ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 4, 2019

L. Seitz
RISE SICS AB
F. Palombini
Ericsson AB
M. Gunnarsson
RISE SICS AB
G. Selander
Ericsson AB

October 1, 2018

OSCORE profile of the Authentication and Authorization for Constrained Environments Framework
draft-ietf-ace-oscore-profile-03

Abstract

This memo specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework. It utilizes Object Security for Constrained RESTful Environments (OSCORE) to provide communication security, server authentication, and proof-of-possession for a key owned by the client and bound to an OAuth 2.0 access token.

Table of Contents

1. Introduction

This memo specifies a profile of the ACE framework [I-D.ietf-ace-oauth-authz]. In this profile, a client and a resource server use CoAP [RFC7252] to communicate. The client uses an access token, bound to a key (the proof-of-possession key) to authorize its access to the resource server. In order to provide communication security, proof of possession, and server authentication they use Object Security for Constrained RESTful Environments (OSCORE) [I-D.ietf-core-object-security].

OSCORE specifies how to use CBOR Object Signing and Encryption (COSE) [RFC8152] to secure CoAP messages. Note that OSCORE can be used to secure CoAP messages, as well as HTTP and combinations of HTTP and CoAP; a profile of ACE similar to the one described in this document, with the difference of using HTTP instead of CoAP as communication protocol, could be specified analogously to this one.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. These words may also appear in this document in lowercase, absent their normative meanings.

Certain security-related terms such as "authentication", "authorization", "confidentiality", "(data) integrity", "message authentication code", and "verify" are taken from [RFC4949].

RESTful terminology follows HTTP [RFC7231].

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

2. Protocol Overview

This section gives an overview on how to use the ACE Framework [I-D.ietf-ace-oauth-authz] to secure the communication between a client and a resource server using OSCORE [I-D.ietf-core-object-security]. The parameters needed to negotiate the use of this profile with the token resource at the authorization server as specified in section 5.6 of [I-D.ietf-ace-oauth-authz] are described in detail in the following sections.

This profile requires a client to retrieve an access token from the AS for the resource it wants to access on a RS, using the token endpoint, as specified in section 5.6.1 of [I-D.ietf-ace-oauth-authz]. To determine the AS in charge of a resource hosted at the RS, the client C MAY send an initial Unauthorized Resource Request message to the RS. The RS then denies the request and sends the address of its AS back to the client C as specified in section 5.1 of [I-D.ietf-ace-oauth-authz]. The access token request and response MUST be confidentiality-protected and ensure authenticity. This profile RECOMMENDS the use of OSCORE between client and AS, but TLS or DTLS MAY be used additionally or instead.

Once the client has retrieved the access token, it forwards it to the RS using the authz-info endpoint and mechanisms specified in section 5.8.1. of [I-D.ietf-ace-oauth-authz]. If the access token is valid, the RS replies to this request with a 2.01 (Created) response, which contains a nonce N1.

After receiving the nonce N1, the client generates a nonce N2, concatenates it with N1 and sets the ID Context in its Security Context (see section 3 of [I-D.ietf-core-object-security]) to N1 concatenated with N2. The client then derives the complete Security Context from the ID Context plus the parameters received from the AS.

Finally, the client sends a request protected with OSCORE to the RS. This message contains the ID Context value. When receiving this request after the 2.01 (Created) response, the server extract the ID Context from it, verifies that the first part is equal to the nonce N1 it previously sent, and if so, sets its own ID Context and derives the complete Security Context from it plus the parameters received in the AS, following section 3.2 of [I-D.ietf-core-object-security]. If the request verifies, then this Security Context is stored in the server, and used in the response, and in further communications with the client, until token expiration. The client will not include the ID Context value in further requests.

An overview of the profile flow for the OSCORE profile is given in Figure 1.

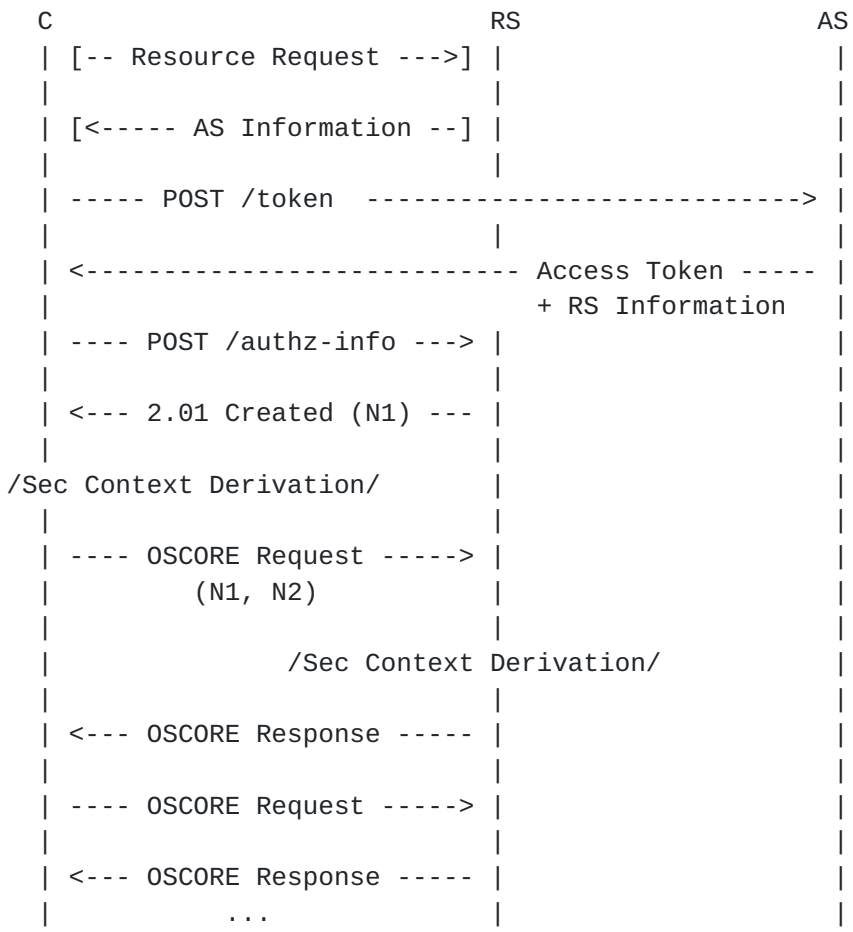

Figure 1: Protocol Overview

3. Client-AS Communication

The following subsections describe the details of the POST /token request and response between client and AS. Section 3.2 of [I-D.ietf-core-object-security] defines how to derive a security context based on a shared master secret and a set of other parameters, established between client and server, which the client receives from the AS in this exchange. The proof-of-possession key (pop-key) provisioned from the AS MUST be used as master secret in OSCORE.

3.1. C-to-AS: POST /token

The client-to-AS request is specified in Section 5.6.1 of [I-D.ietf-ace-oauth-authz].

If the client wants to update its access rights using the same OSCORE Security Context, it MUST include in its POST /token request a cnf

object carrying the Sender ID in the kid field. This identifier can be used by the AS to determine the shared secret to construct the proof-of-possession token and therefore MUST specify a symmetric key that was previously generated by the AS as a shared secret for the communication between the client and the RS.

The client MUST send this POST /token request over a secure channel that guarantees authentication, message integrity and confidentiality (see Section 5).

An example of such a request, in CBOR diagnostic notation without the tag and value abbreviations is reported in Figure 2

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
{
  "grant_type" : "client_credentials",
  "client_id" : "myclient",
  "aud" : "tempSensor4711"
 }
```

Figure 2: Example C-to-AS POST /token request for an access token bound to a symmetric key.

3.2. AS-to-C: Access Token

After verifying the POST /token request and that the client is authorized to obtain an access token corresponding to its access token request, the AS responds as defined in section 5.6.2 of [I-D.ietf-ace-oauth-authz]. It signals that the use of OSCORE is REQUIRED for a specific access token by including the "profile" parameter with the value "coap_oscore" in the access token response. This means that the client MUST use OSCORE towards all resource servers for which this access token is valid, and follow Section 4.3 to derive the security context to run OSCORE.

The error response procedures defined in section 5.6.3 of the ACE framework are unchanged by this profile.

Moreover, the AS MUST provision the following data:

- a master secret
- a client identifier

- a server identifier

Additionally, the AS MAY provision the following data, in the same response.

- an AEAD algorithm
- an HKDF algorithm
- a salt
- a replay window type and size

The master secret MUST be communicated as COSE_Key in the 'cnf' parameter of the access token response as defined in Section 5.6.4.5 of [I-D.ietf-ace-oauth-authz]. The AEAD algorithm MAY be included as the 'alg' parameter in the COSE_Key; the HKDF algorithm MAY be included as the 'hkdf' parameter of the COSE_Key and the salt MAY be included as the 'slt' parameter of the COSE_Key as defined in Figure 3.

The same parameters MUST be included as metadata of the access token. This profile RECOMMENDS the use of CBOR web token (CWT) as specified in [RFC8392]. If the token is a CWT, the same COSE_Key structure defined above MUST be placed in the 'cnf' claim of this token.

The AS MUST also assign identifiers to both client and RS, which are then used as Sender ID and Recipient ID in the OSCORE context as described in section 3.1 of [I-D.ietf-core-object-security]. These identifiers MUST be unique in the set of all clients and RS identifiers for a certain AS. Moreover, these MUST be included in the COSE_Key as header parameters, as defined in Figure 3.

We assume in this document that a resource is associated to one single AS, which makes it possible to assume unique identifiers for each client requesting a particular resource to a RS. If this is not the case, collisions of identifiers may appear in the RS, in which case the RS needs to have a mechanism in place to disambiguate identifiers or mitigate their effect.

Note that C should set the Sender ID of its Security Context to the clientId value received and the Recipient ID to the serverId value, and RS should do the opposite.

name	label	CBOR type	registry	description
clientId	TBD1	bstr		Identifies the client in an OSCORE context using this key
serverId	TBD2	bstr		Identifies the server in an OSCORE context using this key
hkdf	TBD3	bstr		Identifies the KDF algorithm in an OSCORE context using this key
slt	TBD4	bstr		Identifies the master salt in an OSCORE context using this key

Figure 3: Additional COSE_Key Common Parameters

Figure 4 shows an example of such an AS response, in CBOR diagnostic notation without the tag and value abbreviations.

```
Header: Created (Code=2.01)
Content-Type: "application/cose+cbor"
Payload:
{
  "access_token" : b64'SlAV32hkKG ...
   (remainder of access token omitted for brevity)',
  "profile" : "coap_oscore",
  "expires_in" : "3600",
  "cnf" : {
    "COSE_Key" : {
      "kty" : "Symmetric",
      "alg" : "AES-CCM-16-64-128",
      "clientId" : b64'qA',
      "serverId" : b64'Qg',
      "k" : b64'+a+Dg2jjU+eIiOFCa9lObw'
    }
  }
}
```

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

Figure 5 shows an example CWT, containing the necessary OSCORE parameters in the 'cnf' claim, in CBOR diagnostic notation without tag and value abbreviations.

```
{
  "aud" : "tempSensorInLivingRoom",
  "iat" : "1360189224",
  "exp" : "1360289224",
  "scope" :  "temperature_g firmware_p",
  "cnf" : {
    "COSE_Key" : {
      "kty" : "Symmetric",
      "alg" : "AES-CCM-16-64-128",
      "clientId" : b64'Qg',
      "serverId" : b64'qA',
      "k" : b64'+a+Dg2jjU+eIiOFCa9lObw'
  }
}
```

Figure 5: Example CWT with OSCORE parameters.

If the client has requested an update to its access rights using the same OSCORE Security Context, and the token associated with it is not expired, the AS MAY omit the master secret and server identifier both in the COSE_Key in the 'cnf' parameter and in the token. The client

identifier needs to be provisioned, in order for the RS to identify the previously generated Security Context.

Figure 6 shows an example of such an AS response, in CBOR diagnostic notation without the tag and value abbreviations.

```
Header: Created (Code=2.01)
Content-Type: "application/cose+cbor"
Payload:
{
  "access_token" : b64'SlAV32hkKG ...
   (remainder of access token omitted for brevity)',
  "profile" : "coap_oscore",
  "expires_in" : "3600",
  "cnf" : {
    "COSE_Key" : {
      "clientId" : b64'qA'
    }
  }
}
```

Figure 6: Example AS-to-C Access Token response with OSCORE profile, for update of access rights.

Figure 7 shows an example CWT, containing the necessary OSCORE parameters in the 'cnf' claim for update of access rights, in CBOR diagnostic notation without tag and value abbreviations.

```
{
  "aud" : "tempSensorInLivingRoom",
  "iat" : "1360189224",
  "exp" : "1360289224",
  "scope" :  "temperature_h",
  "cnf" : {
    "COSE_Key" : {
      "clientId" : b64'Qg'
  }
}
```

Figure 7: Example CWT with OSCORE parameters for update of access rights.

4. Client-RS Communication

The following subsections describe the details of the POST /authz-info request and response between client and RS. The client posts the token that includes the materials provisioned by the AS to the RS, which can then use Section 3.2 of [I-D.ietf-core-object-security] to derive a security context based on a shared master secret and a set of other parameters, established between client and server.

Note that the proof-of-possession required to bind the access token to the client is implicitly performed by generating the shared OSCORE Security Context using the pop-key as master secret, for both client and RS. An attacker using a stolen token will not be able to generate a valid OSCORE context and thus not be able to prove possession of the pop-key.

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

The client MUST use CoAP and the Authorization Information endpoint as described in section 5.8.1 of [I-D.ietf-ace-oauth-authz] to transport the token to the RS.

The authz-info endpoint is not protected, nor are the responses from this endpoint.

The access token MUST be encrypted, since it is transferred from the client to the RS over an unprotected channel.

Figure 8 shows an example of the request sent from the client to the RS.

```
Header: POST (Code=0.02)
Uri-Host: "rs.example.com"
Uri-Path: "authz-info"
Content-Format: "application/cwt"
Payload:
 b64'SlAV32hkKG ...
   (remainder of access token omitted for brevity)',
```

Figure 8: Example C-to-RS POST /authz-info request using CWT

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

The RS MUST follow the procedures defined in section 5.8.1 of [I-D.ietf-ace-oauth-authz]: the RS MUST verify the validity of the token. If the token is valid, the RS MUST respond to the POST request with 2.01 (Created). This response MAY contain an identifier

of the token (e.g., the cti for a CWT) as a payload, in order to allow the client to refer to the token. If the token is valid but is associated to claims that the RS cannot process (e.g., an unknown scope) the RS MUST respond with a response code equivalent to the CoAP code 4.00 (Bad Request). In the latter case the RS MAY provide additional information in the error response, in order to clarify what went wrong. The RS MAY make an introspection request to validate the token before responding to the POST request to the authz-info endpoint.

Additionally, the RS MUST generate a nonce (N1) with a good amount of randomness, and include it in the payload of the 2.01 (Created) response as a CBOR byte string. This profile RECOMMENDS to use a nonce of 64 bits. The RS MUST store this nonce as long as the access token related to it is still valid.

Figure 9 shows an example of the response sent from the RS to the client.

```
Header: Created (Code=2.01)
Content-Format: "application/cbor"
Payload:
 h'018a278f7faab55a',
```

Figure 9: Example RS-to-C 2.01 (Created) response

As specified in section 5.8.3 of [I-D.ietf-ace-oauth-authz], the RS MUST notify the client with an error response with code 4.01 (Unauthorized) for any long running request before terminating the session, when the access token expires.

4.3. OSCORE Setup

Once receiving the 2.01 (Created) response from the RS, following the POST /authz-info request, the client MUST extract the nonce N1 from the CBOR byte string in the payload of the response. The client MUST generate itself a nonce (N2) with a good amount of randomness. This profile RECOMMENDS to use a nonce of 64 bits. Then, the client MUST set the ID Context of the Security Context created to communicate with the RS to the concatenation of N1 and N2, in this order: ID Context = N1 | N2, where | denotes byte string concatenation. The client MUST set the Master Secret, Sender ID and Recipient ID from the parameters received from the AS in Section 3.2. The client MUST set the AEAD Algorithm, Master Salt, HKDF and Replay Window from the parameters received from the AS in Section 3.2, if present. In case these parameters are omitted, the default values are used as described in section 3.2 of [I-D.ietf-core-object-security]. After

that, the client MUST derive the complete Security Context following section 3.2.1 of [I-D.ietf-core-object-security]. From this point on, the client MUST use this Security Context to communicate with the RS when accessing the resources as specified by the authorization information.

The client then uses this Security Context to send requests to RS using OSCORE. In the first request sent to the RS, the client MUST include the kid context, with value ID Context, i.e. N1 concatenated with N2. The client needs to make sure the RS receives the kid context, possibly adding the kid context to later requests, until it receives a valid OSCORE response from the RS using the same Security Context.

When the RS receives this first OSCORE-protected request, it MUST extract the kid context from the message first. Then, it needs to verify that the first part of the kid context corresponds to the nonce N1 it previously sent, and that it is followed by a non-zero-length byte string. If that is verified, the RS MUST set the ID Context to the kid context value. Then, the RS MUST set the Master Secret, Sender ID and Recipient ID from the parameters received from the client in the access token in Section 4.1. The RS MUST set the AEAD Algorithm, Master Salt, HKDF and Replay Window from the parameters received from the client in the access token in Section 4.1, if present. In case these parameters are omitted, the default values are used as described in section 3.2 of [I-D.ietf-core-object-security]. After that, the RS MUST derive the complete Security Context following section 3.2.1 of [I-D.ietf-core-object-security], and MUST associate this Security Context with the authorization information from the access token. Then, the RS MUST delete the nonce N1 from memory.

The RS then uses this Security Context to verify the request and send responses to RS using OSCORE. If OSCORE verification fails, error responses are used, as specified in section 8 of [I-D.ietf-core-object-security]. Additionally, if OSCORE verification succeeds, the verification of access rights is performed as described in section Section 4.4. The RS MUST NOT use the Security Context after the related token has expired, and MUST respond with a unprotected 4.01 (Unauthorized) error message.

4.4. Access rights verification

The RS MUST follow the procedures defined in section 5.8.2 of [I-D.ietf-ace-oauth-authz]: if an RS receives a OSCORE-protected request from a client, then it processes according to [I-D.ietf-core-object-security]. If OSCORE verification succeeds, and the target resource requires authorization, the RS retrieves the

authorization information from the access token associated to the Security Context. The RS then MUST verify that the authorization information covers the resource and the action requested.

The response code MUST be 4.01 (Unauthorized) in case the client has not used the Security Context associated with the access token, or if RS has no valid access token for the client. If RS has an access token for the client but not for the resource that was requested, RS MUST reject the request with a 4.03 (Forbidden). If RS has an access token for the client but it does not cover the action that was requested on the resource, RS MUST reject the request with a 4.05 (Method Not Allowed).

5. Secure Communication with AS

As specified in the ACE framework (section 5.7 of [I-D.ietf-ace-oauth-authz]), the requesting entity (RS and/or client) and the AS communicates via the introspection or token endpoint. The use of CoAP and OSCORE for this communication is RECOMMENDED in this profile, other protocols (such as HTTP and DTLS or TLS) MAY be used instead.

If OSCORE is used, the requesting entity and the AS are expected to have pre-established security contexts in place. How these security contexts are established is out of scope for this profile. Furthermore the requesting entity and the AS communicate using OSCORE ([I-D.ietf-core-object-security]) through the introspection endpoint as specified in section 5.7 of [I-D.ietf-ace-oauth-authz] and through the token endpoint as specified in section 5.6 of [I-D.ietf-ace-oauth-authz].

6. Security Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [I-D.ietf-ace-oauth-authz]. Thus the general security considerations from the framework also apply to this profile.

Furthermore the general security considerations of OSCORE [I-D.ietf-core-object-security] also apply to this specific use of the OSCORE protocol.

OSCORE is designed to secure point-to-point communication, providing a secure binding between the request and the response(s). Thus the basic OSCORE protocol is not intended for use in point-to-multipoint communication (e.g. multicast, publish-subscribe). Implementers of this profile should make sure that their usecase corresponds to the

expected use of OSCORE, to prevent weakening the security assurances provided by OSCORE.

TODO: explain the rationale for the nonces construction, and the security implications for Man-in-the-Middle attacks.

7. Privacy Considerations

TBD.

8. IANA Considerations

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

The following registration is done for the ACE OAuth Profile Registry following the procedure specified in section 8.6 of [I-D.ietf-ace-oauth-authz]:

- Profile name: coap_oscore
- Profile Description: Profile for using OSCORE to secure communication between constrained nodes using the Authentication and Authorization for Constrained Environments framework.
- Profile ID: TBD (value between 1 and 255)
- Change Controller: IESG
- Specification Document(s): [[this specification]]

The following registrations are done for the COSE Key Common Parameter Registry specified in section 16.5 of [RFC8152]:

- Name: clientId
- Label: TBD1 (value between 1 and 255)
- CBOR Type: bstr
- Value Registry: N/A
- Description: Identifies the client in an OSCORE context
- Reference: [[this specification]]

- Name: serverId
- Label: TBD2 (value between 1 and 255)
- Value Type: bstr
- Value Registry: N/A
- Description: Identifies the server in an OSCORE context
- Reference: [[this specification]]

- Name: hkdf
- Label: TBD3 (value between 1 and 255)
- Value Type: bstr

- Value Registry: COSE Algorithms registry
- Description: Identifies the KDF algorithm to be used in an OSCORE context
- Reference: [[this specification]]

- Name: slt
- Label: TBD4 (value between 1 and 255)
- Value Type: bstr
- Value Registry: N/A
- Description: Identifies the master salt of to be used in an OSCORE context
- Reference: [[this specification]]

9. References

9.1. Normative References

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-15 (work in progress), September 2018.

[I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", draft-ietf-core-object-security-15 (work in progress), August 2018.

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

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

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

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

9.2. Informative References

[I-D.gerdes-ace-dcaf-authorize]
Gerdes, S., Bergmann, O., and C. Bormann, "Delegated CoAP Authentication and Authorization Framework (DCAF)", draft-gerdes-ace-dcaf-authorize-04 (work in progress), October 2015.

[I-D.selander-ace-cose-ecdhe]
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", draft-selander-ace-cose-ecdhe-10 (work in progress), September 2018.

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

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

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

Appendix A. Profile Requirements

This section lists the specifications on this profile based on the requirements on the framework, as requested in Appendix C of [I-D.ietf-ace-oauth-authz].

- (Optional) discovery process of how the client finds the right AS for an RS it wants to send a request to: Not specified
- communication protocol the client and the RS must use: CoAP
- security protocol the client and RS must use: OSCORE
- how the client and the RS mutually authenticate: Implicitly by possession of a common OSCORE security context
- Content-format of the protocol messages: "application/cose+cbor"
- proof-of-possession protocol(s) and how to select one; which key types (e.g. symmetric/asymmetric) supported: OSCORE algorithms; pre-established symmetric keys
- profile identifier: coap_oscore

- (Optional) how the RS talks to the AS for introspection: HTTP/CoAP (+ TLS/DTLS/OSCORE)
- how the client talks to the AS for requesting a token: HTTP/CoAP (+ TLS/DTLS/OSCORE)
- how/if the /authz-info endpoint is protected: Security protocol above
- (Optional)other methods of token transport than the /authz-info endpoint: no

Appendix B. Using the pop-key with EDHOC (EDHOC+OSCORE)

EDHOC specifies an authenticated Diffie-Hellman protocol that allows two parties to use CBOR [RFC7049] and COSE in order to establish a shared secret key with perfect forward secrecy. The use of Ephemeral Diffie-Hellman Over COSE (EDHOC) [I-D.selander-ace-cose-ecdhe] in this profile in addition to OSCORE, provides perfect forward secrecy (PFS) and the initial proof-of-possession, which ties the proof-of-possession key to an OSCORE security context.

If EDHOC is used together with OSCORE, and the pop-key (symmetric or asymmetric) is used to authenticate the messages in EDHOC, then the AS MUST provision the following data, in response to the access token request:

- a symmetric or public key (associated to the RS)
- a key identifier;

How these parameters are communicated depends on the type of key (asymmetric or symmetric). Moreover, the AS MUST signal the use of OSCORE + EDHOC with the 'profile' parameter set to "coap_oscore_edhoc" and follow Appendix B to derive the security context to run OSCORE.

Note that in the case described in this section, the 'expires_in' parameter, defined in Section 4.2.2. of [RFC6749] defines the lifetime in seconds of both the access token and the shared secret. After expiration, C MUST acquire a new access token from the AS, and run EDHOC again, as specified in this section

B.1. Using Asymmetric Keys

In case of an asymmetric key, C MUST communicate its own asymmetric key to the AS in the 'cnf' parameter of the access token request, as specified in Section 5.6.1 of [I-D.ietf-ace-oauth-authz]; the AS MUST communicate the RS's public key to C in the response, in the 'rs_cnf' parameter, as specified in Section 5.6.1 of [I-D.ietf-ace-oauth-authz]. Note that the RS's public key MUST include a 'kid' parameter, and that the value of the 'kid' MUST be

included in the access token, to let the RS know which of its public keys C used. If the access token is a CWT [RFC8392], the key identifier MUST be placed directly in the 'cnf' structure (if the key is only referenced).

Figure 3 shows an example of such a request in CBOR diagnostic notation without tag and value abbreviations.

```
Header: POST (Code=0.02)
Uri-Host: "server.example.com"
Uri-Path: "token"
Content-Type: "application/cose+cbor"
Payload:
{
 "grant_type" : "client_credentials",
 "cnf" : {
   "COSE_Key" : {
     "kid" : "client_key"
     "kty" : "EC",
     "crv" : "P-256",
     "x" : b64'usWxHK2PmfnHKwXPS54m0kTcGJ90UiglWiGahtagnv8',
     "y" : b64'IBOL+C3BttVivg+lSreASjpkttcsz+1rb7btKLv8EX4'
   }
 }
}
```

Figure 3: Example access token request (OSCORE+EDHOC, asymmetric).

Figure 4 shows an example of a corresponding response in CBOR diagnostic notation without tag and value abbreviations.

```
Header: Created (Code=2.01)
Content-Type: "application/cose+cbor"
Payload:
{
  "access_token" : b64'SlAV32hkKG ...
   (contains "kid" : "client_key")',
  "profile" : "coap_oscore_edhoc",
  "expires_in" : "3600",
  "cnf" : {
    "COSE_Key" : {
      "kid" : "server_key"
      "kty" : "EC",
      "crv" : "P-256",
      "x" : b64'cGJ90UiglWiGahtagnv8usWxHK2PmfnHKwXPS54m0kT',
      "y" : b64'reASjpkttcsz+1rb7btKLv8EX4IBOL+C3BttVivg+lS'
   }
  }
}
```

Figure 4: Example AS response (EDHOC+OSCORE, asymmetric).

B.2. Using Symmetric Keys

In the case of a symmetric key, the AS MUST communicate the key to the client in the 'cnf' parameter of the access token response, as specified in Section 5.6.2. of [I-D.ietf-ace-oauth-authz]. AS MUST also select a key identifier, that MUST be included as the 'kid' parameter either directly in the 'cnf' structure, as in figure 4 of [I-D.ietf-ace-oauth-authz], or as the 'kid' parameter of the COSE_key, as in figure 6 of [I-D.ietf-ace-oauth-authz].

Figure 5 shows an example of the necessary parameters in the AS response to the access token request when EDHOC is used. The example uses CBOR diagnostic notation without tag and value abbreviations.

```
  Header: Created (Code=2.01)
  Content-Type: "application/cose+cbor"
  Payload:
  {
    "access_token" : b64'SlAV32hkKG ...
     (remainder of access token omitted for brevity)',
    "profile" : "coap_oscore_edhoc",
    "expires_in" : "3600",
    "cnf" : {
      "COSE_Key" : {
        "kty" : "Symmetric",
        "kid" : b64'5tOS+h42dkw',
        "k" : b64'+a+Dg2jjU+eIiOFCa9lObw'
      }
    }
  }
```

Figure 5: Example AS response (EDHOC+OSCORE, symmetric).

In both cases, the AS MUST also include the same key identifier as 'kid' parameter in the access token metadata. If the access token is a CWT [RFC8392], the key identifier MUST be placed inside the 'cnf' claim as 'kid' parameter of the COSE_Key or directly in the 'cnf' structure (if the key is only referenced).

Figure 6 shows an example CWT containing the necessary EDHOC+OSCORE parameters in the 'cnf' claim, in CBOR diagnostic notation without tag and value abbreviations.

```
{
  "aud" : "tempSensorInLivingRoom",
  "iat" : "1360189224",
  "exp" : "1360289224",
  "scope" :  "temperature_g firmware_p",
  "cnf" : {
    "COSE_Key" : {
      "kty" : "Symmetric",
      "kid" : b64'5tOS+h42dkw',
      "k" : b64'+a+Dg2jjU+eIiOFCa9lObw'
  }
}
```

Figure 6: Example CWT with EDHOC+OSCORE, symmetric case.

All other parameters defining OSCORE security context are derived from EDHOC message exchange, including the master secret (see Appendix C.2 of [I-D.selander-ace-cose-ecdhe]).

B.3. Processing

To provide forward secrecy and mutual authentication in the case of pre-shared keys, pre-established raw public keys or with X.509 certificates it is RECOMMENDED to use EDHOC [I-D.selander-ace-cose-ecdhe] to generate the keying material. EDHOC MUST be used as defined in Appendix C of [I-D.selander-ace-cose-ecdhe], with the following additions and modifications.

The first EDHOC message is sent after the access token is posted to the /authz-info resource of the RS as specified in Section 5.8.1 of [I-D.ietf-ace-oauth-authz]. Then the EDHOC message_1 is sent and the EDHOC protocol is initiated [I-D.selander-ace-cose-ecdhe]).

Before the RS continues with the EDHOC protocol and responds to this token submission request, additional verifications on the access token are done: the RS SHALL process the access token according to [I-D.ietf-ace-oauth-authz]. If the token is valid then the RS continues processing EDHOC following Appendix C of [I-D.selander-ace-cose-ecdhe], otherwise it discontinues EDHOC and responds with the error code as specified in [I-D.ietf-ace-oauth-authz].

- In case the EDHOC verification fails, the RS MUST return an error response to the client with code 4.01 (Unauthorized).
- If RS has an access token for C but not for the resource that C has requested, RS MUST reject the request with a 4.03 (Forbidden).
- If RS has an access token for C but it does not cover the action C requested on the resource, RS MUST reject the request with a 4.05 (Method Not Allowed).
- If all verifications above succeeds, further communication between client and RS is protected with OSCORE, including the RS response to the OSCORE request.

In the case of EDHOC being used with symmetric keys, the protocol in Section 5 of [I-D.selander-ace-cose-ecdhe] MUST be used. If the key is asymmetric, the RS MUST also use an asymmetric key for authentication. This key is known to the client through the access token response (see Section 5.6.2 of [I-D.ietf-ace-oauth-authz]). In this case the protocol in Section 4 of [I-D.selander-ace-cose-ecdhe] MUST be used.

Figure 7 illustrates the message exchanges for using OSCORE+EDHOC (step C in figure 1 of [I-D.ietf-ace-oauth-authz]).

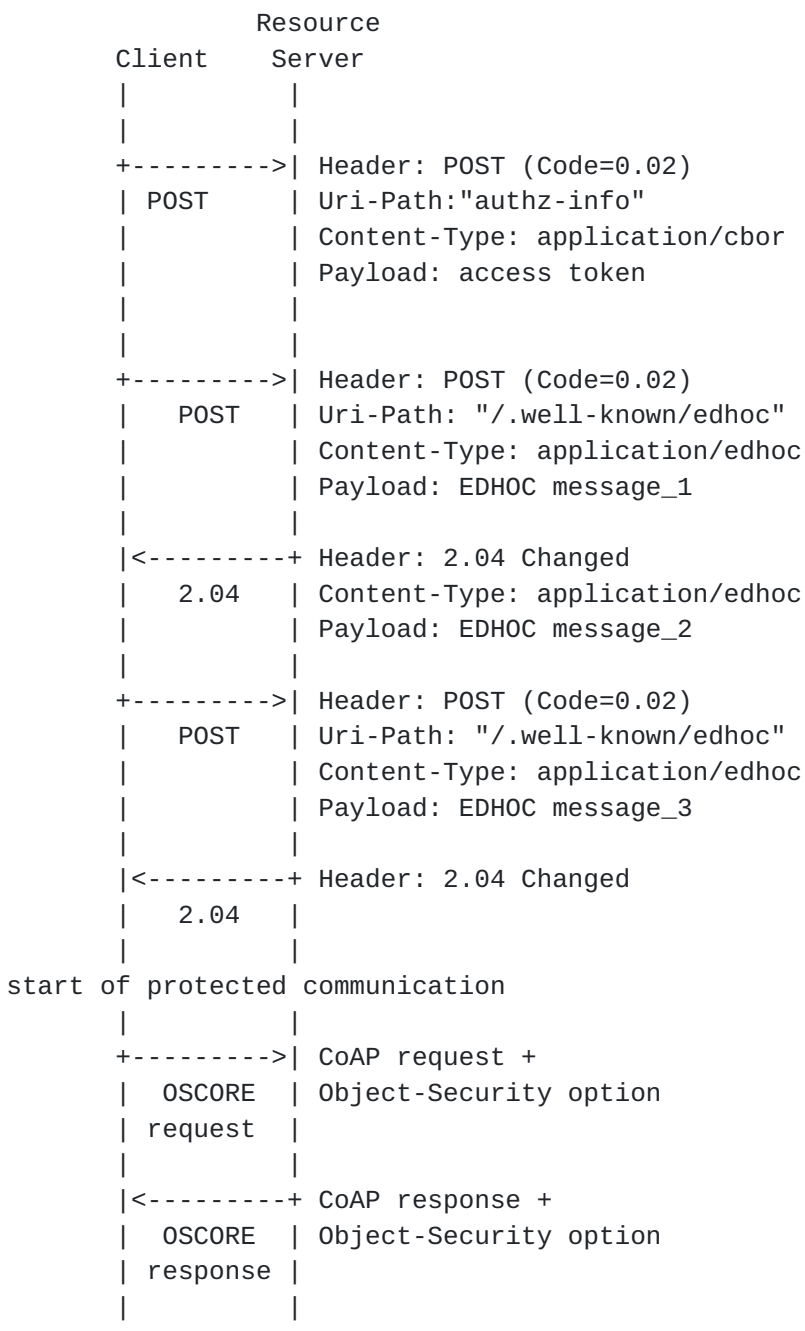

Figure 7: Access token and key establishment with EDHOC

Acknowledgments

The authors wish to thank Jim Schaad and Marco Tiloca for the input on this memo. The error responses specified in Appendix B.3 were originally specified by Gerdes et al. in [I-D.gerdes-ace-dcaf-authorize].

Authors' Addresses

Ludwig Seitz
RISE SICS AB
Scheelevagen 17
Lund 22370
Sweden

Email: ludwig.seitz@ri.se

Francesca Palombini
Ericsson AB
Farogatan 6
Kista SE-16480 Stockholm
Sweden

Email: francesca.palombini@ericsson.com

Martin Gunnarsson
RISE SICS AB
Scheelevagen 17
Lund 22370
Sweden

Email: martin.gunnarsson@ri.se

Goeran Selander
Ericsson AB
Farogatan 6
Kista SE-16480 Stockholm
Sweden

Email: goran.selander@ericsson.com