

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 7, 2019

F. Palombini
Ericsson AB
L. Seitz
RISE SICS AB
G. Selander
Ericsson AB
M. Gunnarsson
RISE SICS AB
January 3, 2019

**OSCORE profile of the Authentication and Authorization for Constrained
Environments Framework
draft-ietf-ace-oscore-profile-06**

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.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 7, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
1.1.	Terminology	3
2.	Protocol Overview	3
3.	Client-AS Communication	5
3.1.	C-to-AS: POST to token endpoint	6
3.2.	AS-to-C: Access Token	7
3.2.1.	OSCORE_Security_Context Object	11
4.	Client-RS Communication	14
4.1.	C-to-RS: POST to authz-info endpoint	15
4.2.	RS-to-C: 2.01 (Created)	16
4.3.	OSCORE Setup	17
4.4.	Access rights verification	18
5.	Secure Communication with AS	19
6.	Discarding the Security Context	19
7.	Security Considerations	20
8.	Privacy Considerations	21
9.	IANA Considerations	21
9.1.	ACE OAuth Profile Registry	21
9.2.	OSCORE Security Context Parameters Registry	22
9.3.	CWT Confirmation Methods Registry	22
9.4.	JWT Confirmation Methods Registry	23
9.5.	Expert Review Instructions	23
10.	References	24
10.1.	Normative References	24
10.2.	Informative References	24
Appendix A.	Profile Requirements	25
	Acknowledgments	26
	Authors' Addresses	26

[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.

Note that the term "endpoint" is used here, as in [[I-D.ietf-ace-oauth-authz](#)], following its OAuth definition, which is to denote resources such as token and introspect at the AS and authz-info at the RS. The CoAP [[RFC7252](#)] definition, which is "An entity participating in the CoAP protocol" is not used in this memo.

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 by the client to negotiate the use of this profile with the authorization server, as well as OSCORE setup process, 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 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 generates a nonce N1 and posts both the token and N1 to the RS using the authz-info endpoint and mechanisms specified in section 5.8 of [\[I-D.ietf-ace-oauth-authz\]](#) and Content-Format = application/ace+cbor.

If the access token is valid, the RS replies to this request with a 2.01 (Created) response with Content-Format = application/ace+cbor, which contains a nonce N2 in a CBOR map. Moreover, the server concatenates N1 with N2 and sets the ID Context in the Security Context (see section 3 of [\[I-D.ietf-core-object-security\]](#)) to N1 concatenated with N2. The RS then derives the complete Security Context associated with the received token from it plus the parameters received in the AS, following section 3.2 of [\[I-D.ietf-core-object-security\]](#).

After receiving the nonce N2, the client 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 may contain the ID Context value. 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. This Security Context is discarded if the same token is re-used to successfully derive a new Security Context. Once the client receives a valid response, it does not continue to include the ID Context value in further requests.

The use of random nonces during the exchange prevents the reuse of AEAD nonces and keys with different messages, in case of re-derivation of the Security Context both for Clients and Resource Servers from an old non-expired access token, e.g. in case of re-boot of either the client or RS. In fact, by using random nonces as ID Context, the request to the authz-info endpoint posting the same token results in a different Security Context, since Master Secret, Sender ID and Recipient ID are the same but ID Context is different. Therefore, the main requirement for the nonces is that they have a

good amount of randomness. If random nonces were not used, a node re-using a non-expired old token would be susceptible to on-path attackers provoking the creation of OSCORE messages using old AEAD keys and nonces.

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

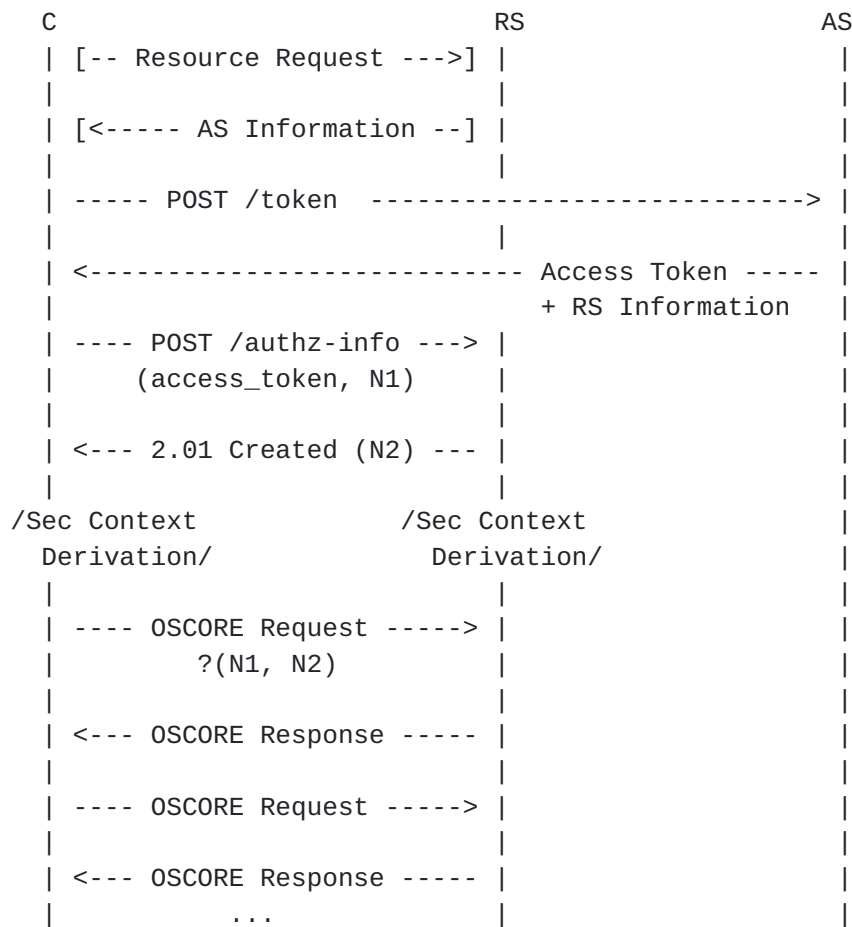


Figure 1: Protocol Overview

3. Client-AS Communication

The following subsections describe the details of the POST request and response to the token endpoint 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 to token endpoint

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

The client MUST send this POST request to the token endpoint 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:
{
  "req_aud" : "tempSensor4711",
  "scope" : "read"
}
```

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

If the client wants to update its access rights without changing an existing OSCORE Security Context, it MUST include in its POST request to the token endpoint a req_cnf object carrying the client's identifier (that was assigned in section [Section 3.2](#)) 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 identify a symmetric key that was previously generated by the AS as a shared secret for the communication between the client and the RS. The AS MUST verify that the received value identifies a proof-of-possession key and token that have previously been issued to the requesting client. If that is not the case, the Client-to-AS request MUST be declined with the error code 'invalid_request' as defined in Section 5.6.3 of [\[I-D.ietf-ace-oauth-authz\]](#).

An example of such a request, in CBOR diagnostic notation without the tag and value abbreviations is reported in Figure 3


```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
{
  "req_aud" : "tempSensor4711",
  "scope" : "write",
  "req_cnf" : {
    "kid" : "myclient"
  }
}
```

Figure 3: Example C-to-AS POST /token request for updating rights to an access token bound to a symmetric key.

3.2. AS-to-C: Access Token

After verifying the POST request to the token endpoint 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](#)]. If the client request was invalid, or not authorized, the AS returns an error response as described in section 5.6.3 of [[I-D.ietf-ace-oauth-authz](#)].

The AS can signal 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. Usually it is assumed that constrained devices will be pre-configured with the necessary profile, so that this kind of profile negotiation can be omitted.

Moreover, the AS MUST provision the following data:

- o a master secret
- o a server identifier

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

- o a client identifier
- o an AEAD algorithm
- o an HKDF algorithm

- o a salt
- o a replay window type and size

The master secret MUST be communicated as the 'ms' field in the OSCORE_Security_Context in the 'cnf' parameter of the access token response as defined in Section 3.2 of [[I-D.ietf-ace-oauth-params](#)]. The OSCORE_Security_Context is a CBOR map object, defined in [Section 3.2.1](#). The AEAD algorithm MAY be included as the 'alg' parameter in the OSCORE_Security_Context; the HKDF algorithm MAY be included as the 'hkdf' parameter of the OSCORE_Security_Context, the salt MAY be included as the 'salt' parameter of the OSCORE_Security_Context, and the replay window type and size MAY be included as the 'rpl' of the OSCORE_Security_Context, as defined in [Section 3.2.1](#).

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 OSCORE_Security_Context structure defined above MUST be placed in the 'cnf' claim of this token.

The AS MUST also assign an identifier to the RS, and MAY assign an identifier to the client. These identifiers 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](#)]. The client identifiers MUST be unique in the set of all clients on a single RS, and RS identifiers MUST be unique in the set of all RS for any given client. Moreover, when assigned, these MUST be included in the OSCORE_Security_Context, as defined in [Section 3.2.1](#).

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 in [Section 4.3](#) C sets the Sender ID of its Security Context to the clientId value received and the Recipient ID to the serverId value, and RS does the opposite.

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/ace+cbor"
Payload:
{
  "access_token" : h'a5037674656d7053656e73 ...'
    (remainder of access token omitted for brevity)',
  "profile" : "coap_oscure",
  "expires_in" : "3600",
  "cnf" : {
    "OSCORE_Security_Context" : {
      "alg" : "AES-CCM-16-64-128",
      "clientId" : b64'qA',
      "serverId" : b64'Qg',
      "ms" : h'f9af838368e353e78888e1426bd94e6f'
    }
  }
}

```

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" : {
    "OSCORE_Security_Context" : {
      "alg" : "AES-CCM-16-64-128",
      "clientId" : "client",
      "serverId" : "server",
      "ms" : h'f9af838368e353e78888e1426bd94e6f'
    }
  }
}

```

Figure 5: Example CWT with OSCORE parameters.

The same CWT token as in Figure 5, using the value abbreviations defined in [[I-D.ietf-ace-oauth-authz](#)] and [[I-D.ietf-ace-cwt-proof-of-possession](#)] and encoded in CBOR is shown in Figure 6.

NOTE TO THE RFC EDITOR: before publishing, it should be checked (and in case fixed) that the values used below (which are not yet registered) are the final values registered in IANA.

```

A5                                     # map(5)
  03                                 # unsigned(3)
  76                                 # text(22)
    74656D7053656E736F72496E4C6976696E67526F6F6D
                                     # "tempSensorInLivingRoom"
  06                                 # unsigned(6)
  1A 5112D728                       # unsigned(1360189224)
  04                                 # unsigned(4)
  1A 51145DC8                       # unsigned(1360289224)
  09                                 # unsigned(9)
  78 18                             # text(24)
    74656D70657261747572655F67206669726D776172655F70
                                     # "temperature_g firmware_p"
  08                                 # unsigned(8)
  A1                                 # map(1)
    04                              # unsigned(4)
    A4                              # map(4)
      05                          # unsigned(5)
      0A                          # unsigned(10)
      02                          # unsigned(2)
      66                          # text(6)
        636C69656E74             # "client"
      03                          # unsigned(3)
      66                          # text(6)
        736572766572             # "server"
      01                          # unsigned(1)
      50                          # bytes(16)
        F9AF838368E353E78888E1426BD94E6F

```

Figure 6: Example CWT with OSCORE parameters.

If the client has requested an update to its access rights using the same OSCORE Security Context, which is valid and authorized, the AS MUST omit the 'cnf' parameter in the response, and MUST carry the client identifier in the 'kid' field in the 'cnf' parameter of the token. The client identifier needs to be provisioned, in order for the RS to identify the previously generated Security Context.

Figure 7 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/ace+cbor"
Payload:
{
  "access_token" : h'a5037674656d7053656e73 ...'
    (remainder of access token omitted for brevity)',
  "profile" : "coap_oscore",
  "expires_in" : "3600"
}
```

Figure 7: Example AS-to-C Access Token response with OSCORE profile, for update of access rights.

Figure 8 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" : {
    "kid" : b64'Qg'
  }
}
```

Figure 8: Example CWT with OSCORE parameters for update of access rights.

3.2.1. OSCORE_Security_Context Object

An OSCORE_Security_Context is an object that represents part or all of an OSCORE Security Context (Section 3.1 of [\[I-D.ietf-core-object-security\]](#)). The OSCORE_Security_Context object can either be encoded as JSON or as CBOR. In both cases, the set of common parameters that can appear in an OSCORE_Security_Context object can be found in the IANA "OSCORE Security Context Parameters" registry (Section [Section 9.2](#)) and is defined below. All parameters are optional. Table 1 provides a summary of the OSCORE_Security_Context parameters defined in this section.

name	CBOR label	CBOR type	registry	description
ms	1	bstr		OSCORE Master Secret value
clientId	2	bstr		OSCORE Sender ID value of the client, OSCORE Recipient ID value of the server
serverId	3	bstr		OSCORE Sender ID value of the server, OSCORE Recipient ID value of the client
hkdf	4	bstr / int	COSE Algorithm Values (HMAC-based)	OSCORE HKDF value
alg	5	tstr / int	COSE Algorithm Values (AEAD)	OSCORE AEAD Algorithm value
salt	6	bstr		OSCORE Master Salt value
contextId	7	bstr		OSCORE ID Context value
rpl	8	bstr / int		OSCORE Replay Window Type and Size

Table 1: OSCORE_Security_Context Parameters

ms: This parameter identifies the OSCORE Master Secret value, which is a byte string. For more information about this field, see

section 3.1 of [[I-D.ietf-core-object-security](#)]. In JSON, the "ms" value is a Base64 encoded byte string. In CBOR, the "ms" type is bstr, and has label 1.

clientId: This parameter identifies a client identifier as a byte string. This identifier is used as OSCORE Sender ID in the client and OSCORE Recipient ID in the server. For more information about this field, see section 3.1 of [[I-D.ietf-core-object-security](#)]. In JSON, the "clientId" value is a Base64 encoded byte string. In CBOR, the "clientId" type is bstr, and has label 2.

serverId: This parameter identifies a server identifier as a byte string. This identifier is used as OSCORE Sender ID in the client and OSCORE Recipient ID in the server. For more information about this field, see section 3.1 of [[I-D.ietf-core-object-security](#)]. In JSON, the "serverId" value is a Base64 encoded byte string. In CBOR, the "serverId" type is bstr, and has label 3.

hkdf: This parameter identifies the OSCORE HKDF Algorithm. For more information about this field, see section 3.1 of [[I-D.ietf-core-object-security](#)]. The values used MUST be registered in the IANA "COSE Algorithms" registry and MUST be HMAC-based HKDF algorithms. The value can either be the integer or the text string value of the HMAC-based HKDF algorithm in the "COSE Algorithms" registry. In JSON, the "hkdf" value is a case-sensitive ASCII string or an integer. In CBOR, the "hkdf" type is tstr or int, and has label 4.

alg: This parameter identifies the OSCORE AEAD Algorithm. For more information about this field, see section 3.1 of [[I-D.ietf-core-object-security](#)]. The values used MUST be registered in the IANA "COSE Algorithms" registry and MUST be AEAD algorithms. The value can either be the integer or the text string value of the HMAC-based HKDF algorithm in the "COSE Algorithms" registry. In JSON, the "alg" value is a case-sensitive ASCII string or an integer. In CBOR, the "alg" type is tstr or int, and has label 5.

salt: This parameter identifies the OSCORE Master Salt value, which is a byte string. For more information about this field, see section 3.1 of [[I-D.ietf-core-object-security](#)]. In JSON, the "salt" value is a Base64 encoded byte string. In CBOR, the "salt" type is bstr, and has label 6.

contextId: This parameter identifies the security context as a byte string. This identifier is used as OSCORE ID Context. For more information about this field, see section 3.1 of [[I-D.ietf-core-object-security](#)]. In JSON, the "contextID" value

is a Base64 encoded byte string. In CBOR, the "contextID" type is bstr, and has label 7.

repl: This parameter is used to carry the OSCORE value, encoded as a bstr. This parameter identifies the OSCORE Replay Window Size and Type value, which is a byte string. For more information about this field, see section 3.1 of [[I-D.ietf-core-object-security](#)]. In JSON, the "repl" value is a Base64 encoded byte string. In CBOR, the "repl" type is bstr, and has label 8.

An example of JSON OSCORE_Security_Context is given in Figure 9.

```
"OSCORE_Security_Context" : {
  "alg" : "AES-CCM-16-64-128",
  "clientId" : b64'qA',
  "serverId" : b64'Qg',
  "ms" : b64'+a+Dg2jjU+eIi0FCa9l0bw'
}
```

Figure 9: Example JSON OSCORE_Security_Context object

The CDDL grammar describing the CBOR OSCORE_Security_Context object is:

```
OSCORE_Security_Context = {
  ? 1 => bstr,           ; ms
  ? 2 => bstr,           ; clientId
  ? 3 => bstr,           ; serverId
  ? 4 => tstr / int,      ; hkdf
  ? 5 => tstr / int,      ; alg
  ? 6 => bstr,           ; salt
  ? 7 => bstr,           ; contextId
  ? 8 => bstr / tstr,     ; repl
  * int / tstr => any
}
```

4. Client-RS Communication

The following subsections describe the details of the POST request and response to the authz-info endpoint between client and RS. The client generates a nonce N1 and posts it together with the token that includes the materials provisioned by the AS to the RS. The RS then derives a nonce N2 and use Section 3.2 of [[I-D.ietf-core-object-security](#)] to derive a security context based on a shared master secret and the two nonces, 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 to authz-info endpoint

The client MUST generate a nonce N1 very unlikely to have been previously used with the same input keying material. This profile RECOMMENDS to use a 64-bit long random number as nonce. The client MUST store this nonce as long as the response from the RS is not received and the access token related to it is still valid. The client MUST use CoAP and the Authorization Information resource as described in section 5.8.1 of [[I-D.ietf-ace-oauth-authz](#)] to transport the token and N1 to the RS. The client MUST use the Content-Format "application/ace+cbor" defined in section 8.14 of [[I-D.ietf-ace-oauth-authz](#)]. The client MUST include the access token using the correct CBOR label (e.g., "cwt" for CWT, "jwt" for JWT) and N1 using the 'nonce' parameter defined in section 5.1.2 of [[I-D.ietf-ace-oauth-authz](#)].

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

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

Note that a client may be required to re-POST the access token, since an RS may delete a stored access token, due to lack of memory.

Figure 10 shows an example of the request sent from the client to the RS, in CBOR diagnostic notation without the tag and value abbreviations.


```
Header: POST (Code=0.02)
Uri-Host: "rs.example.com"
Uri-Path: "authz-info"
Content-Format: "application/ace+cbor"
Payload:
{
  "access_token": h'a5037674656d7053656e73 ...'
  (remainder of access token omitted for brevity)',
  "nonce": h'018a278f7faab55a'
}
```

Figure 10: 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). If the token is valid but is associated to claims that the RS cannot process (e.g., an unknown scope), or if any of the expected parameters in the OSCORE_Security_Context is missing (e.g. any of the mandatory parameters from the AS), 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 N2 very unlikely to have been previously used with the same input keying material, and send it within the 2.01 (Created) response. The payload of the 2.01 (Created) response MUST be a CBOR map containing the 'nonce' parameter defined in section 5.1.2 of [\[I-D.ietf-ace-oauth-authz\]](#), set to N2. This profile RECOMMENDS to use a 64-bit long random number as nonce. Moreover, if the OSCORE_Security_Context in the token did not contain a 'clientId' parameter, the RS MUST generate an identifier, unique in the set of all its existing client identifiers, and send it in a 'clientId' parameter in the CBOR map as a CBOR bstr. The RS MAY generate and send a 'ClientId' identifier even though the OSCORE_Security_Context contained such a parameter, in order to guarantee the uniqueness of the client identifier. The RS MUST use the Content-Format "application/ace+cbor" defined in section 8.14 of [\[I-D.ietf-ace-oauth-authz\]](#).

Figure 11 shows an example of the response sent from the RS to the client, in CBOR diagnostic notation without the tag and value abbreviations.

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

Figure 11: Example RS-to-C 2.01 (Created) response

When receiving an updated access token with updated authorization information from the client (see section [Section 3.1](#)), it is RECOMMENDED that the RS overwrites the previous token, that is only the latest authorization information in the token received by the RS is valid. This simplifies for the RS to keep track of authorization information for a given client.

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 request to authz-info endpoint, the client MUST extract the nonce N2 from the 'nonce' parameter and the client identifier from the 'clientId' in the CBOR map in the payload of the response. 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 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](#)]. The client MUST set the Sender ID from the 'clientId' in the 2.01 (Created) response, if present; otherwise, the client MUST set the Sender ID from the parameters received from the AS in [Section 3.2](#). 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.

If any of the expected parameters is missing (e.g. any of the mandatory parameters from the AS, or the 'clientId', either received from the AS or in the 2.01 (Created) response from the RS), the client MUST stop the exchange, and MUST NOT derive the Security Context. The client MAY restart the exchange, to get the correct security material.

The client then uses this Security Context to send requests to RS using OSCORE. In the first request sent to the RS, the client MAY include the kid context, with value ID Context, i.e. N1 concatenated with N2.

After sending the 2.01 (Created) response, the RS MUST set the ID Context of the Security Context created to communicate with the client to the concatenation of N1 and N2, in this order: ID Context = N1 | N2, where | denotes byte string concatenation. 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](#) after validation of the token as specified in [Section 4.2](#). 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](#) after validation of the token as specified in [Section 4.2](#), 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 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.

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 an OSCORE-protected request from a client, then the RS processes it 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. Discarding the Security Context

There are a number of scenarios where a client or RS needs to discard the OSCORE security context, and acquire a new one.

The client MUST discard the current security context associated with an RS when:

- o the Sequence Number space ends.
- o the access token associated with the context expires.
- o the client receives a number of 4.01 Unauthorized responses to OSCORE requests using the same security context. The exact number needs to be specified by the application.

- o the client receives a new nonce in the 2.01 (Created) response (see [Section 4.2](#)) to a POST request to the authz-info endpoint, when re-posting a non-expired token associated to the existing context.

The RS MUST discard the current security context associated with a client when:

- o Sequence Number space ends.
- o Access token associated with the context expires.

7. 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.

Since the use of nonces in the exchange guarantees uniqueness of AEAD keys and nonces, it is REQUIRED that nonces are not reused with the same input keying material even in case of re-boots. This document RECOMMENDS the use of 64 bit random nonces to guarantee non-reuse; if applications use something else, such as a counter, they need to guarantee that reboot and lost of state on either node does not provoke re-use. If that is not guaranteed, nodes are still susceptible to re-using AEAD nonces and keys, in case the Security Context is lost, and on-path attacker replay messages.

This profiles recommends that the RS maintains a single access token for a client. The use of multiple access tokens for a single client increases the strain on the resource server as it must consider every access token and calculate the actual permissions of the client. Also, tokens may contradict each other which may lead the server 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 client.

8. Privacy 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 privacy considerations from the framework also apply to this profile.

As this document uses OSCORE, thus the privacy considerations from [[I-D.ietf-core-object-security](#)] apply here as well.

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

Note that some information might still leak after OSCORE is established, due to observable message sizes, the source, and the destination addresses.

9. IANA Considerations

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

9.1. ACE OAuth Profile Registry

The following registration is done for the ACE OAuth Profile Registry following the procedure specified in section 8.7 of [[I-D.ietf-ace-oauth-authz](#)]:

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

9.2. OSCORE Security Context Parameters Registry

It is requested that IANA create a new registry entitled "OSCORE Security Context Parameters" registry. The registry is to be created as Expert Review Required. Guidelines for the experts is provided [Section 9.5](#). 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 The JSON name requested (e.g., "ms"). Because a core goal of this specification 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 state that there is a compelling reason to allow an exception. The name is not used in the CBOR encoding.

CBOR label The value to be used to identify this algorithm. Key map labels MUST be unique. The label can be a positive integer, a negative integer or a string. Integer values between 0 and 255 and strings of length 1 are designated as Standards Track Document required. Integer values from 256 to 65535 and strings of length 2 are designated as Specification Required. Integer values of greater than 65535 and strings of length greater than 2 are designated as expert review. Integer values less than -65536 are marked as private use.

CBOR Type This field contains the CBOR type for the field.

registry This field denotes the registry that values may come from, if one exists.

description This field contains a brief description for the field.

specification This contains a pointer to the public specification for the field if one exists

This registry will be initially populated by the values in Table 1. The specification column for all of these entries will be this document.

9.3. CWT Confirmation Methods Registry

The following registration is done for the CWT Confirmation Methods Registry following the procedure specified in section 7.2.1 of [\[I-D.ietf-ace-cwt-proof-of-possession\]](#):

- o Confirmation Method Name: "OSCORE_Security_Context"
- o Confirmation Method Description: OSCORE_Security_Context carrying the OSCORE Security Context parameters
- o Confirmation Key: TBD (value between 4 and 255)

- o Confirmation Value Type(s): map
- o Change Controller: IESG
- o Specification Document(s): [Section 3.2.1](#) of [[this specification]]

9.4. JWT Confirmation Methods Registry

The following registration is done for the JWT Confirmation Methods Registry following the procedure specified in [section 6.2.1 of \[RFC7800\]](#):

- o Confirmation Method Value: "osc"
- o Confirmation Method Description: OSCORE_Security_Context carrying the OSCORE Security Context parameters
- o Change Controller: IESG
- o Specification Document(s): [Section 3.2.1](#) of [[this specification]]

9.5. Expert Review Instructions

The IANA registry established in this document is defined as 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:

- o 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.
- o Specifications are required for the standards track 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.
- o 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.

10. References

10.1. 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-17](#) (work in progress), November 2018.
- [I-D.ietf-ace-oauth-params]
Seitz, L., "Additional OAuth Parameters for Authorization in Constrained Environments (ACE)", [draft-ietf-ace-oauth-params-01](#) (work in progress), November 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>>.

10.2. Informative References

[I-D.ietf-ace-cwt-proof-of-possession]

Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", [draft-ietf-ace-cwt-proof-of-possession-05](#) (work in progress), November 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>>.

[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>>.

[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/info/rfc7800>>.

[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 [\[I-D.ietf-ace-oauth-authz\]](#).

- o (Optional) discovery process of how the client finds the right AS for an RS it wants to send a request to: Not specified
- o communication protocol the client and the RS must use: CoAP
- o security protocol the client and RS must use: OSCORE
- o how the client and the RS mutually authenticate: Implicitly by possession of a common OSCORE security context
- o Content-format of the protocol messages: "application/ace+cbor"
- o proof-of-possession protocol(s) and how to select one; which key types (e.g. symmetric/asymmetric) supported: OSCORE algorithms; pre-established symmetric keys
- o profile identifier: coap_oscore
- o (Optional) how the RS talks to the AS for introspection: HTTP/CoAP (+ TLS/DTLS/OSCORE)
- o how the client talks to the AS for requesting a token: HTTP/CoAP (+ TLS/DTLS/OSCORE)
- o how/if the authz-info endpoint is protected: Security protocol above

- o (Optional)other methods of token transport than the authz-info endpoint: no

Acknowledgments

The authors wish to thank Jim Schaad and Marco Tiloca for the input on this memo.

Authors' Addresses

Francesca Palombini
Ericsson AB

Email: francesca.palombini@ericsson.com

Ludwig Seitz
RISE SICS AB
Scheelevagen 17
Lund 22370
Sweden

Email: ludwig.seitz@ri.se

Goeran Selander
Ericsson AB

Email: goran.selander@ericsson.com

Martin Gunnarsson
RISE SICS AB
Scheelevagen 17
Lund 22370
Sweden

Email: martin.gunnarsson@ri.se

