

Workgroup: ACE Working Group

Internet-Draft:

draft-tiloca-ace-group-oscore-profile-11

Published: 10 July 2023

Intended Status: Standards Track

Expires: 11 January 2024

Authors: M. Tiloca R. Höglund L. Seitz F. Palombini
 RISE AB RISE AB Combitech Ericsson AB

The Group Object Security for Constrained RESTful Environments (Group OSCORE) Profile of the Authentication and Authorization for Constrained Environments (ACE) Framework

Abstract

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework. The profile uses Group Object Security for Constrained RESTful Environments (Group OSCORE) to provide communication security between a Client and one or multiple Resource Servers that are members of an OSCORE group. The profile securely binds an OAuth 2.0 Access Token to the public key of the Client associated with the private key used by that Client in the OSCORE group. The profile uses Group OSCORE to achieve server authentication, as well as proof-of-possession for the Client's public key. Also, it provides proof of the Client's membership to the OSCORE group by binding the Access Token to information from the Group OSCORE Security Context, thus allowing the Resource Server(s) to verify the Client's membership upon receiving a message protected with Group OSCORE from the Client. Effectively, the profile enables fine-grained access control paired with secure group communication, in accordance with the Zero Trust principles.

Discussion Venues

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

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (ace@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/ace/>.

Source for this draft and an issue tracker can be found at <https://gitlab.com/crimson84/draft-tiloca-ace-group-oscore-profile>.

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 11 January 2024.

Copyright Notice

Copyright (c) 2023 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

- [1. Introduction](#)
 - [1.1. Terminology](#)
- [2. Protocol Overview](#)
 - [2.1. Pre-Conditions](#)
 - [2.2. Access Token Retrieval](#)
 - [2.3. Access Token Posting](#)
 - [2.4. Secure Communication](#)
- [3. Client-AS Communication](#)
 - [3.1. C-to-AS: POST to Token Endpoint](#)
 - [3.1.1. 'context id' Parameter](#)
 - [3.1.2. 'salt input' Parameter](#)
 - [3.1.3. 'client cred verify' Parameter](#)
 - [3.1.4. 'client cred verify mac' Parameter](#)
 - [3.2. AS-to-C: Access Token](#)
 - [3.2.1. Salt Input Claim](#)
 - [3.2.2. Context ID Input Claim](#)
- [4. Client-RS Communication](#)
 - [4.1. C-to-RS POST to authz-info Endpoint](#)
 - [4.2. RS-to-C: 2.01 \(Created\)](#)

4.3. Client-RS Secure Communication
4.3.1. Client Side
4.3.2. Resource Server Side
4.4. Access Rights Verification
4.5. Change of Client's Authentication Credential in the Group
5. Secure Communication with the AS
6. Discarding the Security Context
7. CBOR Mappings
8. Security Considerations
9. Privacy Considerations
10. IANA Considerations
10.1. ACE Profile Registry
10.2. OAuth Parameters Registry
10.3. OAuth Parameters CBOR Mappings Registry
10.4. CBOR Web Token (CWT) Claims Registry
10.5. TLS Exporter Label Registry
11. References
11.1. Normative References
11.2. Informative References
Appendix A. Profile Requirements
Acknowledgments
Authors' Addresses

1. Introduction

A number of applications rely on a group communication model where a Client can access a resource shared by multiple Resource Servers at once, e.g., over IP multicast. Typical examples are switching of luminaries, actuators control, and distribution of software updates. Secure communication in the group can be achieved by sharing a set of keying material, which is typically provided upon joining the group.

For some of such applications, it may be just fine to enforce access control in a straightforward fashion. That is, any Client authorized to join the group, hence to obtain the group keying material, can be also implicitly authorized to perform any action at any resource of any Server in the group. An example of application where such implicit authorization might be used is a simple lighting scenario, where the lightbulbs are the Servers, while the user account on an app on the user's phone is the Client. In this case, it might be fine to not require additional authorization evidence from any user account, if it is acceptable that any current group member is also authorized to switch on and off any light, or to check their status.

However, in different instances of such applications, the approach above is not desirable, as different group members are intended to have different access rights to resources of other group members.

For instance, enforcing access control in accordance with a more fine-grained approach is required in the two following use cases.

As a first case, an application provides control of smart locks acting as Servers in the group, where: a first type of Client, e.g., a user account of a child, is allowed to only query the status of the smart locks; while a second type of Client, e.g., a user account of a parent, is allowed to both query and change the status of the smart locks. Further similar applications concern the enforcement of different sets of permissions in groups with sensor/actuator devices, e.g., thermostats acting as Servers. Also, some group members may even be intended as Servers only. Hence, they must be prevented from acting as Clients altogether and from accessing resources at other Servers, especially when attempting to perform non-safe operations.

As a second case, building automation scenarios often rely on Servers that, under different circumstances, enforce different level of priority for processing received commands. For instance, BACnet deployments consider multiple classes of Clients, e.g., a normal light switch (C1) and an emergency fire panel (C2). Then, a C1 Client is not allowed to override a command from a C2 Client, until the latter relinquishes control at its higher priority. That is: i) only C2 Clients should be able to adjust the minimum required level of priority on the Servers, so rightly locking out C1 Clients if needed; and ii) when a Server is set to accept only high-priority commands, only C2 Clients should be able to perform such commands otherwise allowed also to C1 Clients. Given the different maximum authority of different Clients, fine-grained access control would effectively limit the execution of high- and emergency-priority commands only to devices that are in fact authorized to do so. Besides, it would prevent a misconfigured or compromised device from initiating a high-priority command and lock out normal control.

In the cases above, being a legitimate group member and storing the group keying material is not supposed to imply any particular access rights. Instead, access control to the secure group communication channel and access control to the resource space provided by Servers in the group should remain logically separated domains.

This is aligned with the Zero Trust paradigm [[NIST-800-207](#)], which focuses on resource protection and builds on the premise that trust is never granted implicitly, but must be continually evaluated. In particular, Zero Trust protections involve "minimizing access to resources (such as data and compute resources and applications/services) to only those subjects and assets identified as needing access as well as continually authenticating and authorizing the identity and security posture of each access request."

Furthermore, [[NIST-800-207](#)] highlights how the Zero Trust goal is to "prevent unauthorized access to data and services coupled with making the access control enforcement as granular as possible", to "enforce least privileges needed to perform the action in the request."

As a step in this direction, one can be tempted to introduce a different security group for each different set of access rights. However, this inconveniently results in additional keying material to distribute and manage. In particular, if the access rights for a single node change, this requires to evict that node from the current group, followed by that node joining a different group aligned with its new access rights. Moreover, the keying material of both groups would have to be renewed for their current members. Overall, this would have a non negligible impact on operations and performance.

Instead, a fine-grained yet flexible access control model can be enforced within the same group, by using the Authentication and Authorization for Constrained Environments (ACE) framework [[RFC9200](#)]. That is, a Client has to first obtain authorization credentials in the form of an Access Token, and post it to the Resource Server(s) in the group before accessing the intended resources.

The ACE framework delegates to separate profile documents how to secure communications between the Client and the Resource Servers. However each of the current profiles of ACE defined in [[RFC9202](#)] [[RFC9203](#)] [[I-D.ietf-ace-mqtt-tls-profile](#)] relies on a security protocol that cannot be used to protect one-to-many group messages, for example sent over IP multicast.

This document specifies the "coap_group_oscure" profile of the ACE framework, where a Client uses the Constrained Application Protocol (CoAP) [[RFC7252](#)] [[I-D.ietf-core-groupcomm-bis](#)] to communicate with one or multiple Resource Servers that are members of an application group and share a common set of resources. This profile uses Group Object Security for Constrained RESTful Environments (Group OSCORE) [[I-D.ietf-core-oscure-groupcomm](#)] as the security protocol to protect messages exchanged between the Client and the Resource Servers. Hence, it requires that both the Client and the Resource Servers have previously joined the same OSCORE group.

That is, this profile describes how access control is enforced for a Client after it has joined an OSCORE group, to access resources at other members in that group. The process for joining the OSCORE group through the respective Group Manager as defined in [[I-D.ietf-ace-key-groupcomm-oscure](#)] takes place before the process described in this document, and is out of the scope of this profile.

The Client proves its access to be authorized to the Resource Server(s) by using an Access Token bound to a key (the proof-of-possession key). This profile uses Group OSCORE to achieve server authentication and proof-of-possession for the Client's public key used in the OSCORE group in question. Note that proof-of-possession is not achieved through a dedicated protocol element, but instead after the first message exchange protected with Group OSCORE.

Furthermore, this profile provides proof of the Client's membership to the OSCORE group, by binding the Access Token to the Client's authentication credential used in the group and including the Client's public key, as well as to information from the pre-established Group OSCORE Security Context. This allows the Resource Server(s) to verify the Client's group membership upon reception of a message protected with Group OSCORE from that Client.

OSCORE [[RFC8613](#)] specifies how to use COSE [[RFC9052](#)][[RFC9053](#)] to secure CoAP messages. Group OSCORE builds on OSCORE to provide secure group communication, and ensures source authentication: by means of digital signatures embedded in the protected message (when using the group mode); or by protecting a message with pairwise keying material derived from the asymmetric keys of the two peers exchanging the message (when using the pairwise mode).

1.1. Terminology

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

Readers are expected to be familiar with the terms and concepts related to CBOR [[RFC8949](#)], COSE [[RFC9052](#)][[RFC9053](#)], CoAP [[RFC7252](#)], OSCORE [[RFC8613](#)], and Group OSCORE [[I-D.ietf-core-oscore-groupcomm](#)]. These especially include:

- *Group Manager, as the entity responsible for a set of groups where communications among members are secured with Group OSCORE.

- *Authentication credential, as the set of information associated with an entity, including that entity's public key and parameters associated with the public key. Examples of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [[RFC8392](#)], X.509 certificates [[RFC7925](#)] and C509 certificates [[I-D.ietf-cose-cbor-encoded-cert](#)].

Members of an OSCORE group have an associated authentication credential in the format used in the group. As per [Section 2.4](#) of [[I-D.ietf-core-oscore-groupcomm](#)], an authentication credential

provides the public key as well as the comprehensive set of information related to the public key algorithm, including, e.g., the used elliptic curve (when applicable).

Readers are expected to be familiar with the terms and concepts described in the ACE framework for authentication and authorization [RFC9200], as well as in the OSCORE profile of ACE [RFC9203]. The terminology for entities in the considered architecture is defined in OAuth 2.0 [RFC6749]. In particular, this includes Client (C), Resource Server (RS), and Authorization Server (AS).

Note that, unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol".

Additionally, this document makes use of the following terminology.

*Pairwise-only group: an OSCORE group that uses only the pairwise mode of Group OSCORE (see [Section 9](#) of [I-D.ietf-core-oscore-groupcomm]).

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

2. Protocol Overview

This section provides an overview of this profile, i.e., of how to use the ACE framework for authentication and authorization [RFC9200] to secure communications between a Client and one or more Resource Servers using Group OSCORE [I-D.ietf-core-oscore-groupcomm].

Note that this profile of ACE describes how access control can be enforced for a node after it has joined an OSCORE group, to access resources at other members in that group.

In particular, the process of joining the OSCORE group through the respective Group Manager as defined in [I-D.ietf-ace-key-groupcomm-oscore] must take place before the process described in this document, and is out of the scope of this profile.

An overview of the protocol flow for this profile is shown in [Figure 1](#), where it is assumed that both the Resource Servers RS1 and RS2 are associated with the same Authorization Server AS. It is also assumed that the Client C, as well as RS1 and RS2 have previously joined an OSCORE group with Group Identifier (gid) 0xabcd0000, and that they got assigned Sender ID (sid) 0x00, 0x01 and 0x02 in the

group, respectively. The names of messages coincide with those of [\[RFC9200\]](#) when applicable.

C	RS1	RS2	AS
[--- Resource Request --->]			
[<----- AS Request -----]			
Creation Hints			
----- POST /token ----->			
(aud: "RS1", sid: 0x00, gid: 0xabcd0000, ...)			
<-----			Access Token T1 -----
			+ Access Information
----- POST /authz-info ----->			
(access_token T1)			
<----- 2.01 Created -----			
----- POST /token ----->			
(aud: "RS2", sid: 0x00, gid: 0xabcd0000, ...)			
<-----			Access Token T2 -----
			+ Access Information
----- POST /authz-info ----->			
(access_token T2)			
<----- 2.01 Created -----			
-- Group OSCORE Request -+-->			
(kid: 0x00, \			
gid: 0xabcd0000) \----->			
	/proof-of-possession/		
<-- Group OSCORE Response ---			
(kid: 0x01)			
/proof-of-possession/			
/Mutual authentication between C and RS1/			

```
|<-- Group OSCORE Response -----|
|      (kid: 0x02)                  |
|                                  |
|/proof-of-possession/              |
|                                  |
|/Mutual authentication              |
|  between C and RS2/               |
|                                  |
|      ...                          |
```

Figure 1: Protocol Overview.

2.1. Pre-Conditions

Using Group OSCORE and this profile requires that both the Client and the Resource Servers have previously joined the same OSCORE group. This especially includes the derivation of the Group OSCORE Security Context and the assignment of unique Sender IDs to use in the group. Nodes can join the OSCORE group through the respective Group Manager by using the approach defined in [[I-D.ietf-ace-key-groupcomm-oscore](#)], which is also based on ACE.

After the Client and Resource Servers have joined the group, this profile provides access control for accessing resources on those Resource Servers, by securely communicating with Group OSCORE.

As a pre-requisite for this profile, the Client has to have successfully joined the OSCORE group where also the Resource Servers (RSs) are members. Depending on the limited information initially available, the Client may have to first discover the exact OSCORE group used by the RSs for the resources of interest, e.g., by using the approach defined in [[I-D.tiloca-core-oscore-discovery](#)].

2.2. Access Token Retrieval

This profile requires that the Client retrieves an Access Token from the AS for the resource(s) that it wants to access at the RS(s), by using the /token endpoint as specified in [Section 5.8](#) of [[RFC9200](#)].

In general, different RSs can be associated with different ASs, even if the RSs are members of the same OSCORE group. However, assuming proper configurations and trust relations, it is possible for multiple RSs associated with the same AS to be part of a single audience (i.e., a group-audience, see [Section 6.9](#) of [[RFC9200](#)]). In such a case, the Client can request a single Access Token intended to the group-audience, hence to all the RSs included therein. A particular group-audience might be defined as including all the RSs in the OSCORE group.

In the Access Token request to the AS, the Client MUST include the Group Identifier of the OSCORE group and its own Sender ID in that group. The AS MUST specify these pieces of information in the Access Token, included in the Access Token response to the Client.

Furthermore, in the Access Token request to the AS, the Client MUST also include: its own authentication credential used in the OSCORE group; and a proof-of-possession (PoP) evidence to prove possession of the corresponding private key. The PoP evidence is computed over a PoP input uniquely related to the secure communication association between the Client and the AS. The AS MUST include also the

authentication credential indicated by the Client in the Access Token.

The Access Token request and response MUST be confidentiality-protected and ensure authenticity. In this profile, it is RECOMMENDED to use OSCORE [RFC8613] between the Client and the AS, to reduce the number of libraries the client has to support. Other protocols fulfilling the security requirements defined in Sections 5 and 6 of [RFC9200] MAY alternatively be used, such as TLS [RFC8446] or DTLS [RFC9147].

2.3. Access Token Posting

After having retrieved the Access Token from the AS, the Client posts the Access Token to the RS, using the /authz-info endpoint and the mechanisms specified in Section 5.10 of [RFC9200]. When using this profile, the communication with the /authz-info endpoint is not protected.

If the Access Token is valid, the RS replies to this POST request with a 2.01 (Created) response. Also, the RS associates the received Access Token with the Group OSCORE Security Context identified by the Group Identifier specified in the Access Token, following Section 3.2 of [RFC8613]. In practice, the RS maintains a collection of Security Contexts with associated authorization information, for all the clients that it is currently communicating with. The authorization information is a policy that is used as input when processing requests from those clients.

Finally, the RS stores the association between i) the authorization information from the Access Token; and ii) the Group Identifier of the OSCORE group together with the Sender ID and the authentication credential of the Client in that group. This binds the Access Token to the Group OSCORE Security Context of the OSCORE group.

Finally, when the Client communicates with the RS using the Group OSCORE Security Context, the RS verifies that the Client is a legitimate member of the OSCORE group and especially the exact group member with the same Sender ID associated with the Access Token. This occurs when verifying a request protected with Group OSCORE, since the request includes the Client's Sender ID and either it embeds a signature computed also over that Sender ID (if protected with the group mode), or it is protected by means of pairwise symmetric keying material derived from the asymmetric keys of the two peers (if protected with the pairwise mode).

The above has considered an Access Token intended to a single RS. However, as discussed in Section 2.2, an Access Token can be intended to a group-audience including multiple RSs in the OSCORE

group. In such a case, the Client could efficiently post the Access Token to many or all of those RSs at once (e.g., over IP multicast), after which each RS individually performs the same steps described above.

2.4. Secure Communication

The Client can send a request protected with Group OSCORE [[I-D.ietf-core-oscore-groupcomm](#)] to the RS. This can be a unicast request addressed to the RS, or a one-to-many group request (e.g., over IP multicast) addressed to the OSCORE group where the RS is also a member. To this end, the Client uses the Group OSCORE Security Context already established upon joining the OSCORE group, e.g., by using the approach defined in [[I-D.ietf-ace-key-groupcomm-oscore](#)]. The RS may send a response back to the Client, protecting it by means of the same Group OSCORE Security Context.

3. Client-AS Communication

This section details the Access Token POST Request that the Client sends to the /token endpoint of the AS, as well as the related Access Token response.

The Access Token MUST be bound to the public key of the Client as proof-of-possession key (pop-key), which is included in the Client's authentication credential specified in the 'cnf' claim of the Access Token.

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.

The POST request is formatted as the analogous Client-to-AS request in the OSCORE profile of ACE (see [Section 3.1](#) of [[RFC9203](#)]), with the following additional parameters that MUST be included in the payload.

- *'context_id', defined in [Section 3.1.1](#) of this document. This parameter specifies the Group Identifier (GID), i.e., the ID Context of an OSCORE group that includes as members both the Client and the RS(s) in the audience for which the Access Token is asked to be issued. In particular, the Client wishes to communicate with the RS(s) in that audience using the Group OSCORE Security Context associated with that OSCORE group.

*'salt_input', defined in [Section 3.1.2](#) of this document. This parameter includes the Sender ID that the Client has in the OSCORE group whose GID is specified in the 'context_id' parameter above.

*'req_cnf', defined in [Section 3.1](#) of [\[RFC9201\]](#). This parameter follows the syntax from [Section 3.1](#) of [\[RFC8747\]](#), and its inner confirmation value specifies the authentication credential that the Client uses in the OSCORE group. The public key included in the authentication credential will be used as the pop-key bound to the Access Token.

At the time of writing this specification, acceptable formats of authentication credentials in Group OSCORE are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [\[RFC8392\]](#), X.509 certificates [\[RFC7925\]](#), and C509 certificates [\[I-D.ietf-cose-cbor-encoded-cert\]](#).

Further formats may be available in the future, and would be acceptable to use as long as they comply with the criteria compiled in [Section 2.3](#) of [\[I-D.ietf-core-oscore-groupcomm\]](#). In particular, an authentication credential has to explicitly include the public key as well as the comprehensive set of information related to the public key algorithm, including, e.g., the used elliptic curve (when applicable).

[As to CWTs and CCSs, the CWT Confirmation Methods 'kcwt' and 'kccs' are under pending registration requested by draft-ietf-ace-edhoc-oscore-profile.]

[As to X.509 certificates, the CWT Confirmation Methods 'x5bag' and '5chain' are under pending registration requested by draft-ietf-ace-edhoc-oscore-profile.]

[As to C509 certificates, the CWT Confirmation Methods 'c5b' and 'c5c' are under pending registration requested by draft-ietf-ace-edhoc-oscore-profile.]

In addition, the Client computes its proof-of-possession (PoP) evidence, in order to prove to the AS the possession of its own private key used in the OSCORE group. This allows the AS to verify that the Client indeed owns the private key associated with the public key of the authentication credential that the Client allegedly uses in the OSCORE group.

To this end, the Client MUST use as PoP input the byte representation of an information that uniquely represents the secure

communication association between the Client and the AS. It is RECOMMENDED that the Client considers the following as PoP input.

*If the Client and the AS communicate over (D)TLS, the PoP input is an exporter value computed as defined in [Section 7.5](#) of [\[RFC8446\]](#). In particular, the exporter label MUST be 'EXPORTER-ACE-Sign-Challenge-Client-AS' defined in [Section 10.5](#) of this document, together with an empty 'context_value', and 32 bytes as 'key_length'.

*If the Client and the AS communicate over OSCORE, the PoP input is the output PRK of a HKDF-Extract step [\[RFC5869\]](#), i.e., $PRK = \text{HMAC-Hash}(\text{salt}, \text{IKM})$. In particular, 'salt' takes $(x1 \parallel x2)$, where $x1$ is the ID Context of the OSCORE Security Context between the Client and the AS, $x2$ is the Sender ID of the Client in that Security Context, and \parallel denotes byte string concatenation. Also, 'IKM' is the OSCORE Master Secret of the OSCORE Security Context between the Client and the AS.

The HKDF MUST be one of the HMAC-based HKDF [\[RFC5869\]](#) algorithms defined for COSE [\[RFC9053\]](#). The Client and AS may agree on the HKDF algorithm to use during the Client's registration at the AS. HKDF SHA-256 is mandatory to implement.

Then, the Client computes the PoP evidence as follows.

*If the OSCORE group is not a pairwise-only group, the PoP evidence MUST be a signature. The Client computes the signature by using the same private key and signature algorithm it uses for signing messages in the OSCORE group. The Client's private key is the one associated with the Client's authentication credential used in the OSCORE group and specified in the 'req_cnf' parameter above.

*If the OSCORE group is a pairwise-only group, the PoP evidence MUST be a MAC computed as follows, by using the HKDF Algorithm HKDF SHA-256, which consists of composing the HKDF-Extract and HKDF-Expand steps [\[RFC5869\]](#).

$MAC = \text{HKDF}(\text{salt}, \text{IKM}, \text{info}, L)$

The input parameters of HKDF are as follows.

-salt takes as value the empty byte string.

-IKM is computed as a cofactor Diffie-Hellman shared secret (see Section 5.7.1.2 of [\[NIST-800-56A\]](#)), using an ECDH algorithm pre-agreed between Client and AS. The Client uses its own Diffie-Hellman private key and the Diffie-Hellman

public key of the AS. For X25519 and X448, the procedure is described in [Section 5](#) of [\[RFC7748\]](#).

The Client's private key is the one associated with the Client's authentication credential used in the OSCORE group and specified in the 'req_cnf' parameter above. The Client may obtain the Diffie-Hellman public key of the AS during its registration process at the AS.

The Client and AS may agree on the ECDH algorithm to use during the Client's registration at the AS. The ECDH-SS + HKDF-256 algorithm specified in [Section 6.3.1](#) of [\[RFC9053\]](#) is mandatory to implement.

-info takes as value the PoP input.

-L is equal to 8, i.e., the size of the MAC, in bytes.

Finally, the Client MUST include one of the two following parameters in the payload of the POST request to the AS.

*'client_cred_verify', defined in [Section 3.1.3](#) of this document, specifying the Client's PoP evidence as a signature, which is computed as defined above. This parameter MUST be included if and only if the OSCORE group is not a pairwise-only group.

*'client_cred_verify_mac', defined in [Section 3.1.4](#) of this document, specifying the Client's PoP evidence as a MAC, which is computed as defined above. This parameter MUST be included if and only if the OSCORE group is a pairwise-only group.

An example of such a request is shown in [Figure 2](#).


```

Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
{
  "audience" : "tempSensor4711",
  "scope" : "read",
  "context_id" : h'abcd0000',
  "salt_input" : h'00',
  "req_cnf" : {
    "kccs" : {
      "sub" : "42-50-31-FF-EF-37-32-39",
      "cnf" : {
        "COSE_Key" : {
          "kty" : 2,
          "crv" : 1,
          "x" : h'd7cc072de2205bdc1537a543d53c60a6
              acb62eccd890c7fa27c9e354089bbe13',
          "y" : h'f95e1d4b851a2cc80fff87d8e23f22af
              b725d535e515d020731e79a3b4e47120'
        }
      }
    }
  },
  "client_cred_verify" : h'...'
  (signature content omitted for brevity)
}

```

Figure 2: Example C-to-AS POST /token request for an Access Token bound to an asymmetric key.

In the example above, the Client specifies that its authentication credential in the OSCORE group is the CCS shown in [Figure 3](#).

```

{
  "sub" : "42-50-31-FF-EF-37-32-39",
  "cnf" : {
    "COSE_Key" : {
      "kty" : 2,
      "crv" : 1,
      "x" : h'd7cc072de2205bdc1537a543d53c60a6
          acb62eccd890c7fa27c9e354089bbe13',
      "y" : h'f95e1d4b851a2cc80fff87d8e23f22af
          b725d535e515d020731e79a3b4e47120'
    }
  }
}

```

Figure 3: Example of Client Authentication Credential as CWT Claims Set (CCS).

3.1.1. 'context_id' Parameter

The 'context_id' parameter is an OPTIONAL parameter of the Access Token request message defined in [Section 5.8.1](#) of [[RFC9200](#)]. This parameter provides a value that the Client wishes to use with the RS as a hint for a security context. Its exact content is profile specific.

3.1.2. 'salt_input' Parameter

The 'salt_input' parameter is an OPTIONAL parameter of the Access Token request message defined in [Section 5.8.1](#) of [[RFC9200](#)]. This parameter provides a value that the Client wishes to use as part of a salt with the RS, for deriving cryptographic keying material. Its exact content is profile specific.

3.1.3. 'client_cred_verify' Parameter

The 'client_cred_verify' parameter is an OPTIONAL parameter of the Access Token request message defined in [Section 5.8.1](#) of [[RFC9200](#)]. This parameter provides a signature computed by the Client to prove the possession of its own private key.

3.1.4. 'client_cred_verify_mac' Parameter

The 'client_cred_verify_mac' parameter is an OPTIONAL parameter of the Access Token request message defined in [Section 5.8.1](#) of [[RFC9200](#)]. This parameter provides a Message Authentication Code (MAC) computed by the Client to prove the possession of its own private key.

3.2. AS-to-C: Access Token

After having verified 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 MUST verify the proof-of-possession (PoP) evidence. In particular, the AS proceeds as follows.

*As PoP input, the AS uses the same value considered by the Client in [Section 3.1](#).

*As public key of the Client, the AS uses the one included in the authentication credential specified in the 'req_cnf' parameter of the Access Token request.

*If the Access Token request includes the 'client_cred_verify' parameter, this specifies the PoP evidence as a signature. Then, the AS verifies the signature by using the public key of the Client.

*If the Access Token request includes the 'client_cred_verify_mac' parameter, this specifies the PoP evidence as a Message Authentication Code (MAC).

Then, the AS recomputes the MAC through the same process taken by the Client when preparing the value of the 'client_cred_verify_mac' parameter for the Access Token (see [Section 3.1](#)), with the difference that the AS uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the Client. The verification succeeds if and only if the recomputed MAC is equal to the MAC conveyed as PoP evidence in the Access Token request.

If both the 'client_cred_verify' and 'client_cred_verify_mac' parameters are present, or if the verification of the PoP evidence fails, the AS considers the Client request invalid.

If the Client request was invalid, or not authorized, the AS returns an error response as described in [Section 5.8.3](#) of [[RFC9200](#)].

If all verifications are successful, the AS responds as defined in [Section 5.8.2](#) of [[RFC9200](#)]. In particular:

*The AS can signal that the use of Group OSCORE is REQUIRED for a specific Access Token by including the 'ace_profile' parameter with the value "coap_group_oscore" in the Access Token response. The Client MUST use Group OSCORE towards all the Resource Servers for which this Access Token is valid. 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.

*The AS MUST NOT include the 'rs_cnf' parameter defined in [[RFC9201](#)]. In general, the AS may not be aware of the authentication credentials (and public keys included thereof) that the RSs use in the OSCORE group. Also, the Client is able to retrieve the authentication credentials of other group members from the responsible Group Manager, both upon joining the group or later on as a group member, as defined in [[I-D.ietf-ace-key-groupcomm-oscore](#)].

The AS MUST include the following information as metadata of the issued Access Token. The use of CBOR web tokens (CWT) as specified in [\[RFC8392\]](#) is RECOMMENDED.

- *The profile "coap_group_oscore". If the Access Token is a CWT, this is placed in the 'ace_profile' claim of the Access Token, as per [Section 5.10](#) of [\[RFC9200\]](#).

- *The salt input specified in the 'salt_input' parameter of the Token Request. If the Access Token is a CWT, the content of the 'salt_input' parameter MUST be placed in the 'salt_input' claim of the Access Token, defined in [Section 3.2.1](#) of this document.

- *The Context Id input specified in the 'context_id' parameter of the Token Request. If the Access Token is a CWT, the content of the 'context_id' parameter MUST be placed in the 'contextId_input' claim of the Access Token, defined in [Section 3.2.2](#) of this document.

- *The authentication credential that the client uses in the OSCORE group and specified in the 'req_cnf' parameter of the Token request.

If the Access Token is a CWT, the Client's authentication credential MUST be specified in the 'cnf' claim, which follows the syntax from [Section 3.1](#) of [\[RFC8747\]](#). In particular, the 'cnf' claim includes the same authentication credential specified in the 'req_cnf' parameter of the Token Request (see [Section 3.1](#)).

[Figure 4](#) shows an example of such an AS response. The access token has been truncated for readability.

```
Header: Created (Code=2.01)
Content-Type: "application/ace+cbor"
Payload:
{
  "access_token" : h'8343a1010aa2044c53 ...'
    (remainder of CWT omitted for brevity),
  "ace_profile" : "coap_group_oscore",
  "expires_in" : 3600
}
```

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

[Figure 5](#) shows an example CWT Claims Set, containing the Client's public key in the group (as pop-key), as specified by the inner confirmation value in the 'cnf' claim.

```

{
  "aud" : "tempSensorInLivingRoom",
  "iat" : "1360189224",
  "exp" : "1360289224",
  "scope" : "temperature_g firmware_p",
  "cnf" : {
    "kccs" : {
      "sub" : "42-50-31-FF-EF-37-32-39",
      "cnf" : {
        "COSE_Key" : {
          "kty" : 2,
          "crv" : 1,
          "x" : h'd7cc072de2205bdc1537a543d53c60a6
              acb62eccd890c7fa27c9e354089bbe13',
          "y" : h'f95e1d4b851a2cc80fff87d8e23f22af
              b725d535e515d020731e79a3b4e47120'
        }
      }
    }
  }
  "salt_input" : h'00',
  "contextId_input" : h'abcd0000'
}

```

Figure 5: Example CWT Claims Set with OSCORE parameters.

The same CWT Claims Set as in [Figure 5](#) and encoded in CBOR is shown in [Figure 6](#), using the value abbreviations defined in [\[RFC9200\]](#) and [\[RFC8747\]](#). The bytes in hexadecimal are reported in the first column, while their corresponding CBOR meaning is reported after the "#" sign on the second column, for easiness of readability.

Editor's note: it should be checked (and in case fixed) that the values used below (which are not yet registered) are the final values registered by IANA.

```

A7                                     # map(7)
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)
    05                               # unsigned(5)
    A2                               # map(2)
        02                           # unsigned(2)
        77                           # text(23)
            34322D35302D33312D46462D45462D33372D33322D3339
            # "42-50-31-FF-EF-37-32-39"
    08                               # unsigned(8)
    A1                               # map(1)
        01                           # unsigned(1)
        A4                           # map(4)
            01                       # unsigned(1)
            02                       # unsigned(2)
            20                       # negative(0)
            01                       # unsigned(1)
            21                       # negative(1)
            58 20                     # bytes(32)
                D7CC072DE2205BDC1537A543D53C60A6
                ACB62ECCD890C7FA27C9E354089BBE13
            22                       # negative(2)
            58 20                     # bytes(32)
                F95E1D4B851A2CC80FFF87D8E23F22AF
                B725D535E515D020731E79A3B4E47120
18 3C                                 # unsigned(60)
41                                     # bytes(1)
    00
18 3D                                 # unsigned(61)
44                                     # bytes(4)
    ABCD0000

```

Figure 6: Example CWT Claims Set with OSCORE parameters, CBOR encoded.

3.2.1. Salt Input Claim

The 'salt_input' claim provides a value that the Client requesting the Access Token wishes to use as a part of a salt with the RS, e.g., for deriving cryptographic material.

This parameter specifies the value of the salt input, encoded as a CBOR byte string.

3.2.2. Context ID Input Claim

The 'contextId_input' claim provides a value that the Client requesting the Access Token wishes to use with the RS, as a hint for a security context.

This parameter specifies the value of the Context ID input, encoded as a CBOR byte string.

4. Client-RS Communication

This section details the POST request and response to the /authz-info endpoint between the Client and the RS.

The proof-of-possession required to bind the Access Token to the Client is explicitly performed when the RS receives and verifies a request from the Client protected with Group OSCORE, either with the group mode (see [Section 8](#) of [[I-D.ietf-core-oscore-groupcomm](#)]) or with the pairwise mode (see [Section 9](#) of [[I-D.ietf-core-oscore-groupcomm](#)]).

In particular, the RS uses the Client's public key bound to the Access Token, either when verifying the signature of the request (if protected with the group mode), or when verifying the request as integrity-protected with pairwise keying material derived from the two peers' authentication credentials and asymmetric keys (if protected with the pairwise mode). In either case, the RS also authenticates the Client.

Similarly, when receiving a protected response from the RS, the Client uses the RS's public key either when verifying the signature of the response (if protected with the group mode), or when verifying the response as integrity-protected with pairwise keying material derived from the two peers' authentication credentials and asymmetric keys (if protected with the pairwise mode). In either case, the Client also authenticates the RS. Mutual authentication is only achieved after the client has successfully verified the protected response from the RS.

Therefore, an attacker using a stolen Access Token cannot generate a valid Group OSCORE message as protected through the Client's private

key, and thus cannot prove possession of the pop-key bound to the Access Token. Also, if a Client legitimately owns an Access Token but has not joined the OSCORE group, it cannot generate a valid Group OSCORE message, as it does not store the necessary keying material shared among the group members.

Furthermore, a Client C1 is supposed to obtain a valid Access Token from the AS, as specifying the Client's authentication credential (and the public key included thereof) associated with the Client's private key used in the OSCORE group, together with its own Sender ID in that OSCORE group (see [Section 3.1](#)). This allows the RS receiving the Access Token to verify with the Group Manager of that OSCORE group whether such a Client indeed has that Sender ID and uses that authentication credential in the OSCORE group.

As a consequence, a different Client C2, also member of the same OSCORE group, is not able to impersonate C1, by: i) getting a valid Access Token, specifying the Sender ID of C1 and a different (made-up) authentication credential; ii) successfully posting the Access Token to the RS; and then iii) attempting to communicate using Group OSCORE impersonating C1, while blaming C1 for the consequences.

4.1. C-to-RS POST to authz-info Endpoint

The Client posts the Access Token to the /authz-info endpoint of the RS, as defined in [Section 5.10.1](#) of [[RFC9200](#)].

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

The RS MUST verify the validity of the Access Token as defined in [Section 5.10.1](#) of [[RFC9200](#)], with the following additions.

- *The RS MUST check that the claims 'salt_input', 'contextId_input', and 'cnf' are included in the Access Token.

- *The RS considers: the content of the 'contextId_input' claim as the GID of the OSCORE group; the content of the 'salt_input' claim as the Sender ID that the Client has in the group; and the inner confirmation value of 'cnf' claim as the authentication credential that the Client uses in the group.

The RS MUST check whether it already stores the authentication credential specified in the inner confirmation value of the 'cnf' claim as associated with the pair (GID, Sender ID) above.

If this is not the case, the RS MUST request the Client's authentication credential to the Group Manager of the OSCORE group as described in [Section 9.3](#) of [[I-D.ietf-ace-key-groupcomm-oscore](#)], specifying the Client's

Sender ID in the OSCORE group, i.e., the value of the 'salt_input' claim. Then, the RS performs the following actions.

- The RS MUST check whether the Client's authentication credential retrieved from the Group Manager matches the one retrieved from the inner confirmation value of the 'cnf' claim of the Access Token.
- The RS MUST check whether the Client's Sender ID provided by the Group Manager together with the Client's authentication credential matches the one retrieved from the 'salt_input' claim of the Access Token.

If any of the checks above fails, the RS MUST consider the Access Token non valid, and MUST respond to the Client with an error response code equivalent to the CoAP code 4.00 (Bad Request).

If the Access Token is valid and further checks on its content are successful, the RS associates the authorization information from the Access Token with the Group OSCORE Security Context.

In particular, the RS associates the authorization information from the Access Token with the triple (GID, SaltInput, AuthCred), where GID is the Group Identifier of the OSCORE group, while SaltInput and AuthCred are the Sender ID and the authentication credential that the Client uses in that OSCORE group, respectively.

The RS MUST keep this association up-to-date over time, as the triple (GID, SaltInput, AuthCred) associated with the Access Token might change. In particular:

*If the OSCORE group is rekeyed (see [Section 3.2](#) of [\[I-D.ietf-core-oscore-groupcomm\]](#) and [Section 11](#) of [\[I-D.ietf-ace-key-groupcomm-oscore\]](#)), the Group Identifier also changes in the group, and the new one replaces the current 'GID' value in the triple (GID, SaltInput, AuthCred).

*If the Client requests and obtains a new OSCORE Sender ID from the Group Manager (see [Section 2.6.3.1](#) of [\[I-D.ietf-core-oscore-groupcomm\]](#) and [Section 9.2](#) of [\[I-D.ietf-ace-key-groupcomm-oscore\]](#)), the new Sender ID replaces the current 'SaltInput' value in the triple (GID, SaltInput, AuthCred).

Finally, the RS MUST send a 2.01 (Created) response to the Client, as defined in [Section 5.10.1](#) of [\[RFC9200\]](#).

4.3. Client-RS Secure Communication

When previously joining the OSCORE group, both the Client and RS have already established the related Group OSCORE Security Context to communicate as group members. Therefore, they can simply start to securely communicate using Group OSCORE, without deriving any additional keying material or security association.

4.3.1. Client Side

After having received the 2.01 (Created) response from the RS, following the POST request to the /authz-info endpoint, the Client starts the communication with the RS, by sending a request protected with Group OSCORE using the Group OSCORE Security Context [[I-D.ietf-core-oscore-groupcomm](#)].

When communicating with the RS to access the resources as specified by the authorization information, the Client MUST use the Group OSCORE Security Context of the OSCORE group, whose GID was specified in the 'context_id' parameter of the Access Token request.

4.3.2. Resource Server Side

After successful validation of the Access Token as defined in [Section 4.2](#) and after having sent the 2.01 (Created) response, the RS can start to communicate with the Client using Group OSCORE [[I-D.ietf-core-oscore-groupcomm](#)].

When processing an incoming request protected with Group OSCORE, the RS MUST consider as valid Client's authentication credential only the one associated with the stored Access Token. As defined in [Section 4.5](#), a possible change of authentication credential requires the Client to upload to the RS a new Access Token bound to the new authentication credential.

Additionally, for every incoming request, if Group OSCORE verification succeeds, the verification of access rights is performed as described in [Section 4.4](#).

After the expiration of the Access Token related to a Group OSCORE Security Context, if the Client uses the Group OSCORE Security Context to send a request for any resource intended for OSCORE group members and that requires an active Access Token, the RS MUST respond with a 4.01 (Unauthorized) error message protected with the Group OSCORE Security Context.

4.4. Access Rights Verification

The RS MUST follow the procedures defined in [Section 5.10.2](#) of [[RFC9200](#)]. If an RS receives a request protected with Group OSCORE

from a Client, the RS processes the request according to [\[I-D.ietf-core-oscore-groupcomm\]](#).

If the Group OSCORE verification succeeds, and the target resource requires authorization, the RS retrieves the authorization information from the Access Token associated with the Group OSCORE Security Context. Then, the RS MUST verify that the action requested on the resource is authorized.

The response code MUST be 4.01 (Unauthorized) if the RS has no valid Access Token for the Client. If the RS has an Access Token for the Client but no actions are authorized on the target resource, the RS MUST reject the request and MUST respond to the Client with a 4.03 (Forbidden) response. If the RS has an Access Token for the Client but the requested action is not authorized, the RS MUST reject the request and MUST respond to the Client with a 4.05 (Method Not Allowed) response.

4.5. Change of Client's Authentication Credential in the Group

During its membership in the OSCORE group, the Client might change the authentication credential that it uses in the group. When this happens, the Client uploads the new authentication credential to the Group Manager, as defined in [Section 9.4](#) of [\[I-D.ietf-ace-key-groupcomm-oscore\]](#).

After that, and in order to continue communicating with the RS, the Client MUST perform the following actions.

1. The Client requests a new Access Token to the AS, as defined in [Section 3](#). In particular, when sending the POST request as defined in [Section 3.1](#), the Client indicates:

- *The current Group Identifier of the OSCORE group, as value of the 'context_id' parameter.

- *The current Sender ID it has in the OSCORE group, as value of the 'salt_input' parameter.

- *The new authentication credential it uses in the OSCORE group, as inner confirmation value of the 'req_cnf' parameter.

- *The proof-of-possession (PoP) evidence corresponding to the public key of the new authentication credential, as value of the 'client_cred_verify' or 'client_cred_verify_mac' parameter.

2. After receiving the response from the AS (see [Section 3.2](#)), the Client performs the same exchanges with the RS as defined in [Section 4](#).

When receiving the new Access Token, the RS performs the same steps defined in [Section 4.2](#), with the following addition in case the new Access Token is successfully verified and stored. The RS also deletes the old Access Token, i.e., the one whose associated triple (GID, SaltInput, AuthCred) has the same GID and SaltInput values as in the triple including the new authentication credential of the Client and associated with the new Access Token.

5. Secure Communication with the AS

As specified in the ACE framework (see Sections [5.8](#) and [5.9](#) of [\[RFC9200\]](#)), the requesting entity (RS and/or Client) and the AS communicate via the /token or /introspection endpoint. The use of CoAP and OSCORE [\[RFC8613\]](#) for this communication is RECOMMENDED in this profile. Other protocols fulfilling the security requirements defined in Sections [5](#) and [6](#) of [\[RFC9200\]](#) (such as HTTP and DTLS or TLS) MAY be used instead.

If OSCORE [\[RFC8613\]](#) is used, the requesting entity and the AS are expected to have a pre-established Security Context in place. How this Security Context is established is out of the scope of this profile. Furthermore, the requesting entity and the AS communicate using OSCORE through the /introspection endpoint as specified in [Section 5.9](#) of [\[RFC9200\]](#), and through the /token endpoint as specified in [Section 5.8](#) of [\[RFC9200\]](#).

6. Discarding the Security Context

As members of an OSCORE group, the Client and the RS may independently leave the group or be forced to, e.g., if compromised or suspected so. Upon leaving the OSCORE group, the Client or RS also discards the Group OSCORE Security Context, which may anyway be renewed by the Group Manager through a group rekeying process (see [Section 3.2](#) of [\[I-D.ietf-core-oscore-groupcomm\]](#)).

The Client or RS can acquire a new Group OSCORE Security Context, by re-joining the OSCORE group, e.g., by using the approach defined in [\[I-D.ietf-ace-key-groupcomm-oscore\]](#). In such a case, the Client SHOULD request a new Access Token and post it to the RS.

7. CBOR Mappings

The new parameters defined in this document MUST be mapped to CBOR types as specified in [Figure 7](#), using the given integer abbreviation for the map key.

Parameter name	CBOR Key	Value Type
context_id	TBD	bstr
salt_input	TBD	bstr
client_cred_verify	TBD	bstr
client_cred_verify_mac	TBD	bstr

Figure 7: CBOR mappings for new parameters.

The new claims defined in this document MUST be mapped to CBOR types as specified in [Figure 8](#), using the given integer abbreviation for the map key.

Claim name	CBOR Key	Value Type
salt_input	TBD	bstr
contextId_input	TBD	bstr

Figure 8: CBOR mappings for new claims.

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.

The proof-of-possession (PoP) key bound to an Access Token is always an asymmetric key, i.e., the public key included in the authentication credential that the Client uses in the OSCORE group. This means that there is never a same shared secret used as PoP key with possible multiple RSs. Therefore, it is possible and safe for the AS to issue an Access Token for an audience that includes multiple RSs (i.e., a group-audience, see [Section 6.9](#) of [\[RFC9200\]](#)).

In such a case, as per [Section 6.1](#) of [\[RFC9200\]](#), the AS has to ensure the integrity protection of the Access Token by protecting it through an asymmetric signature. In addition, the used group-audience has to correctly identify all the RSs that are intended recipients of the Access Token, and for which the single scope specified in the Access Token applies. As a particular case, the audience can be the name of the OSCORE group, if the Access Token is intended to all the RSs in that group.

Furthermore, this document inherits the general security considerations about Group OSCORE [[I-D.ietf-core-oscore-groupcomm](#)], as to the specific use of Group OSCORE according to this profile.

Group OSCORE is designed to secure point-to-point as well as point-to-multipoint communications, providing a secure binding between a single request and multiple corresponding responses. In particular, Group OSCORE fulfills the same security requirements of OSCORE.

Group OSCORE ensures source authentication of messages both in group mode (see [Section 8](#) of [[I-D.ietf-core-oscore-groupcomm](#)]) and in pairwise mode (see [Section 9](#) of [[I-D.ietf-core-oscore-groupcomm](#)]).

When protecting an outgoing message in group mode, the sender uses its private key to compute a digital signature, which is embedded in the protected message. The group mode can be used to protect messages sent to multiple recipients (e.g., over IP multicast) or to a single recipient.

When protecting an outgoing message in pairwise mode, the sender uses a pairwise symmetric key, as derived from the asymmetric keys of the two peers exchanging the message. The pairwise mode can be used to protect only messages intended to a single recipient.

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.

As this profile uses Group OSCORE, the privacy considerations from [[I-D.ietf-core-oscore-groupcomm](#)] apply to this document as well.

An unprotected response to an unauthorized request may disclose information about the RS and/or its existing relationship with the Client. It is advisable to include as little information as possible in an unencrypted response. However, since both the Client and the RS share a Group OSCORE Security Context, unauthorized, yet protected requests are followed by protected responses, which can thus include more detailed information.

Although it may be encrypted, the Access Token is sent in the clear to the /authz-info endpoint at the RS. Thus, if the Client uses the same single Access Token from multiple locations with multiple Resource Servers, it can risk being tracked through the Access Token's value.

Note that, even though communications are protected with Group OSCORE, some information might still leak, due to the observable size, source address, and destination address of exchanged messages.

10. IANA Considerations

This document has the following actions for IANA.

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

10.1. ACE Profile Registry

IANA is asked to add the following entry to the "ACE Profile" registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group, following the procedure specified in [Section 8.8](#) of [\[RFC9200\]](#).

*Name: coap_group_oscore

*Description: Profile to secure communications between constrained nodes using the Authentication and Authorization for Constrained Environments framework, by enabling authentication and fine-grained authorization of members of an OSCORE group, that use a pre-established Group OSCORE Security Context to communicate with Group OSCORE.

*CBOR Value: TBD (value between 1 and 255)

*Reference: [RFC-XXXX]

10.2. OAuth Parameters Registry

IANA is asked to add the following entries to the "OAuth Parameters" registry, following the procedure specified in [Section 11.2](#) of [\[RFC6749\]](#).

*Name: "context_id"

*Parameter Usage Location: token request

*Change Controller: IESG

*Specification Document(s): [Section 3.1.1](#) of [RFC-XXXX]

*Name: "salt_input"

*Parameter Usage Location: token request

*Change Controller: IESG

*Specification Document(s): [Section 3.1.2](#) of [RFC-XXXX]

*Name: "client_cred_verify"

*Parameter Usage Location: token request

*Change Controller: IESG

*Specification Document(s): [Section 3.1.3](#) of [RFC-XXXX]

*Name: "client_cred_verify_mac"

*Parameter Usage Location: token request

*Change Controller: IESG

*Specification Document(s): [Section 3.1.4](#) of [RFC-XXXX]

10.3. OAuth Parameters CBOR Mappings Registry

IANA is asked to add the following entries to the "OAuth Parameters CBOR Mappings" registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group, following the procedure specified in [Section 8.10](#) of [RFC9200].

*Name: "context_id"

*CBOR Key: TBD

*Value Type: bstr

*Reference: [Section 3.1.1](#) of [RFC-XXXX]

*Name: "salt_input"

*CBOR Key: TBD

*Value Type: bstr

*Reference: [Section 3.1.2](#) of [RFC-XXXX]

*Name: "client_cred_verify"

*CBOR Key: TBD

*Value Type: bstr

*Reference: [Section 3.1.3](#) of [RFC-XXXX]

*Name: "client_cred_verify_mac"

*CBOR Key: TBD

*Value Type: bstr

*Reference: [Section 3.1.4](#) of [RFC-XXXX]

10.4. CBOR Web Token (CWT) Claims Registry

IANA is asked to add the following entries to the "CBOR Web Token (CWT) Claims" registry, following the procedure specified in [Section 9.1](#) of [[RFC8392](#)].

*Claim Name: "salt_input"

*Claim Description: Client provided salt input

*JWT Claim Name: "N/A"

*Claim Key: TBD

*Claim Value Type(s): bstr

*Change Controller: IESG

*Specification Document(s): [Section 3.2.1](#) of [RFC-XXXX]

*Claim Name: "contextId_input"

*Claim Description: Client context id input

*JWT Claim Name: "N/A"

*Claim Key: TBD

*Claim Value Type(s): bstr

*Change Controller: IESG

*Specification Document(s): [Section 3.2.2](#) of [RFC-XXXX]

10.5. TLS Exporter Label Registry

IANA is asked to add the following entry to the "TLS Exporter Label" registry within the "Transport Layer Security (TLS) Parameters" registry group, following the procedure specified in [Section 6](#) of [[RFC5705](#)] and updated in [Section 12](#) of [[RFC8447](#)].

*Value: EXPORTER-ACE-Sign-Challenge-Client-AS

*DTLS-OK: Y

*Recommended: N

*Reference: [Section 3.1](#) of [RFC-XXXX]

11. References

11.1. Normative References

- [I-D.ietf-ace-key-groupcomm-oscore] Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-oscore-16, 6 March 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-oscore-16>>.
- [I-D.ietf-core-groupcomm-bis] Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-08, 11 January 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-08>>.
- [I-D.ietf-core-oscore-groupcomm] Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and J. Park, "Group Object Security for Constrained RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-18, 22 June 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-18>>.
- [NIST-800-56A] Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography - NIST Special Publication 800-56A, Revision 3", April 2018, <<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf>>.
- [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>>.
- [RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <<https://www.rfc-editor.org/rfc/rfc5705>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/rfc/rfc5869>>.

[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>>.

[RFC7748]

Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <<https://www.rfc-editor.org/rfc/rfc7748>>.

[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>>.

[RFC8447]

Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <<https://www.rfc-editor.org/rfc/rfc8447>>.

[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>>.

[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>>.

[RFC9052]

Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <<https://www.rfc-editor.org/rfc/rfc9052>>.

[RFC9053]

Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <<https://www.rfc-editor.org/rfc/rfc9053>>.

[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>>.

11.2. Informative References

[I-D.ietf-ace-mqtt-tls-profile] Sengul, C. and A. Kirby, "Message Queuing Telemetry Transport (MQTT)-TLS profile of Authentication and Authorization for Constrained Environments (ACE) Framework", Work in Progress, Internet-Draft, draft-ietf-ace-mqtt-tls-profile-17, 23 March 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-ace-mqtt-tls-profile-17>>.

[I-D.ietf-cose-cbor-encoded-cert]

Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuheid, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-06, 7 July 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-06>>.

[I-D.tiloca-core-oscore-discovery] Tiloca, M., Amsüss, C., and P. Van der Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-discovery-13, 8 March 2023, <<https://datatracker.ietf.org/doc/html/draft-tiloca-core-oscore-discovery-13>>.

[NIST-800-207] Rose, S., Borchert, O., Mitchell, S., and S. Connelly, "Zero Trust Architecture - NIST Special Publication 800-207", August 2020, <<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf>>.

[RFC7925]

Tschafenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <<https://www.rfc-editor.org/rfc/rfc7925>>.

[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>>.

[RFC9147]

Rescorla, E., Tschafenig, 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>>.

[RFC9202]

Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)", RFC 9202, DOI 10.17487/RFC9202, August 2022, <<https://www.rfc-editor.org/rfc/rfc9202>>.

Appendix A. Profile Requirements

This appendix lists the specifications of this profile based on the requirements of the ACE 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: Not specified.

*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: Group OSCORE, by using a pre-established Group OSCORE Security Context.

*Specify how the Client and the RS mutually authenticate: Explicitly, by possession of a common Group OSCORE Security Context, and by either: usage of digital signatures embedded in messages, if protected with the group mode of Group OSCORE; or protection of messages with the pairwise mode of Group OSCORE, by

using pairwise symmetric keys derived from the asymmetric keys of the two peers exchanging the message. Note that mutual authentication is not completed before the Client has verified a Group OSCORE response using the corresponding Group OSCORE Security Context.

*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: Group OSCORE algorithms; asymmetric keys verified and distributed by a Group Manager.

*Specify a unique ace_profile identifier: coap_group_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 other methods of token transport than the authz-info endpoint: Not defined.

Acknowledgments

The authors sincerely thank Christian Amsüss, Benjamin Kaduk, John Preuß Mattsson, Dave Robin, Jim Schaad, and Göran Selander for their comments and feedback.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; and by the H2020 project SIFIS-Home (Grant agreement 952652).

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Stockholm Kista
Sweden

Email: marco.tiloca@ri.se

Rikard Höglund
RISE AB
Isafjordsgatan 22

SE-16440 Stockholm Kista
Sweden

Email: rikard.hoglund@ri.se

Ludwig Seitz
Combitech
Djäknegatan 31
SE-21135 Malmö Malmö
Sweden

Email: ludwig.seitz@combitech.com

Francesca Palombini
Ericsson AB
Torshamnsgatan 23
SE-16440 Stockholm Kista
Sweden

Email: francesca.palombini@ericsson.com