type of the item, or a pointer to the registry that defines its type, when that depends on another item.

* Registry: The registry that values of the item may come from, if one exists.

* Description: A brief description of this item.

* Specification: A pointer to the public specification for the item, if one exists.

This registry will be initially populated by the values in Figure 6. The "Specification" column for all of these entries will be this document and [I-D.ietf-lake-edhoc].

10.10. Expert Review Instructions

The IANA registry established in this document is defined to use the registration policy Expert Review. This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

* Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. The zones tagged as private use are intended for testing purposes and closed environments; code points in other ranges should not be assigned for testing.

* Specifications are required for the Standards Action range of point assignment. Specifications should exist for Specification Required ranges, but early assignment before a specification is available is considered to be permissible. Specifications are needed for the first-come, first-serve range if they are expected to be used outside of closed environments in an interoperable way. When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for.

* Experts should take into account the expected usage of fields when approving point assignment. The fact that there is a range for Standards Track documents does not mean that a Standards Track document cannot have points assigned outside of that range. The length of the encoded value should be weighed against how many code points of that length are left, the size of device it will be used on, and the number of code points left that encode to that size.

11. References

11.1. Normative References

[I-D.ietf-core-oscore-edhoc] Palombini, F., Tiloca, M., Höglund, R., Hristozov, S., and G. Selander, "Profiling EDHOC for CoAP and OSCORE", Work in Progress, Internet-Draft, draft-ietf-core-oscore-edhoc-06, 23 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-edhoc-06>.

[I-D.ietf-cose-cbor-encoded-cert] Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-05, 10 January 2023,

<https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-05>.

[I-D.ietf-lake-edhoc] Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-19, 3 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-19>.

[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>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

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

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/rfc/rfc7800>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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>.

[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>.

[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/rfc/rfc8610>.

[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>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/rfc/rfc8742>.

[RFC8747] Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", RFC 8747, DOI 10.17487/RFC8747, March 2020, <https://www.rfc-editor.org/rfc/rfc8747>.

[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>.

[RFC9200] Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework (ACE-OAuth)", RFC 9200, DOI 10.17487/RFC9200, August 2022, <https://www.rfc-editor.org/rfc/rfc9200>.

[RFC9201] Seitz, L., "Additional OAuth Parameters for Authentication and Authorization for Constrained Environments (ACE)", RFC 9201, DOI 10.17487/RFC9201, August 2022, <https://www.rfc-editor.org/rfc/rfc9201>.

[RFC9203] Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "The Object Security for Constrained RESTful Environments (OSCORE) Profile of the Authentication and Authorization for Constrained Environments (ACE) Framework", RFC 9203, DOI 10.17487/RFC9203, August 2022, <https://www.rfc-editor.org/rfc/rfc9203>.

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

11.2. Informative References

[I-D.ietf-core-oscore-key-update] Höglund, R. and M. Tiloca, "Key Update for OSCORE (KUDOS)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-key-update-03, 24 October

2022, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-key-update-03>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/rfc/rfc4949>.

[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/rfc/rfc8446>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

Appendix A. Examples

This appendix provides examples where this profile of ACE is used. In particular:

* Appendix A.1 does not make use of use of any optimization.

* Appendix A.2 makes use of the optimizations defined in this specification, hence reducing the roundtrips of the interactions between the Client and the Resource Server.

* Appendix A.3 does not make use of any optimization, but consider an alternative workflow where AS uploads the access token to RS.

All these examples build on the following assumptions, as relying on expected early procedures performed at AS. These include the registration of RSs by the respective Resource Owners as well as the registrations of Clients authorized to request access token for those RSs.

* AS knows the authentication credential AUTH_CRED_C of the Client C.

* The Client knows the authentication credential AUTH_CRED_AS of AS.

* AS knows the authentication credential AUTH_CRED_RS of RS.

* RS knows the authentication credential AUTH_CRED_AS of AS.

This is relevant in case AS and RS actually require a secure association (e.g., for RS to perform token introspection at AS, or for AS to upload an access token to RS on behalf of the Client).

As a result of the assumptions above, it is possible to limit the transport of AUTH_CRED_C and AUTH_CRED_RS by value only to the following two cases, and only when the Client requests an access token for RS in question for the first time when considering the pair (AUTH_CRED_C, AUTH_CRED_RS).

* In the Token Response from AS to the Client, where AUTH_CRED_RS is specified by the 'rs_cnf' parameter.

* In the access token, where AUTH_CRED_C is specified by the 'cnf' claim.

Note that, even under the circumstances mentioned above, AUTH_CRED_C might rather be indicated by reference. This is possible if RS can effectively use such a reference from the access token to retrieve AUTH_CRED_C (e.g., from a trusted repository of authentication credentials reachable through a non-constrained link), and if AS is in turn aware of that.

In any other case, it is otherwise possible to indicate both AUTH_CRED_C and AUTH_CRED_RS by reference, when performing the ACE access control workflow as well as later on when the Client and RS run EDHOC.

A.1. Workflow without Optimizations

The example below considers the simplest (though least efficient) interaction between the Client and RS. That is: first C uploads the access token to RS; then C and RS run EDHOC; and, finally, the Client accesses the protected resource at RS.

```
    C                                 AS                             RS
    |                                  |                              |
    |  EDHOC message_1 to /edhoc       |                              |
M01 |--------------------------------->|                              |
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_2                 |                              |
M02 |<---------------------------------|                              |
    |  ID_CRED_R identifies            |                              |
    |     CRED_R = AUTH_CRED_AS        |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_3 to /edhoc       |                              |
M03 |--------------------------------->|                              |
    |  ID_CRED_I identifies            |                              |
    |     CRED_I = AUTH_CRED_C         |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  Token request to /token         |                              |
    |  (OSCORE-protected message)      |                              |
M04 |--------------------------------->|                              |
    |  'req_cnf' identifies            |                              |
    |     AUTH_CRED_C by reference     |                              |
    |                                  |                              |
    |                                  |                              |
    |  Token response                  |                              |
    |  (OSCORE-protected message)      |                              |
M05 |<---------------------------------|                              |
    |  'rs_cnf' specifies              |                              |
    |     AUTH_CRED_RS by value        |                              |
    |                                  |                              |
    |  'ace_profile' =                 |                              |
    |             coap_edhoc_oscore    |                              |
    |                                  |                              |
    |  'edhoc_info' specifies:         |                              |
    |     {                            |                              |
    |       id : h'01',                |                              |
    |       cipher_suites : 2,         |                              |
    |       methods : 3                |                              |
    |     }                            |                              |
    |                                  |                              |
    |  In the access token:            |                              |
    |     * the 'cnf' claim specifies  |                              |
    |       AUTH_CRED_C by value       |                              |
    |     * the 'edhoc_info' claim     |                              |
    |       specifies the same as      |                              |
    |       'edhoc_info' above         |                              |
    |                                  |                              |

 // Possibly after chain verification, the Client adds AUTH_CRED_RS
 // to the set of its trusted peer authentication credentials,
 // relying on AS as trusted provider

    |                                  |                              |
    |  Token upload to /authz-info     |                              |
    |  (unprotected message)           |                              |
M06 |---------------------------------------------------------------->|
    |                                  |                              |

 // Possibly after chain verification, RS adds AUTH_CRED_C
 // to the set of its trusted peer authentication credentials,
 // relying on AS as trusted provider

    |                                  |                              |
    |   2.01 (Created)                 |                              |
    |   (unprotected message)          |                              |
M07 |<----------------------------------------------------------------|
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_1 to /edhoc       |                              |
    |  (no access control is enforced) |                              |
M08 |---------------------------------------------------------------->|
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_2                 |                              |
M09 |<----------------------------------------------------------------|
    |  ID_CRED_R identifies            |                              |
    |     CRED_R = AUTH_CRED_RS        |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_3 to /edhoc       |                              |
    |  (no access control is enforced) |                              |
M10 |---------------------------------------------------------------->|
    |  ID_CRED_I identifies            |                              |
    |     CRED_I = AUTH_CRED_C         |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  Access to protected resource    |                              |
    |  (OSCORE-protected message)      |                              |
    |  (access control is enforced)    |                              |
M11 |---------------------------------------------------------------->|
    |                                  |                              |
    |  Response                        |                              |
    |  (OSCORE-protected message)      |                              |
M12 |<----------------------------------------------------------------|
    |                                  |                              |

 // Later on, the access token expires ...
 //  - The Client and RS delete their OSCORE Security Context and
 //    purge the EDHOC session used to derive it (unless the same
 //    session is also used for other reasons).
 //  - RS retains AUTH_CRED_C as still valid,
 //    and AS knows about it.
 //  - The Client retains AUTH_CRED_RS as still valid,
 //    and AS knows about it.

    |                                  |                              |
    |                                  |                              |

 // Time passes ...

    |                                  |                              |
    |                                  |                              |

 // The Client asks for a new access token; now all the
 // authentication credentials can be indicated by reference

 // The price to pay is on AS, about remembering that at least
 // one access token has been issued for the pair (Client, RS)
 // and considering the pair (AUTH_CRED_C, AUTH_CRED_RS)

    |                                  |                              |
    |  Token request to /token         |                              |
    |  (OSCORE-protected message)      |                              |
M13 |--------------------------------->|                              |
    |  'req_cnf' identifies            |                              |
    |     CRED_I = AUTH_CRED_C         |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  Token response                  |                              |
    |  (OSCORE-protected message)      |                              |
M14 |<---------------------------------|                              |
    |  'rs_cnf' identifies             |                              |
    |     AUTH_CRED_RS by reference    |                              |
    |                                  |                              |
    |  'ace_profile' =                 |                              |
    |             coap_edhoc_oscore    |                              |
    |                                  |                              |
    |  'edhoc_info' specifies:         |                              |
    |     {                            |                              |
    |       id : h'05',                |                              |
    |       cipher_suites : 2,         |                              |
    |       methods : 3               |                               |
    |     }                           |                               |
    |                                 |                               |
    |  In the access token:           |                               |
    |     * the 'cnf' claim specifies |                               |
    |       AUTH_CRED_C by reference  |                               |
    |     * the 'edhoc_info' claim    |                               |
    |       specifies the same as     |                               |
    |       'edhoc_info' above        |                               |
    |                                 |                               |
    |                                 |                               |
    |  Token upload to /authz-info    |                               |
    |  (unprotected message)          |                               |
M15 |---------------------------------------------------------------->|
    |                                 |                               |
    |                                 |                               |
    |  2.01 (Created)                 |                               |
    |  (unprotected message)          |                               |
M16 |<----------------------------------------------------------------|
    |                                 |                               |
    |                                 |                               |
    |  EDHOC message_1 to /edhoc      |                               |
    |  (no access control is enforced) |                              |
M17 |---------------------------------------------------------------->|
    |                                 |                               |
    |                                 |                               |
    |  EDHOC message_2                |                               |
    |  (no access control is enforced) |                              |
M18 |<----------------------------------------------------------------|
    |  ID_CRED_R specifies            |                               |
    |     CRED_R = AUTH_CRED_RS       |                               |
    |     by reference                |                               |
    |                                 |                               |
    |                                 |                               |
    |  EDHOC message_3 to /edhoc      |                               |
    |  (no access control is enforced) |                              |
M19 |---------------------------------------------------------------->|
    |  ID_CRED_I identifies           |                               |
    |     CRED_I = AUTH_CRED_C        |                               |
    |     by reference                |                               |
    |                                 |                               |
    |                                 |                               |
    |  Access to protected resource /r |                              |
    |  (OSCORE-protected message)     |                               |
    |  (access control is enforced)   |                               |
M20 |---------------------------------------------------------------->|
    |                                 |                               |
    |                                 |                               |
    |  Response                       |                               |
    |  (OSCORE-protected message)      |                              |
M21 |<-----------------------------------------------------------------|
    |                                  |                              |
```

A.2. Workflow with Optimizations

The example below builds on the example in Appendix A.1, while additionally relying on the two following optimizations.

* The access token is not separately uploaded to the /authz-info endpoint at RS, but rather included in the EAD_1 field of EDHOC message_1 sent by the C to RS.

* The Client uses the EDHOC+OSCORE request defined in [I-D.ietf-core-oscore-edhoc] is used, when running EDHOC both with AS and with RS.

These two optimizations used together result in the most efficient interaction between the C and RS, as consisting of only two roundtrips to upload the access token, run EDHOC and access the protected resource at RS.

Also, a further optimization is used upon uploading a second access token to RS, following the expiration of the first one. That is, after posting the second access token, the Client and RS do not run EDHOC again, but rather EDHOC-KeyUpdate() and EDHOC-Exporter() building on the same, previously completed EDHOC execution.

```
       C                                 AS                             RS
       |                                  |                              |
       |  EDHOC message_1 to /edhoc       |                              |
   M01 |--------------------------------->|                              |
       |                                  |                              |
       |                                  |                              |
       |  EDHOC message_2                 |                              |
   M02 |<---------------------------------|                              |
       |  ID_CRED_R identifies            |                              |
       |     CRED_R = AUTH_CRED_AS        |                              |
       |     by reference                 |                              |
       |                                  |                              |
       |                                  |                              |
       |  EDHOC+OSCORE request to /token  |                              |
   M03 |--------------------------------->|                              |
       |  * EDHOC message_3               |                              |
       |      ID_CRED_I identifies        |                              |
       |         CRED_I = AUTH_CRED_C     |                              |
       |         by reference             |                              |
       |  --- --- ---                     |                              |
       |  * OSCORE-protected part         |                              |
       |      Token request               |                              |
       |         'req_cnf' identifies     |                              |
       |         AUTH_CRED_C by reference |                              |
       |                                  |                              |
       |                                  |                              |
       |  Token response                  |                              |
       |  (OSCORE-protected message)      |                              |
   M04 |<---------------------------------|                              |
       |  'rs_cnf' specifies              |                              |
       |     AUTH_CRED_RS by value        |                              |
       |                                  |                              |
       |  'ace_profile' =                 |                              |
       |             coap_edhoc_oscore    |                              |
       |                                  |                              |
       |  'edhoc_info' specifies:         |                              |
       |     {                            |                              |
       |       id : h'01',                |                              |
       |       cipher_suites : 2,         |                              |
       |       methods : 3                |                              |
       |     }                            |                              |
       |                                  |                              |
       |  In the access token:            |                              |
       |     * the 'cnf' claim specifies  |                              |
       |       AUTH_CRED_C by value       |                              |
       |     * the 'edhoc_info' claim     |                              |
       |       specifies the same as      |                              |
       |       'edhoc_info' above         |                              |
       |                                  |                              |
    // Possibly after chain verification, the Client adds AUTH_CRED_RS
    // to the set of its trusted peer authentication credentials,
    // relying on AS as trusted provider

       |                                  |                              |
       |  EDHOC message_1 to /edhoc       |                              |
       |  (no access control is enforced) |                              |
   M05 |---------------------------------------------------------------->|
       |  Access token specified in EAD_1 |                              |
       |                                  |                              |

    // Possibly after chain verification, RS adds AUTH_CRED_C
    // to the set of its trusted peer authentication credentials,
    // relying on AS as trusted provider

       |                                  |                              |
       |  EDHOC message_2                 |                              |
   M06 |<----------------------------------------------------------------|
       |  ID_CRED_R identifies            |                              |
       |     CRED_R = AUTH_CRED_RS        |                              |
       |     by reference                 |                              |
       |                                  |                              |
       |                                  |                              |
       |  EDHOC+OSCORE request to /r      |                              |
   M07 |---------------------------------------------------------------->|
       |  * EDHOC message_3               |                              |
       |      ID_CRED_I identifies        |                              |
       |         CRED_I = AUTH_CRED_C     |                              |
       |         by reference             |                              |
       |  --- --- ---                     |                              |
       |  * OSCORE-protected part         |                              |
       |      Application request to /r   |                              |
       |                                  |                              |

    // After the EDHOC processing is completed, access control
    // is enforced on the rebuilt OSCORE-protected request,
    // like if it had been sent stand-alone

       |                                  |                              |
       |  Response                        |                              |
       |  (OSCORE-protected message)      |                              |
   M08 |<----------------------------------------------------------------|
       |                                  |                              |

    // Later on, the access token expires ...
    //  - The Client and RS delete their OSCORE Security Context
    //    but do not purge the EDHOC session used to derive it.
    //  - RS retains AUTH_CRED_C as still valid,
    //    and AS knows about it.
    //  - The Client retains AUTH_CRED_RS as still valid,
    //    and AS knows about it.

       |                                  |                              |
       |                                  |                              |

    // Time passes ...

       |                                  |                              |
       |                                  |                              |

    // The Client asks for a new access token; now all the
    // authentication credentials can be indicated by reference

    // The price to pay is on AS, about remembering that at least
    // one access token has been issued for the pair (Client, RS)
    // and considering the pair (AUTH_CRED_C, AUTH_CRED_RS)

       |                                  |                              |
       |  Token request to /token         |                              |
       |  (OSCORE-protected message)      |                              |
   M09 |--------------------------------->|                              |
       |  'req_cnf' identifies            |                              |
       |     CRED_I = AUTH_CRED_C         |                              |
       |     by reference                 |                              |
       |                                  |                              |
       |  Token response                  |                              |
       |  (OSCORE-protected message)      |                              |
   M10 |<---------------------------------|                              |
       |  'rs_cnf' identifies             |                              |
       |     AUTH_CRED_RS by reference    |                              |
       |                                  |                              |
       |  'ace_profile' =                 |                              |
       |             coap_edhoc_oscore    |                              |
       |                                  |                              |
       |  'edhoc_info' specifies:         |                              |
       |     {                            |                              |
       |       id : h'05',                |                              |
       |       cipher_suites : 2,         |                              |
       |       methods : 3                |                              |
       |     }                            |                              |
       |                                  |                              |
       |  In the access token:            |                              |
       |     * the 'cnf' claim specifies  |                              |
       |       AUTH_CRED_C by reference   |                              |
       |     * the 'edhoc_info' claim     |                              |
       |       specifies the same as      |                              |
       |       'edhoc_info' above         |                              |
       |                                  |                              |
       |                                  |                              |
       |  Token upload to /authz-info     |                              |
       |  (unprotected message)           |                              |
   M11 |---------------------------------------------------------------->|
       |   Payload {                      |                              |
       |     access_token : access token  |                              |
       |     nonce_1 : N1  // nonce       |                              |
       |   }                              |                              |
       |                                  |                              |
       |                                  |                              |
       |  2.01 (Created)                  |                              |
       |  (unprotected message)           |                              |
   M12 |<----------------------------------------------------------------|
       |   Payload {                      |                              |
       |     nonce_2 : N2  // nonce       |                              |
       |   }                              |                              |
       |                                  |                              |

    // The Client and RS first run EDHOC-KeyUpdate(N1 | N2), and
    // then EDHOC-Exporter() to derive a new OSCORE Master Secret and
    // OSCORE Master Salt, from which a new OSCORE Security Context is
    // derived. The Sender/Recipient IDs are the same C_I and C_R from
    // the previous EDHOC execution

       |                                  |                              |
       |  Access to protected resource /r |                              |
       |  (OSCORE-protected message)      |                              |
       |  (access control is enforced)    |                              |
   M13 |---------------------------------------------------------------->|
       |                                  |                              |
       |                                  |                              |
       |  Response                        |                              |
       |  (OSCORE-protected message)      |                              |
   M14 |<----------------------------------------------------------------|
       |                                  |                              |
```

A.3. Workflow without Optimizations (AS token posting)

The example below builds on the example in Appendix A.1, but assumes that AS is uploading the access token to RS on behalf of C.

In order to save roundtrips between the Client and RS, further, more efficient interactions can be seamlessly considered, e.g., as per the example in Appendix A.2.

```
    C                                 AS                             RS
    |                                  |                              |
    |                                  | Establish secure association |
    |                                  | (e.g., OSCORE using EDHOC)   |
    |                                  |<---------------------------->|
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_1 to /edhoc       |                              |
M01 |--------------------------------->|                              |
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_2                 |                              |
M02 |<---------------------------------|                              |
    |  ID_CRED_R identifies            |                              |
    |     CRED_R = AUTH_CRED_AS        |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_3 to /edhoc       |                              |
M03 |--------------------------------->|                              |
    |  ID_CRED_I identifies            |                              |
    |     CRED_I = AUTH_CRED_C         |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  Token request to /token         |                              |
    |  (OSCORE-protected message)      |                              |
M04 |--------------------------------->|                              |
    |  'req_cnf' identifies            |                              |
    |     AUTH_CRED_C by reference     |                              |
    |                                  |                              |
    |                                  |                              |
    |                                  |  Token upload to /authz-info |
M05 |                                  |----------------------------->|
    |                                  |  In the access token:        |
    |                                  |     * the 'cnf' claim        |
    |                                  |       specifies AUTH_CRED_C  |
    |                                  |       by value               |
    |                                  |     * the 'edhoc_info'       |
    |                                  |       claim specifies        |
    |                                  |         {                    |
    |                                  |           id : h'01',        |
    |                                  |           cipher_suites : 2, |
    |                                  |           methods: 3         |
    |                                  |         }                    |
    |                                  |                              |

 // Possibly after chain verification, RS adds AUTH_CRED_C
 // to the set of its trusted peer authentication credentials,
 // relying on AS as trusted provider

    |                                  |                              |
    |                                  |  2.01 (Created)              |
M06 |                                  |<-----------------------------|
    |                                  |                              |
    |                                  |                              |
    |  Token response                  |                              |
    |  (OSCORE-protected message)      |                              |
M07 |<---------------------------------|                              |
    |  'rs_cnf' specifies              |                              |
    |     AUTH_CRED_RS by value        |                              |
    |                                  |                              |
    |  'ace_profile' =                 |                              |
    |             coap_edhoc_oscore    |                              |
    |                                  |                              |
    |  'token_uploaded' = true         |                              |
    |                                  |                              |
    |  'edhoc_info' specifies:         |                              |
    |     {                            |                              |
    |       id : h'01',                |                              |
    |       cipher_suites  : 2,        |                              |
    |       methods : 3                |                              |
    |     }                            |                              |
    |                                  |                              |


 // Possibly after chain verification, the Client adds AUTH_CRED_RS
 // to the set of its trusted peer authentication credentials,
 // relying on AS as trusted provider

    |                                  |                              |
    |  EDHOC message_1 to /edhoc       |                              |
    |  (no access control is enforced) |                              |
M08 |---------------------------------------------------------------->|
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_2                 |                              |
M09 |<----------------------------------------------------------------|
    |  ID_CRED_R identifies            |                              |
    |     CRED_R = AUTH_CRED_RS        |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_3 to /edhoc       |                              |
    |  (no access control is enforced) |                              |
M10 |---------------------------------------------------------------->|
    |  ID_CRED_I identifies            |                              |
    |     CRED_I = AUTH_CRED_C         |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  Access to protected resource    |                              |
    |  (OSCORE-protected message)      |                              |
    |  (access control is enforced)    |                              |
M11 |---------------------------------------------------------------->|
    |                                  |                              |
    |                                  |                              |
    |  Response                        |                              |
    |  (OSCORE-protected message)      |                              |
M12 |<----------------------------------------------------------------|
    |                                  |                              |

 // Later on, the access token expires ...
 //  - The Client and RS delete their OSCORE Security Context and
 //    purge the EDHOC session used to derive it (unless the same
 //    session is also used for other reasons).
 //  - RS retains AUTH_CRED_C as still valid,
 //    and AS knows about it.
 //  - The Client retains AUTH_CRED_RS as still valid,
 //    and AS knows about it.

    |                                  |                              |
    |                                  |                              |

 // Time passes ...

    |                                  |                              |
    |                                  |                              |

 // The Client asks for a new access token; now all the
 // authentication credentials can be indicated by reference

 // The price to pay is on AS, about remembering that at least
 // one access token has been issued for the pair (Client, RS)
 // and considering the pair (AUTH_CRED_C, AUTH_CRED_RS)

    |                                  |                              |
    |  Token request to /token         |                              |
    |  (OSCORE-protected message)      |                              |
M13 |--------------------------------->|                              |
    |  'req_cnf' identifies            |                              |
    |     CRED_I = AUTH_CRED_C         |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |                                  |  Token upload to /authz-info |
M14 |                                  |----------------------------->|
    |                                  |  In the access token:        |
    |                                  |     * the 'cnf' claim        |
    |                                  |       specifies AUTH_CRED_C  |
    |                                  |       by reference           |
    |                                  |     * the 'edhoc_info'       |
    |                                  |       claim specifies        |
    |                                  |         {                    |
    |                                  |           id : h'05',        |
    |                                  |           cipher_suites : 2, |
    |                                  |           methods : 3        |
    |                                  |         }                    |
    |                                  |                              |
    |                                  |                              |
    |                                  |  2.01 (Created)              |
M15 |                                  |<-----------------------------|
    |                                  |                              |
    |                                  |                              |
    |  Token response                  |                              |
    |  (OSCORE-protected message)      |                              |
M16 |<---------------------------------|                              |
    |  'rs_cnf' specifies              |                              |
    |     AUTH_CRED_RS by reference    |                              |
    |                                  |                              |
    |  'ace_profile' =                 |                              |
    |             coap_edhoc_oscore    |                              |
    |                                  |                              |
    |  'token_uploaded' = true         |                              |
    |                                  |                              |
    |  'edhoc_info' specifies:         |                              |
    |     {                            |                              |
    |       id : h'05',                |                              |
    |       cipher_suites : 2,         |                              |
    |       methods : 3                |                              |
    |     }                            |                              |
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_1 to /edhoc       |                              |
    |  (no access control is enforced) |                              |
M17 |---------------------------------------------------------------->|
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_2                 |                              |
    |  (no access control is enforced) |                              |
M18 |<----------------------------------------------------------------|
    |  ID_CRED_R specifies             |                              |
    |     CRED_R = AUTH_CRED_RS        |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  EDHOC message_3 to /edhoc       |                              |
    |  (no access control is enforced) |                              |
M19 |---------------------------------------------------------------->|
    |  ID_CRED_I identifies            |                              |
    |     CRED_I = AUTH_CRED_C         |                              |
    |     by reference                 |                              |
    |                                  |                              |
    |                                  |                              |
    |  Access to protected resource /r |                              |
    |  (OSCORE-protected message)      |                              |
    |  (access control is enforced)    |                              |
M20 |---------------------------------------------------------------->|
    |                                  |                              |
    |                                  |                              |
    |  Response                        |                              |
    |  (OSCORE-protected message)      |                              |
M21 |<----------------------------------------------------------------|
    |                                  |                              |
```

Appendix B. Profile Requirements

This section lists the specifications of this profile based on the requirements of the framework, as requested in Appendix C of [RFC9200].

* Optionally, define new methods for the client to discover the necessary permissions and AS for accessing a resource, different from the one proposed in [RFC9200]: Not specified

* Optionally, specify new grant types: Not specified

* Optionally, define the use of client certificates as client credential type: C can use authentication credentials of any type admitted by the EDHOC protocol, including public key certificates such as X.509 and C509 certificates.

* Specify the communication protocol the client and RS must use: CoAP

* Specify the security protocol the client and RS must use to protect their communication: OSCORE

* Specify how the client and the RS mutually authenticate: Explicitly, by successfully executing the EDHOC protocol, after which a common OSCORE Security Context is established from the EDHOC session keying material. As per the EDHOC authentication method used during the EDHOC session, authentication is provided by digital signatures, or by Message Authentication Codes (MACs) computed from an ephemeral-static ECDH shared secret.

* Specify the proof-of-possession protocol(s) and how to select one, if several are available. Also specify which key types (e.g., symmetric/asymmetric) are supported by a specific proof-of- possession protocol: proof-of-possession is first achieved by RS when successfully processing EDHOC message_3 during the EDHOC execution with C, through EDHOC algorithms and symmetric EDHOC session keys. Also, proof-of-possession is later achieved by C when receiving from RS: i) the optional EDHOC message_4 during the EDHOC execution with RS, through EDHOC algorithms and symmetric EDHOC session keys; or ii) the first response protected with the OSCORE Security Context established after the EDHOC execution with RS, through OSCORE algorithms and OSCORE symmetric keys derived from the completed EDHOC session.

* Specify a unique ace_profile identifier: coap_edhoc_oscore

* If introspection is supported, specify the communication and security protocol for introspection: HTTP/CoAP (+ TLS/DTLS/ OSCORE)

* Specify the communication and security protocol for interactions between client and AS: HTTP/CoAP (+ TLS/DTLS/OSCORE)

* Specify if/how the authz-info endpoint is protected, including how error responses are protected: Not protected

* Optionally, define methods of token transport other than the authz-info endpoint: C can upload the access token when executing EDHOC with RS, by including the access token in the EAD_1 field of EDHOC message_1 (see Section 4.3).

Appendix C. Document Updates

RFC EDITOR: PLEASE REMOVE THIS SECTION.

C.1. Version -00 to -01

* Fixed semantics of the ead_value for transporting an Access Token in the EAD_1 field.

* Error handling aligned with EDHOC.

* Precise characterization of the EDHOC execution considered for EDHOC-KeyUpdate.

* Fixed message exchange examples.

* Added appendix with profile requirements.

* Updated references.

* Clarifications and editorial improvements.

Acknowledgments

The authors sincerely thank Christian Amsüss and Carsten Bormann for their comments and feedback.

Work on this document has in part been supported by the H2020 project SIFIS-Home (grant agreement 952652).

Authors' Addresses

Göran Selander
Ericsson

Email: goran.selander@ericsson.com

John Preuß Mattsson
Ericsson

Email: john.mattsson@ericsson.com

Marco Tiloca
RISE

Email: marco.tiloca@ri.se

Rikard Höglund
RISE

Email: rikard.hoglund@ri.se