Key Provisioning for Group Communication using ACE
draft-ietf-ace-key-groupcomm-02

Abstract

This document defines message formats and procedures for requesting and distributing group keying material using the ACE framework, to protect communications between group members.

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 January 6, 2020.

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.
1. Introduction

This document expands the ACE framework [I-D.ietf-ace-oauth-authz] to define the format of messages used to request, distribute and renew the keying material in a group communication scenario, e.g. based on multicast [RFC7390][I-D.dijk-core-groupcomm-bis] or on publishing-subscribing [I-D.ietf-core-coap-pubsub]. The ACE framework is based on CBOR [RFC7049], so CBOR is the format used in this specification. However, using JSON [RFC8259] instead of CBOR is possible, using the conversion method specified in Sections 4.1 and 4.2 of [RFC7049].

Profiles that use group communication can build on this document to specify the selection of the message parameters defined in this document to use and their values. Known applications that can benefit from this document would be, for example, those addressing group communication based on multicast [RFC7390][I-D.dijk-core-groupcomm-bis] or publishing/subscribing [I-D.ietf-core-coap-pubsub] in ACE.

If the application requires backward and forward security, updated keying material is generated and distributed to the group members (rekeying), when membership changes. A key management scheme performs the actual distribution of the updated keying material to the group. In particular, the key management scheme rekeys the current group members when a new node joins the group, and the remaining group members when a node leaves the group. This document provides a message format for group rekeying that allows to fulfill these requirements. Rekeying mechanisms can be based on [RFC2093], [RFC2094] and [RFC2627].

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.

Readers are expected to be familiar with the terms and concepts described in [I-D.ietf-ace-oauth-authz] and [RFC8152], such as Authorization Server (AS) and Resource Server (RS).

This document additionally uses the following terminology:

- Transport profile, to indicate a profile of ACE as per Section 5.6.4.3 of [I-D.ietf-ace-oauth-authz]. That is, a transport profile specifies the communication protocol and communication security protocol between an ACE Client and Resource...
Server, as well as proof-of-possession methods, if it supports proof-of-possession access tokens. Transport profiles of ACE include, for instance, [I-D.ietf-ace-oscose-profile], [I-D.ietf-ace-dtls-authorize] and [I-D.ietf-ace-mqtt-tls-profile].

- Application profile, to indicate a profile of ACE that defines how applications enforce and use supporting security services they require. These services include, for instance, provisioning, revocation and (re-)distribution of keying material. An application profile may define specific procedures and message formats.

2. Overview

![Key Distribution Participants](image.png)

Figure 1: Key Distribution Participants

The following participants (see Figure 1) take part in the authorization and key distribution.

- **Client (C):** node that wants to join the group communication. It can request write and/or read rights.

- **Authorization Server (AS):** same as AS in the ACE Framework; it enforces access policies, and knows if a node is allowed to join the group with write and/or read rights.

- **Key Distribution Center (KDC):** maintains the keying material to protect group communications, and provides it to Clients authorized to join the group. During the first part of the exchange (Section 3), it takes the role of the RS in the ACE Framework. During the second part (Section 4), which is not based on the ACE Framework, it distributes the keying material. In addition, it provides the latest keying material to group members when requested. If required by the application, the KDC renews...
and re-distributes the keying material in the group when membership changes.

- Dispatcher: entity through which the Clients communicate with the group and which distributes messages to the group members. Examples of dispatchers are: the Broker node in a pub-sub setting; a relayer node for group communication that delivers group messages as multiple unicast messages to all group members; an implicit entity as in a multicast communication setting, where messages are transmitted to a multicast IP address and delivered on the transport channel.

This document specifies the message flows and formats for:

- Authorizing a new node to join the group (Section 3), and providing it with the group keying material to communicate with the other group members (Section 4).

- Removing of a current member from the group (Section 5).

- Retrieving keying material as a current group member (Section 6 and Section 7).

- Renewing and re-distributing the group keying material (rekeying) upon a membership change in the group (Section 4.2 and Section 5).

Figure 2 provides a high level overview of the message flow for a node joining a group communication setting.

<table>
<thead>
<tr>
<th>C</th>
<th>AS</th>
<th>KDC</th>
<th>Dispatcher</th>
<th>Group Member</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Authorization Request -&gt;</td>
<td></td>
<td>\</td>
<td>Defined in the ACE framework</td>
</tr>
<tr>
<td></td>
<td>Authorization Response</td>
<td>\</td>
<td></td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Token Post --------------&gt;</td>
<td>\</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Key Distribution Request ----&gt;</td>
<td>\</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Key Distribution Response ----</td>
<td>---</td>
<td>---</td>
<td>Group Rekeying ----&gt;</td>
</tr>
<tr>
<td></td>
<td>Protected communication ===</td>
<td>=</td>
<td>=</td>
<td>=</td>
</tr>
</tbody>
</table>

Figure 2: Message Flow Upon New Node’s Joining
The exchange of Authorization Request and Authorization Response between Client and AS MUST be secured, as specified by the transport profile of ACE used between Client and KDC.

The exchange of Key Distribution Request and Key Distribution Response between Client and KDC MUST be secured, as a result of the transport profile of ACE used between Client and KDC.

All further communications between the Client and the KDC MUST be secured, for instance with the same security mechanism used for the Key Distribution exchange.

All communications between a Client and the other group members MUST be secured using the keying material provided in Section 4.

3. Authorization to Join a Group

This section describes in detail the format of messages exchanged by the participants when a node requests access to a group. The first part of the exchange is based on ACE [I-D.ietf-ace-oauth-authz].

As defined in [I-D.ietf-ace-oauth-authz], the Client requests from the AS an authorization to join the group through the KDC (see Section 3.1). If the request is approved and authorization is granted, the AS provides the Client with a proof-of-possession access token and parameters to securely communicate with the KDC (see Section 3.2). Communications between the Client and the AS MUST be secured, according to the transport profile of ACE used. The Content-Format used in the messages is the one specified by the used transport profile of ACE (e.g. application/ace+cbor for the first two messages and application/cwt for the third message, depending on the format of the access token).

Figure 3 gives an overview of the exchange described above.

```
Client          AS            KDC
                │            └───>
                │              │
                │      ---- Authorization Request: POST /token ---->
                │
                │              │
                │        <--- Authorization Response: 2.01 (Created) ---
                │
                │              │
                │      ----- POST Token: POST /authz-info ---------->
                │
```

Figure 3: Message Flow of Join Authorization
3.1. Authorization Request

The Authorization Request sent from the Client to the AS is as defined in Section 5.6.1 of [I-D.ietf-ace-oauth-authz] and MUST contain the following parameters:

- 'grant_type', with value "client_credentials".

Additionally, the Authorization Request MAY contain the following parameters, which, if included, MUST have the corresponding values:

- 'scope', containing the identifier of the specific group (or topic in the case of pub-sub) that the Client wishes to access, and optionally the role(s) that the Client wishes to take. This value is a CBOR array encoded as a byte string, which contains:
  * As first element, the identifier of the specific group or topic.
  * Optionally, as second element, the role (or CBOR array of roles) the Client wishes to take in the group.

The encoding of the group or topic identifier and of the role identifiers is application specific.

- 'audience', with an identifier of a KDC.

- 'req_cnf', as defined in Section 3.1 of [I-D.ietf-ace-oauth-params], optionally containing the public key or a reference to the public key of the Client, if it wishes to communicate that to the AS.

- Other additional parameters as defined in [I-D.ietf-ace-oauth-authz], if necessary.

3.2. Authorization Response

The Authorization Response sent from the AS to the Client is as defined in Section 5.6.2 of [I-D.ietf-ace-oauth-authz] and MUST contain the following parameters:

- 'access_token', containing the proof-of-possession access token.

- 'cnf' if symmetric keys are used, not present if asymmetric keys are used. This parameter is defined in Section 3.2 of [I-D.ietf-ace-oauth-params] and contains the symmetric proof-of-possession key that the Client is supposed to use with the KDC.
o ‘rs_cnf’ if asymmetric keys are used, not present if symmetric keys are used. This parameter is as defined in Section 3.2 of [I-D.ietf-ace-oauth-params] and contains information about the public key of the KDC.

o ‘exp’, contains the lifetime in seconds of the access token. This parameter MAY be omitted if the application defines how the expiration time is communicated to the Client via other means, or if it establishes a default value.

Additionally, the Authorization Response MAY contain the following parameters, which, if included, MUST have the corresponding values:

o ‘scope’, which mirrors the ‘scope’ parameter in the Authorization Request (see Section 3.1). Its value is a CBOR array encoded as a byte string, containing:

* As first element, the identifier of the specific group or topic the Client is authorized to access.

* Optionally, as second element, the role (or CBOR array of roles) the Client is authorized to take in the group.

The encoding of the group or topic identifier and of the role identifiers is application specific.

o Other additional parameters as defined in [I-D.ietf-ace-oauth-authz], if necessary.

The access token MUST contain all the parameters defined above (including the same ‘scope’ as in this message, if present, or the ‘scope’ of the Authorization Request otherwise), and additionally other optional parameters that the transport profile of ACE requires.

When receiving an Authorization Request from a Client that was previously authorized, and which still owns a valid non expired access token, the AS replies with an Authorization Response with a new access token.

3.3. Token Post

The Client sends a CoAP POST request including the access token to the KDC, as specified in Section 5.8.1 of [I-D.ietf-ace-oauth-authz]. If the specific transport profile of ACE defines it, the Client MAY use a different endpoint than /authz-info at the KDC to post the access token to.
Optionally, the Client might need to request necessary information concerning the public keys in the group, as well as concerning the algorithm and related parameters for computing signatures in the group. In such a case, the joining node MAY ask for that information to the KDC in this same request. To this end, it sends the CoAP POST request to the /authz-info endpoint using the Content-Format "application/ace+cbor" defined in Section 8.14 of [I-D.ietf-ace-oauth-authz], and includes also the following parameters:

- 'sign_info' defined in Section 3.3.1, encoding the CBOR simple value Null, to require information and parameters on the signature algorithm and on the public keys used in the group.

- 'pub_key_enc' defined in Section 3.3.2, encoding the CBOR simple value Null, to require information on the exact encoding of public keys used in the group.

The CDDL notation of the 'sign_info' and 'pub_key_enc' parameters formatted as in the request is given below.

\[
\begin{align*}
sign_info\_req & = \text{nil} \\
\text{pub_key_enc}\_req & = \text{nil}
\end{align*}
\]

Alternatively, the joining node may retrieve this information by other means.

After successful verification, the Client is authorized to receive the group keying material from the KDC and join the group. In particular, the KDC replies to the Client with a 2.01 (Created) response, using Content-Format "application/ace+cbor" defined in Section 8.14 of [I-D.ietf-ace-oauth-authz].

The payload of the 2.01 response is a CBOR map, which MUST include a nonce N generated by the KDC. The Client may use this nonce for proving the possession of its own private key (see the 'client_cred_verify' parameter in Section 4).

Optionally, if they were included in the request, the AS MAY include the 'sign_info' parameter as well as the 'pub_key_enc' parameter defined in Section 3.3.1 and Section 3.3.2 of this specification, respectively.

The 'sign_info' parameter MUST be present if the POST request included the 'sign_info' parameter with value Null. If present, the 'sign_info' parameter of the 2.01 (Created) response is a CBOR array formatted as follows.
The first element ‘sign_alg’ is an integer or a text string, indicating the signature algorithm used in the group. It is required of the application profiles to define specific values for this parameter.

The second element ‘sign_parameters’ indicates the parameters of the signature algorithm. Its structure depends on the value of ‘sign_alg’. It is required of the application profiles to define specific values for this parameter. If no parameters of the signature algorithm are specified, ‘sign_parameters’ MUST be encoding the CBOR simple value Null.

The third element ‘sign_key_parameters’ indicates the parameters of the key used with the signature algorithm. Its structure depends on the value of ‘sign_alg’. It is required of the application profiles to define specific values for this parameter. If no parameters of the key used with the signature algorithm are specified, ‘sign_key_parameters’ MUST be encoding the CBOR simple value Null.

The ‘pub_key_enc’ parameter MUST be present if the POST request included the ‘pub_key_enc’ parameter with value Null. If present, the ‘pub_key_enc’ parameter of the 2.01 (Created) response is a CBOR integer, indicating the encoding of public keys used in the group. The values of this field are registered in the "ACE Public Key Encoding" Registry, defined in Section 11.2. It is required of the application profiles to define specific values to use for this parameter.

The CDDL notation of the ‘sign_info’ and ‘pub_key_enc’ parameters formatted as in the response is given below.

```cddl
sign_info_res = [  
    sign_alg : int / tstr,  
    sign_parameters : any / nil,  
    sign_key_parameters : any / nil  
]

pub_key_enc_res = int
```

Note that the CBOR map specified as payload of the 2.01 (Created) response may include further parameters, e.g. according to the signalled transport profile of ACE.

Note that this step could be merged with the following message from the Client to the KDC, namely Key Distribution Request.
3.3.1. ‘sign_info’ Parameter

The ‘sign_info’ parameter is an OPTIONAL parameter of the AS Request Creation Hints message defined in Section 5.1.2. of [I-D.ietf-ace-oauth-authz]. This parameter contains information and parameters about the signature algorithm and the public keys to be used between the Client and the RS. Its exact content is application specific.

3.3.2. ‘pub_key_enc’ Parameter

The ‘pub_key_enc’ parameter is an OPTIONAL parameter of the AS Request Creation Hints message defined in Section 5.1.2. of [I-D.ietf-ace-oauth-authz]. This parameter contains information about the exact encoding of public keys to be used between the Client and the RS. Its exact content is application specific.

4. Key Distribution

This section defines how the keying material used for group communication is distributed from the KDC to the Client, when joining the group as a new member.

If not previously established, the Client and the KDC MUST first establish a pairwise secure communication channel using ACE. The exchange of Key Distribution Request-Response MUST occur over that secure channel. The Client and the KDC MAY use that same secure channel to protect further pairwise communications, that MUST be secured.

During this exchange, the Client sends a request to the AS, specifying the group it wishes to join (see Section 4.1). Then, the KDC verifies the access token and that the Client is authorized to join that group; if so, it provides the Client with the keying material to securely communicate with the member of the group (see Section 4.2). The Content-Format used in the messages is set to application/cbor.

Figure 4 gives an overview of the exchange described above.

```
Client                                               KDC
|                                                  |
|---- Key Distribution Request: POST /group-id ---> |
|                                                  |
|<--- Key Distribution Response: 2.01 (Created) ---|
```

Figure 4: Message Flow of Key Distribution to a New Group Member
The same set of message can also be used for the following cases, when the Client is already a group member:

- The Client wishes to (re-)get the current keying material, for cases such as expiration, loss or suspected mismatch, due to e.g. reboot or missed group rekeying. This is further discussed in Section 6.

- The Client wishes to (re-)get the public keys of other group members, e.g. if it is aware of new nodes joining the group after itself. This is further discussed in Section 7.

Additionally, the format of the payload of the Key Distribution Response (Section 4.2) can be reused for messages sent by the KDC to distribute updated group keying material, in case of a new node joining the group or of a current member leaving the group. The key management scheme used to send such messages could rely on, e.g., multicast in case of a new node joining or unicast in case of a node leaving the group.

Note that proof-of-possession to bind the access token to the Client is performed by using the proof-of-possession key bound to the access token for establishing secure communication between the Client and the KDC.

If the application requires backward security, the KDC SHALL generate new group keying material and securely distribute it to all the current group members, using the message format defined in this section. Application profiles may define alternative message formats.

4.1. Key Distribution Request

The Client sends a Key Distribution Request to the KDC. This corresponds to a CoAP POST request to the endpoint in the KDC associated to the group to join. The endpoint in the KDC is associated to the 'scope' value of the Authorization Request/Response. The payload of this request is a CBOR map which MUST contain the following fields:

- 'type', encoded as a CBOR int, with value 1 ("key distribution").

Additionally, the CBOR map in the payload MAY contain the following fields, which, if included, MUST have the corresponding values:

- 'scope', with value the specific resource that the Client is authorized to access (i.e. group or topic identifier) and role(s), encoded as in Section 3.1.
o 'get_pub_keys', if the Client wishes to receive the public keys of the other nodes in the group from the KDC. The value is an empty CBOR array. This parameter may be present if the KDC stores the public keys of the nodes in the group and distributes them to the Client; it is useless to have here if the set of public keys of the members of the group is known in another way, e.g. it was provided by the AS.

o 'client_credential', with value the public key or certificate of the Client, encoded as a CBOR byte string. If the KDC is managing (collecting from/distributing to the Client) the public keys of the group members, this field contains the public key of the Client. The default encoding for public keys is COSE Keys. Alternative specific encodings of this parameter MAY be defined in applications of this specification.

o 'client_credential_verify', encoded as a CBOR byte string. This parameter contains a signature computed by the Client over the nonce N received from the KDC in the 2.01 (Created) response to the token POST request (see Section 3.3). The Client computes the signature by using its own private key, whose corresponding public key is either directly specified in the 'client_credential' parameter or included in the certificate specified in the 'client_credential' parameter. This parameter MUST be present if the 'client_credential' parameter is present.

o 'pub_keys_repos', can be present if a certificate is present in the 'client_credential' field, with value a list of public key repositories storing the certificate of the Client. This parameter is encoded as a CBOR array of CBOR text strings, each of which specifies the URI of a key repository.

4.2. Key Distribution Response

The KDC verifies that the 'scope' received in the Key Distribution Request, if present, is a subset of the 'scope' stored in the access token associated to this client. If verification fails, the KDC MUST respond with a 4.01 (Unauthorized) error message.

If the Key Distribution Request is not formatted correctly (e.g. no 'scope' field present while expected, or unknown fields present), the KDC MUST respond with 4.00 (Bad Request) error message.

If verification succeeds, the KDC sends a Key Distribution success Response to the Client. The Key Distribution success Response corresponds to a 2.01 Created message. The payload of this response is a CBOR map, which MUST contain:
o 'kty', identifying the key type of the 'key' parameter. The set of values can be found in the "Key Type" column of the "ACE Groupcomm Key" Registry. Implementations MUST verify that the key type matches the application profile being used, if present, as registered in the "ACE Groupcomm Key" registry.

o 'key', containing the keying material for the group communication, or information required to derive it.

The exact format of the 'key' value MUST be defined in applications of this specification. Additionally, documents specifying the key format MUST register it in the "ACE Groupcomm Key" registry, including its name, type and application profile to be used with, as defined in the "ACE Groupcomm Key" registry, defined in Section 11.5.

+----------+----------------+---------+-------------------------+
| Name     | Key Type Value | Profile | Description             |
+----------+----------------+---------+-------------------------+
| Reserved | 0              |         | This value is reserved   |
+----------+----------------+---------+-------------------------+

Figure 5: Key Type Values

Optionally, the Key Distribution Response MAY contain the following parameters, which, if included, MUST have the corresponding values:

o 'profile', with value a CBOR integer that MUST be used to uniquely identify the application profile for group communication. The value MUST be registered in the "ACE Groupcomm Profile" Registry.

o 'exp', with value the expiration time of the keying material for the group communication, encoded as a CBOR unsigned integer or floating-point number. This field contains a numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds, analogous to what specified in Section 2 of [RFC7519].

o 'pub_keys', may only be present if 'get_pub_keys' was present in the Key Distribution Request. This parameter is a CBOR byte string, which encodes the public keys of all the group members paired with the respective member identifiers. The default encoding for public keys is COSE Keys, so the default encoding for 'pub_keys' is a CBOR byte string wrapping a COSE_KeySet (see [RFC8152]), which contains the public keys of all the members of the group. In particular, each COSE Key in the COSE_KeySet includes the identifier of the corresponding group member as value of its 'kid' key parameter. Alternative specific encodings of
this parameter MAY be defined in applications of this specification.

- ‘group_policies’, with value a CBOR map, whose entries specify how the group handles specific management aspects. These include, for instance, approaches to achieve synchronization of sequence numbers among group members. The elements of this field are registered in the "ACE Groupcomm Policy" Registry. This specification defines the two elements "Sequence Number Synchronization Method" and "Key Update Check Interval", which are summarized in Figure 6. Application profiles that build on this document MUST specify the exact content format of included map entries.

<table>
<thead>
<tr>
<th>Name</th>
<th>CBOR label</th>
<th>CBOR type</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Number Synchronization Method</td>
<td>TBD1</td>
<td>tstr/int</td>
<td>Method for a recipient node to synchronize with sequence numbers of a sender node. Its value is taken from the ‘Value’ column of the Sequence Number Synchronization Method registry.</td>
<td>[[this document]]</td>
</tr>
<tr>
<td>Key Update Check Interval</td>
<td>TBD2</td>
<td>int</td>
<td>Polling interval in seconds, to check for new keying material at the KDC.</td>
<td>[[this document]]</td>
</tr>
</tbody>
</table>

Figure 6: ACE Groupcomm Policies

- ‘mgt_key_material’, encoded as a CBOR byte string and containing the administrative keying material to participate in the group rekeying performed by the KDC. The exact format and content depend on the specific rekeying scheme used in the group, which may be specified in the application profile.

Specific application profiles that build on this document need to specify how exactly the keying material is used to protect the group communication.
5. Removal of a Node from the Group

This section describes at a high level how a node can be removed from the group.

If the application requires forward security, the KDC SHALL generate new group keying material and securely distribute it to all the current group members but the leaving node, using the message format defined in Section 4.2. Application profiles may define alternative message formats.

5.1. Expired Authorization

If the AS provides Token introspection (see Section 5.7 of [I-D.ietf-ace-oauth-authz]), the KDC can optionally use and check whether:

- the node is not authorized anymore;
- the access token is still valid, upon its expiration.

Either case, once aware that a node is not authorized anymore, the KDC has to remove the unauthorized node from the list of group members, if the KDC keeps track of that.

5.2. Request to Leave the Group

A node can actively request to leave the group. In this case, the Client can send a request formatted as follows to the KDC, to abandon the group. The client MUST use the protected channel established with ACE, mentioned in Section 4.

To request to leave a group, the client MUST send a CoAP POST request to the endpoint in the KDC associated to the group to leave (same endpoint used in Section 4.1 for Key Distribution requests). The payload of this Leave Request is a CBOR map which MUST contain:

- 'type', encoded as a CBOR int, with value 2 ("leave").
- 'scope', with value the specific resource that the Client is authorized to access (i.e. group or topic identifier) and wants to leave, encoded as in Section 3.1. The 'role' field is omitted.

Note that the 'role' field is omitted since such a request should only be used to leave a group altogether. If the leaving node wants to be part of a group with fewer roles, it does not need to communicate that to the KDC, and can simply stop acting according to such roles.
If the Leave Request is such that the KDC cannot extract all the necessary information to understand and process it correctly (e.g. no ‘scope’ field present), the KDC MUST respond with a 4.00 (Bad Request) error message. Otherwise, the KDC MUST remove the leaving node from the list of group members, if the KDC keeps track of that.

Note that, after having left the group, a node may wish to join it again. Then, as long as the node is still authorized to join the group, i.e. it has a still valid access token, it can re-request to join the group directly to the KDC without needing to retrieve a new access token from the AS. This means that the KDC needs to keep track of nodes with valid access tokens, before deleting all information about the leaving node.

6. Retrieval of New or Updated Keying Material

A node stops using the group keying material upon its expiration, according to the ‘exp’ parameter specified in the retained COSE Key. Then, if it wants to continue participating in the group communication, the node has to request new updated keying material to the KDC. In this case, and depending on what part of the keying material is expired, the client may need to communicate to the KDC its need for that part to be renewed: for example, if the Client uses an individual key to protect outgoing traffic and has to renew it, the node may request a new one, or new input material to derive it, without renewing the whole group keying material.

The Client may perform the same request to the KDC also upon receiving messages from other group members without being able to correctly decrypt them. This may be due to a previous update of the group keying material (rekeying) triggered by the KDC, that the Client was not able to receive or decrypt.

Note that policies can be set up so that the Client sends a request to the KDC only after a given number of unsuccessfully decrypted incoming messages. It is application dependent and pertaining to the particular message exchange (e.g. [I-D.ietf-core-oscore-groupcomm]) to set up policies that instruct clients to retain unsuccessfully decrypted messages and for how long, so that they can be decrypted after getting updated keying material, rather than just considered non valid messages to discard right away.

The same request could also be sent by the client without being triggered by a failed decryption of a message, if the client wants to confirm that it has the latest group keying material. If that is the case, the client will receive from the KDC the same group keying material it has in memory.
Note that the difference between the keying material renewal request and the keying material update request is that the first one triggers the KDC to produce new keying material for that node, while the second one only triggers distribution (the renewal might have happened independently, because of expiration). Once a node receives new individual keying material, other group members may need to use the update keying material request to retrieve it.

Alternatively, the re-distribution of keying material can be initiated by the KDC, which e.g.:

- Can maintain an Observable resource to send notifications to Clients when the keying material is updated. Such a notification would have the same payload as the Key Re-Distribution Response defined in Section 6.2.
- Can send the payload of the Key Re-Distribution Response as one or multiple multicast requests to the members of the group, using secure rekeying schemes such as [RFC2093][RFC2094][RFC2627].
- Can send unicast requests to each Client over a secure channel, with the Key Re-Distribution Response as payload.
- Can act as a publisher in a pub-sub scenario, and update the keying material by publishing on a specific topic on a broker, which all the members of the group are subscribed to.

Note that these methods of KDC-initiated key re-distribution have different security properties and require different security associations.

6.1. Key Re-Distribution Request

To request a re-distribution of keying material, the Client sends a shortened Key Distribution Request to the KDC (Section 4.1), formatted as follows. The payload MUST contain the following fields:

- ‘type’, encoded as a CBOR int, with value 3 (“update key”) if the request is intended to retrieve updated group keying material, and 4 (“new”) if the request is intended for the KDC to produce and provide new individual keying material for the Client.
- ‘scope’, which contains only the identifier of the specific group or topic, encoded as in Section 3.1. That is, the role field is not present.
6.2. Key Re-Distribution Response

The KDC receiving a Key Re-Distribution Request MUST check that it is storing a valid access token from that client for that scope.

If that is not the case, i.e. it does not store the token or the token is not valid for that client for the scope requested, the KDC MUST respond with a 4.01 (Unauthorized) error message. Analogously to Section 4.2, if the Key Re-Distribution Request is not formatted correctly (e.g. no ‘scope’ field present, or unknown fields present), the KDC MUST respond with a 4.00 (Bad Request) error message.

Otherwise, the KDC replies to the Client with a Key Distribution Response, which MUST include the ‘kty’, ‘key’ and ‘exp’ parameters specified in Section 4.2. The Key Distribution Response MAY also include the ‘profile’, ‘group_policies’ and ‘mgt_key_material’ parameters specified in Section 4.2.

Note that this response might simply re-provide the same keying material currently owned by the Client, if it has not been renewed.

7. Retrieval of Public Keys for Group Members

In case the KDC maintains the public keys of group members, a node in the group can contact the KDC to request public keys of either all group members or a specified subset, using the messages defined below.

Figure 7 gives an overview of the exchange described above.

```
Client                     KDC
|                            |
|---- Public Key Request: POST /group-id ---> |
|                            |
|<--- Public Key Response: 2.01 (Created) ---|
```

Figure 7: Message Flow of Public Key Request-Response

Note that these messages can be combined with the Key Re-Distribution messages in Section 6, to request at the same time the keying material and the public keys. In this case, either a new endpoint at the KDC may be used, or additional information needs to be sent in the request payload, to distinguish these combined messages from the Public Key messages described below, since they would be identical otherwise.
7.1. Public Key Request

To request public keys, the Client sends a shortened Key Distribution Request to the KDC (Section 4.1), formatted as follows. The payload of this request MUST contain the following fields:

- 'type', encoded as a CBOR int, with value 5 ("pub keys").
- 'get_pub_keys', which has as value a CBOR array including either:
  - no elements, i.e. an empty array, in order to request the public key of all current group members; or
  - N elements, each of which is the identifier of a group member encoded as a CBOR byte string, in order to request the public key of the specified nodes.
- 'scope', which contains only the identifier of the specific group or topic, encoded as in Section 3.1. That is, the role field is not present.

7.2. Public Key Response

The KDC replies to the Client with a Key Distribution Response containing only the ‘pub_keys’ parameter, as specified in Section 4.2. The payload of this response contains the following field:

- 'pub_keys', which contains either:
  - the public keys of all the members of the group, if the ‘get_pub_keys’ parameter of the Public Key request was an empty array; or
  - the public keys of the group members with the identifiers specified in the ‘get_pub_keys’ parameter of the Public Key request.

The KDC may enforce one of the following policies, in order to handle possible identifiers that are included in the ‘get_pub_keys’ parameter of the Public Key request but are not associated to any current group member.

- The KDC silently ignores those identifiers.
- The KDC retains public keys of group members for a given amount of time after their leaving, before discarding them. As long as such
public keys are retained, the KDC provides them to a requesting Client.

Either case, a node that has left the group should not expect any of its outgoing messages to be successfully processed, if received after its leaving, due to a possible group rekeying occurred before the message reception.

8. ACE Groupcomm Parameters

This specification defines a number of fields used during the message exchange. The table below summarizes them, and specifies the CBOR key to use instead of the full descriptive name.
<table>
<thead>
<tr>
<th>Name</th>
<th>CBOR Key</th>
<th>CBOR Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope</td>
<td>TBD</td>
<td>array</td>
</tr>
<tr>
<td>get_pub_keys</td>
<td>TBD</td>
<td>array</td>
</tr>
<tr>
<td>client_cred</td>
<td>TBD</td>
<td>byte string</td>
</tr>
<tr>
<td>client_cred_verify</td>
<td>TBD</td>
<td>byte string</td>
</tr>
<tr>
<td>pub_keys_repos</td>
<td>TBD</td>
<td>array</td>
</tr>
<tr>
<td>kty</td>
<td>TBD</td>
<td>int / byte string</td>
</tr>
<tr>
<td>key</td>
<td>TBD</td>
<td>see &quot;ACE Groupcomm Key&quot; Registry</td>
</tr>
<tr>
<td>profile</td>
<td>TBD</td>
<td>int</td>
</tr>
<tr>
<td>exp</td>
<td>TBD</td>
<td>int / float</td>
</tr>
<tr>
<td>pub_keys</td>
<td>TBD</td>
<td>byte string</td>
</tr>
<tr>
<td>group_policies</td>
<td>TBD</td>
<td>map</td>
</tr>
<tr>
<td>mgt_key_material</td>
<td>TBD</td>
<td>byte string</td>
</tr>
<tr>
<td>type</td>
<td>TBD</td>
<td>int</td>
</tr>
</tbody>
</table>

9. ACE Groupcomm Request Type

This specification defines a number of types of requests. The table below summarizes them.
10. Security Considerations

When a Client receives a message from a sender for the first time, it needs to have a mechanism in place to avoid replay, e.g. Appendix B.2 of [I-D.ietf-core-object-security].

The KDC must renew the group keying material upon its expiration.

The KDC should renew the keying material upon group membership change, and should provide it to the current group members through the rekeying scheme used in the group.

The KDC may enforce a rekeying policy that takes into account the overall time required to rekey the group, as well as the expected rate of changes in the group membership.

That is, the KDC may not rekey the group at every membership change, for instance if members’ joining and leaving occur frequently and performing a group rekeying takes too long. Instead, the KDC may rekey the group after a minimum number of group members have joined or left within a given time interval, or during predictable network inactivity periods.

However, this would result in the KDC not constantly preserving backward and forward security. In fact, newly joining group members could be able to access the keying material used before their joining, and thus could access past group communications. Also, until the KDC performs a group rekeying, the newly leaving nodes would still be able to access upcoming group communications that are protected with the keying material that has not yet been updated.
10.1. Update of Keying Material

A group member can receive a message shortly after the group has been rekeyed, and new keying material has been distributed by the KDC. In the following two cases, this may result in misaligned keying material between the group members.

In the first case, the sender protects a message using the old keying material. However, the recipient receives the message after having received the new keying material, hence not being able to correctly process it. A possible way to ameliorate this issue is to preserve the old, recent, keying material for a maximum amount of time defined by the application. By doing so, the recipient can still try to process the received message using the old retained keying material as second attempt. Note that a former (compromised) group member can take advantage of this by sending messages protected with the old retained keying material. Therefore, a conservative application policy should not admit the storage of old keying material.

In the second case, the sender protects a message using the new keying material, but the recipient receives that request before having received the new keying material. Therefore, the recipient would not be able to correctly process the request and hence discards it. If the recipient receives the new keying material shortly after that and the sender endpoint uses CoAP retransmissions, the former will still be able to receive and correctly process the message. In any case, the recipient should actively ask the KDC for an updated keying material according to an application-defined policy, for instance after a given number of unsuccessfully decrypted incoming messages.

10.2. Block-Wise Considerations

If the block-wise options [RFC7959] are used, and the keying material is updated in the middle of a block-wise transfer, the sender of the blocks just changes the keying material to the updated one and continues the transfer. As long as both sides get the new keying material, updating the keying material in the middle of a transfer will not cause any issue. Otherwise, the sender will have to transmit the message again, when receiving an error message from the recipient.

Compared to a scenario where the transfer does not use block-wise, depending on how fast the keying material is changed, the nodes might consume a larger amount of the network resending the blocks again and again, which might be problematic.
11. IANA Considerations

This document has the following actions for IANA.

11.1. ACE Authorization Server Request Creation Hints Registry

IANA is asked to register the following entries in the "ACE Authorization Server Request Creation Hints" Registry defined in Section 8.1 of [I-D.ietf-ace-oauth-authz].

- Name: sign_info
  - CBOR Key: TBD (range -256 to 255)
  - Value Type: any
  - Reference: [[[This specification]]]

- Name: pub_key_enc
  - CBOR Key: TBD (range -256 to 255)
  - Value Type: integer
  - Reference: [[[This specification]]]

11.2. ACE Public Key Encoding Registry

This specification establishes the "ACE Public Key Encoding" IANA Registry. The Registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.9. It should be noted that, in addition to the expert review, some portions of the Registry require a specification, potentially a Standards Track RFC, be supplied as well.

The columns of this Registry are:

- Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

- Value: The value to be used to identify this public key encoding. This value MUST be unique. The value can be a positive or a negative integer. Integer values between 0 and 255 are designated as Standards Track Document required. Integer values from 256 to 65535 are designated as Specification Required. Integer values of
greater than 65535 are designated as expert review. Integer values less than -65536 are marked as private use.

- **Description**: This field contains a brief description for this public key encoding.

- **Reference**: This field contains a pointer to the public specification providing the public key encoding, if one exists.

The value 0 is to be marked as "Reserved".

### 11.3. ACE Groupcomm Parameters Registry

This specification establishes the "ACE Groupcomm Parameters" IANA Registry. The Registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.9.

The columns of this Registry are:

- **Name**: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

- **CBOR Key**: This is the value used as CBOR key of the item. These values MUST be unique. The value can be a positive integer, a negative integer, or a string.

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

- **Reference**: This contains a pointer to the public specification for the format of the item, if one exists.

This Registry has been initially populated by the values in Section 8. The specification column for all of these entries will be this document.

### 11.4. Ace Groupcomm Request Type Registry

This specification establishes the "ACE Groupcomm Request Type" IANA Registry. The Registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.9.

The columns of this Registry are:
Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

Value: This is the value used to identify the request. These values MUST be unique. The value must be a positive integer.

Reference: This contains a pointer to the public specification for the format of the item, if one exists.

This Registry has been initially populated by the values in Section 9. The reference column for all of these entries will be this document. The value 0 is to be marked as "Reserved".

11.5. ACE Groupcomm Key Registry

This specification establishes the "ACE Groupcomm Key" IANA Registry. The Registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.9.

The columns of this Registry are:

Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

Key Type Value: This is the value used to identify the keying material. These values MUST be unique. The value can be a positive integer, a negative integer, or a string.

Profile: This field may contain one or more descriptive strings of application profiles to be used with this item. The values should be taken from the Name column of the "ACE Groupcomm Profile" Registry.

Description: This field contains a brief description of the keying material.

References: This contains a pointer to the public specification for the format of the keying material, if one exists.

This Registry has been initially populated by the values in Figure 5. The specification column for all of these entries will be this document.
11.6. ACE Groupcomm Profile Registry

This specification establishes the "ACE Groupcomm Profile" IANA Registry. The Registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.9. It should be noted that, in addition to the expert review, some portions of the Registry require a specification, potentially a Standards Track RFC, be supplied as well.

The columns of this Registry are:

- Name: The name of the application profile, to be used as value of the profile attribute.
- Description: Text giving an overview of the application profile and the context it is developed for.
- CBOR Value: CBOR abbreviation for the name of this application profile. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.
- Reference: This contains a pointer to the public specification of the abbreviation for this application profile, if one exists.

11.7. ACE Groupcomm Policy Registry

This specification establishes the "ACE Groupcomm Policy" IANA Registry. The Registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.9. It should be noted that, in addition to the expert review, some portions of the Registry require a specification, potentially a Standards Track RFC, be supplied as well.

The columns of this Registry are:

- Name: The name of the group communication policy.
- CBOR label: The value to be used to identify this group communication policy. 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: the CBOR type used to encode the value of this group communication policy.
- Description: This field contains a brief description for this group communication policy.
- Reference: This field contains a pointer to the public specification providing the format of the group communication policy, if one exists.

This registry will be initially populated by the values in Figure 6.

11.8. Sequence Number Synchronization Method Registry

This specification establishes the "Sequence Number Synchronization Method" IANA Registry. The Registry has been created to use the "Expert Review Required" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.9. It should be noted that, in addition to the expert review, some portions of the Registry require a specification, potentially a Standards Track RFC, be supplied as well.

The columns of this Registry are:

- Name: The name of the sequence number synchronization method.
- Value: The value to be used to identify this sequence number synchronization method.
- Description: This field contains a brief description for this sequence number synchronization method.
- Reference: This field contains a pointer to the public specification describing the sequence number synchronization method.

11.9. Expert Review Instructions

The IANA Registries established in this document are 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:

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

- Specifications are required for the standards 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.

- 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.

12. References

12.1. Normative References

[I-D.ietf-ace-oauth-authz]

[I-D.ietf-ace-oauth-params]
12.2. Informative References

[I-D.dijk-core-groupcomm-bis]

[I-D.ietf-ace-dtls-authorize]

[I-D.ietf-ace-mqtt-tls-profile]

[I-D.ietf-ace-oscore-profile]
[I-D.ietf-core-coap-pubsub]

[I-D.ietf-core-object-security]


Appendix A. Requirements on Application Profiles

This section lists the requirements on application profiles of this specification, for the convenience of application profile designers.

- Specify the communication protocol the members of the group must use (e.g., multicast CoAP).

- Specify the security protocol the group members must use to protect their communication (e.g., group OSCORE). This must provide encryption, integrity and replay protection.

- Specify the encoding and value of the identifier of group or topic and role of 'scope' (see Section 3.1).

- Specify and register the application profile identifier (see Section 4.1).

- Specify the acceptable values of 'kty' (see Section 4.2).

- Specify the format and content of 'group_policies' entries (see Section 4.2).

- Optionally, specify the format and content of 'mgt_key_material' (see Section 4.2).

- Optionally, specify transport profile of ACE [I-D.ietf-ace-oauth-authz] to use between Client and KDC.

- Optionally, specify the encoding of public keys, of 'client_cred', and of 'pub_keys' if COSE_Keys are not used (see Section 4.2).

- Optionally, specify the acceptable values for parameters related to signature algorithm and signature keys: 'sign_alg', 'sign_parameters', 'sign_key_parameters', 'pub_key_enc' (see Section 3.3).

- Optionally, specify the negotiation of parameter values for signature algorithm and signature keys, if 'sign_info' and 'pub_key_enc' are not used (see Section 3.3).

Appendix B. Document Updates

RFC EDITOR: PLEASE REMOVE THIS SECTION.
B.1. Version -01 to -02

- Editorial fixes.
- Distinction between transport profile and application profile (Section 1.1).
- New parameters 'sign_info' and 'pub_key_enc' to negotiate parameter values for signature algorithm and signature keys (Section 3.3).
- New parameter 'type' to distinguish different Key Distribution Request messages (Section 4.1).
- New parameter 'client_cred_verify' in the Key Distribution Request to convey a Client signature (Section 4.1).
- Encoding of 'pub_keys_repos' (Section 4.1).
- Encoding of 'mgt_key_material' (Section 4.1).
- Improved description on retrieval of new or updated keying material (Section 6).
- Encoding of 'get_pub_keys' in Public Key Request (Section 7.1).
- Extended security considerations (Sections 10.1 and 10.2).
- New "ACE Public Key Encoding" IANA Registry (Section 11.2).
- New "ACE Groupcomm Parameters" IANA Registry (Section 11.3), populated with the entries in Section 8.
- New "Ace Groupcomm Request Type" IANA Registry (Section 11.4), populated with the values in Section 9.
- New "ACE Groupcomm Policy" IANA Registry (Section 11.7) populated with two entries "Sequence Number Synchronization Method" and "Key Update Check Interval" (Section 4.2).
- Improved list of requirements for application profiles (Appendix A).

B.2. Version -00 to -01

- Changed name of 'req_aud' to 'audience' in the Authorization Request (Section 3.1).
o Defined error handling on the KDC (Sections 4.2 and 6.2).

o Updated format of the Key Distribution Response as a whole (Section 4.2).

o Generalized format of ‘pub_keys’ in the Key Distribution Response (Section 4.2).

o Defined format for the message to request leaving the group (Section 5.2).

o Renewal of individual keying material and methods for group rekeying initiated by the KDC (Section 6).

o CBOR type for node identifiers in ‘get_pub_keys’ (Section 7.1).

o Added section on parameter identifiers and their CBOR keys (Section 8).

o Added request types for requests to a Join Response (Section 9).

o Extended security considerations (Section 10).

o New IANA registries "ACE Groupcomm Key Registry", "ACE Groupcomm Profile Registry", "ACE Groupcomm Policy Registry" and "Sequence Number Synchronization Method Registry" (Section 11).

o Added appendix about requirements for application profiles of ACE on group communication (Appendix A).

Acknowledgments

The following individuals were helpful in shaping this document: Ben Kaduk, John Mattsson, Jim Schaad, Ludwig Seitz, Goeran Selander and Peter van der Stok.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; and by the EIT-Digital High Impact Initiative ACTIVE.

Authors’ Addresses
CoAP Pub-Sub Profile for Authentication and Authorization for Constrained Environments (ACE)
draft-palombini-ace-coap-pubsub-profile-05

Abstract

This specification defines a profile for authentication and authorization for publishers and subscribers in a pub-sub setting scenario in a constrained environment, using the ACE framework. This profile relies on transport layer or application layer security to authorize the publisher to the broker. Moreover, it relies on application layer security for publisher-broker and subscriber-broker communication.

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 January 9, 2020.

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
The publisher-subscriber setting allows for devices with limited reachability to communicate via a broker that enables store-and-forward messaging between the devices. The pub-sub scenario using the Constrained Application Protocol (CoAP) is specified in [I-D.ietf-core-coap-pubsub]. This document defines a way to authorize nodes in a CoAP pub-sub type of setting, using the ACE framework [I-D.ietf-ace-oauth-authz], and to provide the keys for protecting the communication between these nodes.

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 RFC 2119 [RFC2119].

Readers are expected to be familiar with the terms and concepts described in [I-D.ietf-ace-oauth-authz], [I-D.ietf-ace-key-groupcomm] and [I-D.ietf-core-coap-pubsub]. In particular, analogously to [I-D.ietf-ace-oauth-authz], terminology for entities in the architecture such as Client (C), Resource Server (RS), and
Authorization Server (AS) is defined in OAuth 2.0 [RFC6749] and [I-D.ietf-ace-actors], and terminology for entities such as the Key Distribution Center (KDC) and Dispatcher in [I-D.ietf-ace-key-groupcomm].

2. Profile Overview

The objective of this document is to specify how to authorize nodes, provide keys, and protect a CoAP pub-sub communication, as described in [I-D.ietf-core-coap-pubsub], using [I-D.ietf-ace-key-groupcomm], which itself expands the Ace framework ([I-D.ietf-ace-oauth-authz]), and profiles ([I-D.ietf-ace-dtls-authorize], [I-D.ietf-ace-oscore-profile]).

The architecture of the scenario is shown in Figure 1.

Figure 1: Architecture CoAP pubsub with Authorization Servers

The RS is the broker, which contains the topic. This node corresponds to the Dispatcher, in [I-D.ietf-ace-key-groupcomm]. The AS1 hosts the policies about the Broker: what endpoints are allowed to Publish on the Broker. The Clients access this node to get write access to the Broker. The AS2 hosts the policies about the topic: what endpoints are allowed to access what topic. This node represents both the AS and Key Distribution Center roles from [I-D.ietf-ace-key-groupcomm].

There are four phases, the first three can be done in parallel.
1. The Publisher requests publishing access to the Broker at the AS1, and communicates with the Broker to set up security.

2. The Publisher requests access to a specific topic at the AS2.

3. The Subscriber requests access to a specific topic at the AS2.

4. The Publisher and the Subscriber securely post to and get publications from the Broker.

This exchange aims at setting up 2 different security associations: on the one hand, the Publisher has a security association with the Broker, to protect the communication and securely authorize the Publisher to publish on a topic (Security Association 1). On the other hand, the Publisher has a security association with the Subscriber, to protect the publication content itself (Security Association 2). The Security Association 1 is set up using AS1 and a profile of [I-D.ietf-ace-oauth-authz], the Security Association 2 is set up using AS2 and [I-D.ietf-ace-key-groupcomm].

Note that, analogously to the Publisher, the Subscriber can also set up an additional security association with the Broker, using an AS, in the same way the Publisher does with AS1. In this case, only authorized Subscribers would be able to get notifications from the Broker. The overhead would be that each Subscriber should access the AS and get all the information to start a secure exchange with the Broker.

```
+------------+             +------------+              +------------+
|   CoAP     |             |   CoAP     |              |    CoAP    |
|  Client -  |             |  Server -  |              |  Client -  |
| Publisher  |             |   Broker   |              | Subscriber |
|------------+             |------------+              |------------+
|            |             |            |              |
|            |             |            |              |
|            |             |            |              |
|            |             |            |              |

: : :
: : '------ Security ------'
: : Association 1
: :

Association 2
```

Note that AS1 and AS2 might either be co-resident or be 2 separate physical entities, in which case access control policies must be exchanged between AS1 and AS2, so that they agree on rights for joining nodes about specific topics. How the policies are exchanged is out of scope for this profile.
3. coap_pubsub_app Profile

This profile uses [I-D.ietf-ace-key-groupcomm], which expands the ACE framework. This document specifies which exact parameters from [I-D.ietf-ace-key-groupcomm] have to be used, and the values for each parameter.

The Publisher and the Subscriber map to the Client in [I-D.ietf-ace-key-groupcomm], the AS2 maps to the AS and to the KDC, the Broker maps to the Dispatcher.

Note that both publishers and subscribers use the same profile, called "coap_pubsub_app".

3.1. Retrieval of COSE Key for protection of content

This phase is common to both Publisher and Subscriber. To maintain the generality, the Publisher or Subscriber is referred as Client in this section.

```
Client                Broker               AS2


[<-- AS1, AS2 Information ---] [------ Pub Key Format Negotiation Request --->] [-- Authorization + Key Distribution Request --->] <- Authorization + Key Distribution Response --

```

Figure 2: B: Access request - response

Complementary to what is defined in [I-D.ietf-ace-oauth-authz] (Section 5.1.1), to determine the AS2 in charge of a topic hosted at the Broker, the Broker MAY send the address of both the AS in charge of the topic back to the Client in the ‘AS’ parameter in the AS Information, as a response to an Unauthorized Resource Request (Section 5.1.2). An example using CBOR diagnostic notation is given below:

```
Palombini                Expires January 9, 2020                [Page 5]`
After retrieving the AS2 address, the Client MAY send a Pub Key Format Negotiation Request to the AS, in order to request necessary information concerning the public keys in the group, as well as concerning the algorithm and related parameters for computing signatures in the group. This request is a subset of the Token Post request defined in Section 3.3 of [I-D.ietf-ace-key-groupcomm], specifically including the parameters ‘sign_info’ and ‘pub_key_enc’. The AS MUST respond with the response defined in Section 3.3 of [I-D.ietf-ace-key-groupcomm], specifically including the same parameters ‘sign_info’ and ‘pub_key_enc’.

After that, the Client sends an Authorization + Key Distribution Request, which is an Authorization Request merged with a Key Distribution Request, as described in [I-D.ietf-ace-key-groupcomm], Sections 3.1 and 4.1. The reason for merging these two messages is that the AS2 is both the AS and the KDC, in this setting, so the Authorization Response and the Post Token message are not necessary.

More specifically, the Client sends a POST request to the /token endpoint on AS2, with Content-Format = "application/ace+cbor" that MUST contain in the payload (formatted as a CBOR map):

- the following fields from the Authorization Request (Section 3.1 of [I-D.ietf-ace-key-groupcomm]):
  - ‘grant_type’ set to "client_credentials",
  - Optionally, if needed, other additional parameters such as ‘client_id’

- the following fields from the Key Distribution Request (Section 4.1 of [I-D.ietf-ace-key-groupcomm]):
  - ‘type’ set to 1 ("key distribution")
  - ‘client_cred’ parameter containing the Client’s public key formatted as a COSE_Key, if the Client needs to directly send that to the AS2,
  - ‘scope’ parameter set to a CBOR array containing:
+ the broker’s topic as first element, and
+ the string "publisher" if the client request to be a publisher, "subscriber" if the client request to be a subscriber, or a CBOR array containing both, if the client request to be both.

* ‘get_pub_keys’ parameter set to the empty array if the Client needs to retrieve the public keys of the other pubsub members

* OPTIONALLY, if needed, the ‘pub_keys_repos’ parameter

Note that the alg parameter in the ‘client_cred’ COSE_Key MUST be a signing algorithm, as defined in section 8 of [RFC8152].

Examples of the payload of a Authorization + Key Distribution Request are specified in Figure 5 and Figure 8.

The AS2 verifies that the Client is authorized to access the topic and, if the ‘client_cred’ parameter is present, stores the public key of the Client.

The AS2 response is an Authorization + Key Distribution Response, see Section 4.2 of [I-D.ietf-ace-key-groupcomm], with Content-Format = "application/ace+cbor". The payload (formatted as a CBOR map) MUST contain:

o the following fields from the Authorization Response (Section 3.2 of [I-D.ietf-ace-key-groupcomm]):

  * ‘profile’ set to "coap_pubsub_app", as specified in Section 8.1
  * OPTIONALLY ‘scope’, set to a CBOR array containing:

    + the broker’s topic as first element, and
    + the string "publisher" if the client is an authorized publisher, "subscriber" if the client is an authorized subscriber, or a CBOR array containing both, if the client is authorized to be both.

o the following fields from the Key Distribution Response (Section 4.2 of [I-D.ietf-ace-key-groupcomm]):

  * ‘kty’ identifies a key type "COSE_Key", as defined in Section 8.2.
* 'key', which contains a "COSE_Key" object (defined in [RFC8152]), containing:
  + 'kty' with value 4 (symmetric)
  + 'alg' with value defined by the AS2 (Content Encryption Algorithm)
  + 'Base IV' with value defined by the AS2
  + 'k' with value the symmetric key value
  + OPTIONALLY, 'kid' with an identifier for the key value
  * OPTIONALLY, exp with the expiration time of the key

* 'pub_keys', containing the public keys of all authorized signing members formatted as COSE_Keys, if the 'get_pub_keys' parameter was present and set to the empty array in the Authorization + Key Distribution Request

Examples for the response payload are detailed in Figure 6 and Figure 9.

4. Publisher

In this section, it is specified how the Publisher requests, obtains and communicates to the Broker the access token, as well as the retrieval of the keying material to protect the publication.
This is a combination of two independent phases:

- The establishment of a secure connection between Publisher and Broker, using an ACE profile such as DTLS [I-D.ietf-ace-dtls-authorize] or OSCORE [I-D.ietf-ace-oscore-profile]. (A)(C)

- The Publisher’s retrieval of keying material to protect the publication. (B)

In detail:

- (A) corresponds to the Access Token Request and Response between Publisher and Authorization Server to retrieve the Access Token and RS (Broker) Information. As specified, the Publisher has the role of a CoAP client, the Broker has the role of the CoAP server.

- (C) corresponds to the exchange between Publisher and Broker, where the Publisher sends its access token to the Broker and establishes a secure connection with the Broker. Depending on the Information received in (A), this can be for example DTLS handshake, or other protocols. Depending on the application, there may not be the need for this set up phase: for example, if OSCORE is used directly.

- (A) and (C) details are specified in the profile used.

- (B) corresponds to the retrieval of the keying material to protect the publication end-to-end with the subscribers (see Section 6.1),
and uses [I-D.ietf-ace-key-groupcomm]. The details are defined in
Section 3.1.

An example of the payload of an Authorization + Key Distribution
Request and corresponding Response for a Publisher is specified in
Figure 5 and Figure 6.

```json
{
    "grant_type": "client_credentials",
    "scope": ["Broker1/Temp", "publisher"],
    "type": 1,
    "client_id": "publisher1",
    "client_cred":
    {
        / COSE_Key /
        / type / 1 : 2, / EC2 /
        / kid / 2 : h'11' ,
        / alg / 3 : -7, / ECDSA with SHA-256 /
        / crv / -1 : 1, / P-256 /
        / x / -2 : h'65eda5a12577c2bae829437fe338701a10aa375e1bb5b5de108de439c08551d',
        / y /-3 : h'1e52ed75701163f7f9e40ddf9f341b3dc9ba860af7e0ca7ca7e9ee60084d19c'
    }
}
```

Figure 5: Authorization + Key Distribution Request payload for a
Publisher

```json
{
    "profile": "coap_pubsub_app",
    "kty": "COSE_Key",
    "key": {1: 4, 2: h'1234', 3: 12, 5: h'f389d14d7dc7', -1: h'02e2cc3a9b92855220f255fff1c615bc'}
}
```

Figure 6: Authorization + Key Distribution Response payload for a
Publisher

5. Subscriber

In this section, it is specified how the Subscriber retrieves the
keying material to protect the publication.
Step (D) between Subscriber and AS2 corresponds to the retrieval of the keying material to verify the publication end-to-end with the publishers (see Section 6.1). The details are defined in Section 3.1. This step is the same as (B) between Publisher and AS2 (Section 3.1), with the following differences:

- The Authorization + Key Distribution Request MUST NOT contain the 'client_cred parameter', the role element in the 'scope' parameter MUST be set to "subscriber". The Subscriber MUST have access to the public keys of all the Publishers; this MAY be achieved in the Authorization + Key Distribution Request by using the parameter 'get_pub_keys' set to empty array.

- The Authorization + Key Distribution Response MUST contain the 'pub_keys' parameter.

An example of the payload of an Authorization + Key Distribution Request and corresponding Response for a Subscriber is specified in Figure 8 and Figure 9.
{  
  "grant_type" : "client_credentials",
  "type" = 1,
  "scope" : ["Broker1/Temp", "subscriber"],
  "get_pub_keys" : [ ]
}

Figure 8: Authorization + Key Distribution Request payload for a Subscriber

{  
  "profile" : "coap_pubsub_app",
  "scope" : ["Broker1/Temp", "subscriber"],
  "kty" : "COSE_Key",
  "key" : {1: 4, 2: h'1234', 3: 12, 5: h'1f389d14d17dc7',
            -1: h'02e2cc3a9b92855220f255fff1c615bc'},
  "pub_keys" : [
    {  
      1 : 2, / type EC2 /
      2 : h'11', / kid /
      3 : -7, / alg ECDSA with SHA-256 /
      -1 : 1, / crv P-256 /
      -2 : h'65eda5a12577c2bae829437fe338701a10aaa375e1bb5b5de108de439c08551d', / x /
      -3 : h'1e52ed75701163f7f9e40ddf9f341b3dc9ba860af7e0ca7ca7e9ee5cd0084d19c' / y /
    }
  ]
}

Figure 9: Authorization + Key Distribution Response payload for a Subscriber

6. Pub-Sub Protected Communication

This section specifies the communication Publisher-Broker and Subscriber-Broker, after the previous phases have taken place. The operations of publishing and subscribing are defined in [I-D.ietf-core-coap-pubsub].

Palombini                Expires January 9, 2020               [Page 12]
The (E) message corresponds to the publication of a topic on the Broker. The publication (the resource representation) is protected with COSE ([RFC8152]). The (F) message is the subscription of the Subscriber, which is unprotected, unless a profile of ACE [I-D.ietf-ace-oauth-authz] is used between Subscriber and Broker. The (G) message is the response from the Broker, where the publication is protected with COSE.

The flow graph is presented below.

```
Publisher                  Broker                  Subscriber
| --- PUT /topic ---->     |                       | <--- GET /topic ------|
| protected with COSE     |                       | <--- response ------->|
|                           |                       | protected with COSE  |
```

Figure 11: (E), (F), (G): Example of protected communication

6.1. Using COSE Objects To Protect The Resource Representation

The Publisher uses the symmetric COSE Key received from AS2 in exchange B (Section 3.1) to protect the payload of the PUBLISH operation (Section 4.3 of [I-D.ietf-core-coap-pubsub]). Specifically, the COSE Key is used to create a COSE_Encrypt0 with algorithm specified by AS2. The Publisher uses the private key corresponding to the public key sent to the AS2 in exchange B (Section 3.1) to countersign the COSE Object as specified in Section 4.5 of [RFC8152]. The CoAP payload is replaced by the COSE object before the publication is sent to the Broker.

The Subscriber uses the kid in the countersignature field in the COSE object to retrieve the right public key to verify the countersignature. It then uses the symmetric key received from AS2 to verify and decrypt the publication received in the payload of the CoAP Notification from the Broker.
The COSE object is constructed in the following way:

- The protected Headers (as described in Section 3 of [RFC8152]) MAY contain the kid parameter, with value the kid of the symmetric COSE Key received in Section 3.1 and MUST contain the content encryption algorithm.

- The unprotected Headers MUST contain the Partial IV, with value a sequence number that is incremented for every message sent, and the counter signature that includes:
  * the algorithm (same value as in the asymmetric COSE Key received in (B)) in the protected header;
  * the kid (same value as the kid of the asymmetric COSE Key received in (B)) in the unprotected header;
  * the signature computed as specified in Section 4.5 of [RFC8152].

- The ciphertext, computed over the plaintext that MUST contain the CoAP payload.

The external_aad is an empty string.

An example is given in Figure 12
Figure 12: Example of COSE Object sent in the payload of a PUBLISH operation

The encryption and decryption operations are described in sections 5.3 and 5.4 of [RFC8152].

7. Security Considerations

In the profile described above, the Publisher and Subscriber use asymmetric crypto, which would make the message exchange quite heavy for small constrained devices. Moreover, all Subscribers must be able to access the public keys of all the Publishers to a specific topic to be able to verify the publications. Such a database could be set up and managed by the same entity having control of the topic, i.e. AS2.

An application where it is not critical that only authorized Publishers can publish on a topic may decide not to make use of the asymmetric crypto and only use symmetric encryption/MAC to confidentiality and integrity protect the publication, but this is not recommended since, as a result, any authorized Subscribers with access to the Broker may forge unauthorized publications without being detected. In this symmetric case the Subscribers would only need one symmetric key per topic, and would not need to know any information about the Publishers, that can be anonymous to it and the Broker.
Subscribers can be excluded from future publications through re-keying for a certain topic. This could be set up to happen on a regular basis, for certain applications. How this could be done is out of scope for this work.

The Broker is only trusted with verifying that the Publisher is authorized to publish, but is not trusted with the publications itself, which it cannot read nor modify. In this setting, caching of publications on the Broker is still allowed.

TODO: expand on security and privacy considerations

8. IANA Considerations

8.1. ACE Groupcomm Profile Registry

The following registrations are done for the "ACE Groupcomm Profile" Registry following the procedure specified in [I-D.ietf-ace-key-groupcomm].

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

Name: coap_pubsub_app

Description: Profile for delegating client authentication and authorization for publishers and subscribers in a pub-sub setting scenario in a constrained environment.

CBOR Key: TBD

Reference: [[This document]]

8.2. ACE Groupcomm Key Registry

The following registrations are done for the ACE Groupcomm Key Registry following the procedure specified in [I-D.ietf-ace-key-groupcomm].

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

Name: COSE_Key

Key Type Value: TBD

Palombini Expires January 9, 2020
Profile: coap_pubsub_app
Description: COSE_Key object

References: [RFC8152], [This document]

9. References

9.1. Normative References

[I-D.ietf-ace-key-groupcomm]

[I-D.ietf-ace-oauth-authz]

[I-D.ietf-core-coap-pubsub]


9.2. Informative References

[I-D.ietf-ace-actors]
Appendix A. Requirements on Application Profiles

This section lists the specifications on this profile based on the requirements defined in Appendix A of [I-D.ietf-ace-key-groupcomm]

- Specify the communication protocol the members of the group must use: CoAP pub/sub.
- Specify the security protocol the group members must use to protect their communication: Object Security of Content using COSE.
- Specify the encoding and value of the identifier of group or topic and role of 'scope': see Section 3.1).
- Specify and register the application profile identifier: "coap_pubsub_app", see Section 8.1.
- Specify the acceptable values of 'kty': "COSE_Key", see Section 3.1.
- Specify the format and content of 'group_policies' entries
- Optionally, specify the format and content of 'mgt_key_material': not defined
- Optionally, specify trandport profile of ACE [I-D.ietf-ace-oauth-authz] to use between Client and KDC: up to the application.
- Optionally, specify the encoding of public keys, of 'client_cred', and of 'pub_keys' if COSE_Keys are not used: COSE_Keys are used.
- Optionally, specify the acceptable values for parameters related to signature algorithm and signature keys: ‘sign_alg’,...
'sign_parameters', 'sign_key_parameters', 'pub_key_enc': not defined

- Optionally, specify the negotiation of parameter values for signature algorithm and signature keys, if 'sign_info' and 'pub_key_enc' are not used: not defined.

Acknowledgments

The author wishes to thank Ari Keraenen, John Mattsson, Ludwig Seitz, Goeran Selander, Jim Schaad and Marco Tiloca for the useful discussion and reviews that helped shape this document.

Author’s Address

Francesca Palombini
Ericsson

Email: francesca.palombini@ericsson.com
Abstract

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework. The profile uses Object Security for Constrained RESTful Environments (OSCORE) and/or Group OSCORE to provide communication security between a Client and (a group of) Resource Server(s). Furthermore, the profile uses (Group) OSCORE to provide server authentication, and OSCORE to achieve proof-of-possession for a key owned by the Client and bound to an OAuth 2.0 Access Token. Also, the profile provides proof-of-group-membership for the Client, by securely binding the pre-established Group OSCORE Security Context to the pairwise OSCORE Security Context newly established with the Resource Server.

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 January 7, 2020.
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 ................................................. 3
   1.1. Terminology ............................................ 4
2. Protocol Overview ........................................... 5
   2.1. Pre-Conditions ......................................... 7
   2.2. Access Token Retrieval ................................ 7
   2.3. Access Token Posting ................................... 7
   2.4. Secure Communication .................................. 9
3. Client-AS Communication ...................................... 9
   3.1. C-to-AS: POST to Token Endpoint ....................... 10
       3.1.1. ‘context_id’ Parameter ............................ 11
       3.1.2. ‘salt’ Parameter .................................. 12
   3.2. AS-to-C: Access Token .................................. 12
4. Client-RS Communication ...................................... 16
   4.1. C-to-RS POST to authz-info Endpoint ................... 17
   4.2. RS-to-C: 2.01 (Created) ................................ 17
   4.3. OSCORE Setup - Client Side .............................. 18
   4.4. OSCORE Setup - Resource Server Side ................... 20
   4.5. Access Rights Verification .............................. 21
5. Secure Communication with the AS ............................. 22
6. Discarding the Security Context ............................. 22
7. CBOR Mappings ............................................... 23
8. Security Considerations ..................................... 23
9. Privacy Considerations ...................................... 24
10. IANA Considerations ........................................ 24
    10.1. ACE Profile Registry .................................. 24
    10.2. OAuth Parameters Registry ............................. 24
    10.3. OAuth Parameters CBOR Mappings Registry ............. 25
11. References ................................................ 25
    11.1. Normative References ................................ 25
    11.2. Informative References ............................... 27
Appendix A. Profile Requirements ............................... 28
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 key material, which is typically provided upon joining the group.

For some instances of such applications, it may be just fine to enforce access control in a straightforward and plain fashion. That is, it is assumed that any Client authorized to join the group and to get the group key material, is also implicitly authorized as a group member to perform any action at any resource of any Server in the group. An example of an application where such implicit authorization might be used is a 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. An example of an application where a more fine-grained authorization approach is preferable is the 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; whereas 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 examples concern the enforcement of different sets of permissions in groups with sensor/actuator devices, e.g. thermostats, acting as Servers.

Hence, in this latter case, being a legitimate group member and having obtained the group key material does not imply any particular access rights. Thus, a more fine-grained access control model has to be enforced, e.g. by using the Authentication and Authorization for Constrained Environments (ACE) framework [I-D.ietf-ace-oauth-authz]. That is, a Client has to first obtain authorization credentials in the form of an OAuth 2.0 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 Server. However each of the current profiles of ACE defined in [I-D.ietf-ace-oscore-profile] [I-D.ietf-ace-dtls-authorize] [I-D.ietf-ace-mqtt-tls-profile] admits a single security protocol that cannot be used to protect group messages sent over IP multicast.

This document specifies a profile of ACE, where a Client uses CoAP [RFC7252] to communicate to a single Resource Server, or CoAP over IP multicast [RFC7390] [I-D.dijk-core-groupcomm-bis] to communicate to multiple Resource Servers that are members of a group and share a common set of resources. This profile uses two complementary security protocols to provide secure communication between the Client and the Resource Server(s).

That is, this document defines the use of either Object Security for Constrained RESTful Environments (OSCORE) [I-D.ietf-core-object-security] or Group OSCORE [I-D.ietf-core-oscore-groupcomm] to protect unicast requests addressed to a single Resource Server, as well as possible responses. Additionally, it defines the use of Group OSCORE to protect multicast requests sent to a group of Resource Servers, as well as possible individual responses. The Client and the Resource Servers need to have already joined an OSCORE group, for instance by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore] which is also based on ACE.

The Client authorizes its access to the Resource Server by using an Access Token, which is bound to a key (the proof-of-possession key). This profile uses OSCORE to achieve proof of possession, and OSCORE or Group OSCORE to achieve server authentication. Furthermore, this profile provides proof of Client’s membership to the correct OSCORE group, by securely binding the pre-established Group OSCORE Security Context to the pairwise OSCORE Security Context newly established between the Client and the Resource Server.

OSCORE specifies how to use CBOR Object Signing and Encryption (COSE) [RFC8152] to secure CoAP messages. Group OSCORE builds on OSCORE to support group communication, and ensures source authentication by means of digital countersignatures embedded in protected messages.

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 the CoAP protocol [RFC7252], as well as to the protection and processing of CoAP messages through OSCORE [I-D.ietf-core-object-security], also in group communication scenarios through Group OSCORE [I-D.ietf-core-oscore-groupcomm]. These include the concept of Group Manager, as the entity responsible for a set of groups where communications among members are secured with Group OSCORE.

This document also refers to "pairwise OSCORE Security Context", i.e. an OSCORE Security Context established between only one Client and one Resource Server, and used to communicate with OSCORE [I-D.ietf-core-object-security].

Readers are expected to be familiar with the terms and concepts described in the ACE framework for authentication and authorization [I-D.ietf-ace-oauth-authz], as well as in the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile]. 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".

2. Protocol Overview

This section provides an overview on how to use the ACE framework for authentication and authorization [I-D.ietf-ace-oauth-authz] to secure communications between a Client and a (set of) Resource Server(s) using OSCORE [I-D.ietf-core-object-security] and/or Group OSCORE [I-D.ietf-core-oscore-groupcomm].

An overview of the protocol flow for this profile is shown in Figure 1. In the figure, it is assumed that C, RS1 and RS2 have previously joined an OSCORE group with Group Identifier (gid) "abcd0000", and got assigned Sender ID (sid) "0", "1" and "2" in the group, respectively. It is also assumed that both RS1 and RS2 are associated with the same AS. For simplicity, the figure does not show the preliminary phase where C, R1 and R2 join the OSCORE group.
Figure 1: Protocol Overview.
2.1. Pre-Conditions

Using Group OSCORE requires both the Client and the Resource Servers to have previously joined an 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 may 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.

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) it wants to access on each of the RSs, using the /token endpoint, as specified in Section 5.6 of [I-D.ietf-ace-oauth-authz]. In a general case, it can be assumed that different RSs are associated to different ASs, even if the RSs are members of a same 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 include these pieces of information in the Access Token and in the Access Token response to the Client.

To gain knowledge of the AS in charge of a resource hosted at a RS, the Client MAY first send an initial Unauthorized Resource Request message to that RS. Then, the RS denies the request and replies to the Client by specifying the address of its AS, as defined 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 the Client and the AS, but TLS [RFC5246][RFC8446] or DTLS [RFC6347][I-D.ietf-tls-dtls13] MAY be used additionally or instead.

2.3. Access Token Posting

After having retrieved the Access Token from the AS, the Client generates a nonce N1 and posts both the Access 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 POST request with a 2.01 (Created) response with Content-Format = application/ace+cbor, which contains a nonce N2 in a CBOR map. Also, the RS concatenates N1 with N2, and further concatenates the result with the Group Identifier of the OSCORE group specified in the Access Token. The RS sets the ID Context in the pairwise OSCORE Security Context (see Section 3 of [I-D.ietf-core-object-security]) to N1 concatenated with N2 concatenated with the Group Identifier of the OSCORE group.

Then, the RS derives the complete pairwise OSCORE Security Context associated with the received Access Token, following Section 3.2 of [I-D.ietf-core-object-security]. During the derivation process, the RS uses the ID Context above, plus the parameters in the Access Token. The derivation process uses also the Master Secret of the OSCORE group, that the RS knows as a group member, as well as the Sender ID of the Client in the OSCORE group, which is specified in the Access Token. This ensures that the pairwise OSCORE Security Context is securely bound to the Group OSCORE Security Context of the OSCORE group.

Finally, the RS stores the association between the authorization information from the Access Token, and the Group Identifier of the OSCORE group together with the Sender ID of the Client in that group.

After having received the nonce N2, the Client sets the ID Context in its pairwise OSCORE Security Context (see Section 3 of [I-D.ietf-core-object-security]) to N1 concatenated with N2 concatenated with the Group Identifier of the OSCORE group. Then, the Client derives the complete pairwise OSCORE Security Context, following Section 3.2 of [I-D.ietf-core-object-security]. During the derivation process, the Client uses the ID Context above, plus the parameters received from the AS. The derivation process uses also the Master Secret of the OSCORE group, that the Client knows as a group member, as well as its own Sender ID in the OSCORE group.

When the Client communicates with the RS using the pairwise OSCORE Security Context, the RS achieves proof-of-possession of the credentials bound to the Access Token. Also, the RS verifies that the Client is a legitimate member 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 the exact group member with the same Sender ID associated to the Access Token. This occurs when verifying a request protected with Group OSCORE, since it embeds a countersignature computed also over the Client’s Sender ID included in the message.
2.4. Secure Communication

The Client can send a request protected with OSCORE to the RS. This message may contain the ID Context value, i.e. N1 concatenated with N2 concatenated with the Group Identifier of the OSCORE group. If the request is correctly verified, then the RS stores the pairwise OSCORE Security Context, and uses it to protect the possible response, as well as further communications with the Client, until the Access Token expires. This pairwise OSCORE Security Context is discarded if the same Access Token is re-used to successfully derive a new pairwise OSCORE Security Context. Once the Client has received a valid secure response, it does not continue to include the ID Context value in following requests.

As discussed in Section 2 of [I-D.ietf-ace-oscore-profile], the use of random nonces N1 and N2 during the exchange between the Client and the RS prevents the reuse of AEAD nonces and keys with different messages, in case of re-derivation of the pairwise OSCORE Security Context both for Clients and Resource Servers from an old non-expired Access Token, e.g. in case of reboot of either the Client or the RS.

Furthermore, the Client can send a request protected with Group OSCORE [I-D.ietf-core-oscore-groupcomm]. This can be a unicast request addressed to the RS, or a multicast request 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.

Section 3.2 of [I-D.ietf-core-object-security] defines how to derive a pairwise OSCORE Security Context based on a shared Master Secret and a set of other parameters, established between the OSCORE client and server.

The Client receives these pieces of information from the AS during the exchange described in this section. In particular, the proof-of-possession key (pop-key) provisioned by the AS MUST be used to build the Master Secret in OSCORE (see Section 4.3 and Section 4.4).
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.

The POST request is formatted as the analogous Client-to-AS request in the OSCORE profile of ACE (see Section 3.1 of [I-D.ietf-ace-oscore-profile]), with the following additional parameters included in the payload.

- `context_id`, defined in Section 3.1.1 of this specification. This parameter includes the Group ID of the OSCORE group that the Client has previously joined and wants to use to communicate with the RS. If the Group ID is structured according to the {Prefix + Epoch} scheme defined in Appendix C of [I-D.ietf-core-oscore-groupcomm], the Client MUST indicate the zeroed-epoch Group ID, i.e. with the Epoch part set to zero.

- `salt`, defined in Section 3.1.2 of this specification. This parameter includes the Sender ID that the Client has received in the OSCORE group, whose identifier is indicated in the `context_id` parameter above, upon previously joining it. That is, its value is the Sender ID that the Client uses to communicate in the OSCORE group, whereas it does not relate to the Sender ID to be assigned for use in the pairwise OSCORE Security Context with the RS.

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

```json
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
{
    "audience" : "tempSensor4711",
    "scope" : "read",
    "salt" : h'00',
    "context_id" : h'abcd0000'
}
```

Figure 2: Example C-to-AS POST /token request for an Access Token bound to a symmetric key.
Later on, the Client may want to update its current access rights, without changing the existing pairwise OSCORE Security Context with the RS. In this case, like in the OSCORE profile of ACE (see Section 3.1 of [I-D.ietf-ace-oscore-profile]), the Client MUST include in its POST request to the /token endpoint a req_cnf object, where the ‘kid’ field carries the Client’s identifier, that was assigned by the AS as per Section 3.2. That is, the Client’s identifier is the value of the ‘clientId’ parameter in the OSCORE Security Context object of the ‘cnf’ parameter, in the AS-to-C Access Token response providing the original Access Token (see Section 3.2).

The AS can use this identifier to determine the shared secret for preparing the proof-of-possession Access Token. Therefore, the received value MUST identify a symmetric key that was previously generated by the AS, as a shared secret for communications between the Client and the RS. In particular, the AS MUST verify that the received value identifies a proof-of-possession key and Access 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:
{
  "audience" : "tempSensor4711",
  "scope" : "read",
  "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.

### 3.1.1. ‘context_id’ Parameter

The ‘context_id’ parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.6.1. of [I-D.ietf-ace-oauth-authz]. 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’ Parameter

The ‘salt’ parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.6.1. of [I-D.ietf-ace-oauth-authz]. This parameter provides a value that the Client wishes to use as salt with the RS, for deriving cryptographic key material. Its exact content is profile specific.

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 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 and Group OSCORE is REQUIRED for a specific Access Token by including the ‘profile’ parameter with the value "coap_group_oscore" in the Access Token response. This means that the Client MUST use OSCORE and/or Group OSCORE towards all the Resource Servers for which this Access Token is valid.

In particular, the Client MUST follow Section 4.3 to derive the pairwise OSCORE Security Context to use for communications with the RS. Additionally, the Client has already established the related Group OSCORE Security Context to communicate with members of the OSCORE group, upon previously joining that group.

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.

The Access Token response to the Client is analogous to the one in the OSCORE profile of ACE, as described in Section 3.2 of [I-D.ietf-ace-oscore-profile]. In particular, the AS provides also the following additional information in the OSCORE_Security_Context object, which is defined in Section 3.2.1 of [I-D.ietf-ace-oscore-profile] and included in the ‘cnf’ parameter (see Section 3.2 of [I-D.ietf-ace-oauth-params]) of the Access Token response.

- The ‘salt’ field in the OSCORE_Security_Context object MUST be present. The field MUST contain the value of the ‘salt’ parameter from the Access Token request received from the Client.
The ‘contextId’ field in the OSCORE_Security_Context object MUST be present. The field MUST contain the value of the ‘context_id’ parameter from the Access Token request received from the Client.

The same parameters MUST be included as metadata of the issued Access Token. This profile RECOMMENDS the use of CBOR web tokens (CWT) as specified in [RFC8392]. If the Access Token is a CWT, the same OSCORE_Security_Context structure considered above MUST be placed in the ‘cnf’ claim of the Access Token.

As discussed in Section 3.2 of [I-D.ietf-ace-oscore-profile], collisions of client identifiers may appear in the RS, in case a resource is associated to multiple ASs. In such a case, the RS needs to have a mechanism in place to disambiguate identifiers or mitigate the effect of their collision.

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_group_oscore",
  "expires_in" : 3600,
  "cnf" : {
    "OSCORE_Security_Context" : {
      "alg" : "AES-CCM-16-64-128",
      "clientId" : b64'qA',
      "serverId" : b64'Qg',
      "ms" : h'f9af838368e353e88888e1426bd94e6f',
      "salt" : h'00',
      "context_id" : h'abcd0000'
    }
  }
}
```

Figure 4: Example AS-to-C Access Token response with the Group 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",
  "cnt" : {
    "OSCORE_Security_Context" : {
      "alg" : "AES-CCM-16-64-128",
      "clientId" : 'client',
      "serverId" : 'server',
      "ms" : h'f9af838368e353e78888e1426bd94e6f',
      "salt" : h'00',
      "context_id" : h'abcd0000'
    }
  }
}

Figure 5: Example CWT with OSCORE parameters (CBOR diagnostic notation).

The same CWT 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: 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.
If the Client has requested an update to its access rights with reference to the same pairwise OSCORE Security Context, which is valid and authorized, the AS MUST omit the 'cnf' parameter in the Access Token response, and MUST include the Client identifier in the 'kid' field of the 'cnf' parameter of the Access Token. The Client identifier needs to be provisioned, in order for the RS to identify the previously generated pairwise OSCORE Security Context.

Figure 7 shows an example of such an AS response, in CBOR diagnostic notation without the tag and value abbreviations.
Payload:
{
  "access_token" : h'a5037674656d7053656e73 ...'
  (remainder of access token omitted for brevity),
  "profile" : "coap_group_oscore",
  "expires_in" : 3600
}

Figure 7: Example AS-to-C Access Token response with the Group 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'qA'
  }
}

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

4. Client-RS Communication

This section details the POST request and response to the /authz-info endpoint between the Client and the RS. With respect to the exchanged messages and their content, the Client and the RS perform as defined in Section 4 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

That is, the Client generates a nonce N1 and posts it to the RS, together with the Access Token that includes the material provisioned by the AS. Then, the RS generates a nonce N2, and derives a pairwise OSCORE Security Context as described Section 3.2 of [I-D.ietf-core-object-security]. In particular, it uses the two nonces established with the Client and two shared secrets, together with additional pieces of information specified in the Access Token.

The proof-of-possession required to bind the Access Token to the Client is implicitly performed by generating the pairwise OSCORE
Security Context using the pop-key as part of the OSCORE Master Secret, for both the Client and the RS. In addition, the derivation of the pairwise OSCORE Security Context takes as input also information related to the OSCORE group, i.e. the Master Secret and Group Identifier of the group, as well as the Sender ID of the Client in the group. Hence, the derived pairwise OSCORE Security Context is also securely bound to the Group OSCORE Security Context of the OSCORE Group.

Therefore, an attacker using a stolen Access Token cannot generate a valid pairwise OSCORE Security Context and thus cannot prove possession of the pop-key. Also, if a Client legitimately owns an Access Token but has not joined the OSCORE group, that Client cannot generate a valid pairwise OSCORE Security Context either, since it lacks the Master Secret used in the OSCORE group.

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

The Client MUST generate a nonce N1 and post it to the /authz-info endpoint of the RS together with the Access Token, as defined in Section 4.1 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

The same recommendations, considerations and behaviors defined in Section 4.1 of [I-D.ietf-ace-oscore-profile] hold for this specification.

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

The RS MUST verify the validity of the Access Token as defined in Section 4.2 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile], with the following additions.

- The RS checks that the OSCORE_Security_Context object in the ‘cnf’ claim of the Access Token includes the ‘salt’ parameter.

- The RS checks that the OSCORE_Security_Context object in the ‘cnf’ claim of the Access Token includes the ‘context_id’ parameter. Also, the RS checks that the value of the ‘context_id’ parameter coincides with the one of the (zeroed-epoch) group identifier of the OSCORE group associated to the resources targeted by the scope in 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 MUST generate a nonce N2 and include it in the 2.01 (Created) response to the Client, as defined in Section 4.2 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

Further recommendations, considerations and behaviors defined in Section 4.2 of [I-D.ietf-ace-oscore-profile] hold for this specification.

4.3. OSCORE Setup - Client Side

Once having received the 2.01 (Created) response from the RS, following the POST request to the authz-info endpoint, the Client MUST extract the nonce N2 from the ‘nonce’ parameter and the client identifier from the ‘clientId’ (if present) in the CBOR map in the payload of the response.

Note that, if present in the 2.01 (Created) response, the ‘clientId’ parameter supersedes the analogous parameter possibly provided by the AS to C in Section 3.2. Also, note that this identifier is used by C as Sender ID in the pairwise OSCORE Security Context to be established with the RS, and is different as well as unrelated to the Sender ID of C in the OSCORE group.

Then, the Client performs the following actions, in order to set up and fully derive the pairwise OSCORE Security Context for communicating with the RS.

- The Client MUST set the ID Context of the pairwise OSCORE Security Context as the concatenation of: the nonce N1; the nonce N2; and the Group Identifier GID of the OSCORE group. The concatenation occurs in this order: ID Context = N1 | N2 | GID, where | denotes byte string concatenation.

- The Client MUST set the Master Salt of the pairwise OSCORE Security Context as the concatenation of MSalt and GMSalt, where: i) MSalt is the Sender ID that the Client has in the OSCORE group; while ii) GMSalt is the (optional) Master Salt in the Group OSCORE Security Context, which is known to the Client as a member of the OSCORE group. The concatenation occurs in this order: Master Salt = MSalt | GMSalt, where | denotes byte string concatenation.

- The Client MUST set the Master Secret of the pairwise OSCORE Security Context to the concatenation of MSec and GMSec, where: i) MSec is the value of the ‘ms’ parameter in the OSCORE_Security_Context object of the ‘cnf’ parameter, received from the AS in Section 3.2; while ii) GMSec is the Master Secret.
of the Group OSCORE Security Context, which is known to the Client as a member of the OSCORE group.

- The Client MUST set the Recipient ID as indicated in the corresponding parameter received from the AS in Section 3.2, i.e. to the value of the ‘serverId’ parameter in the OSCORE_Security_Context object of the ‘cnf’ parameter.

- The Client MUST set the AEAD Algorithm, HKDF, and Replay Window as indicated in the corresponding parameters received from the AS in Section 3.2, if present in the OSCORE_Security_Context object of the ‘cnf’ parameter. 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 as indicated in the ‘clientId’ parameter from the 2.01 (Created) response, if present. Otherwise, the Client MUST set the Sender ID as indicated in the response from the AS in Section 3.2, i.e. to the value of the ‘clientId’ parameter in the OSCORE_Security_Context object of the ‘cnf’ parameter.

Finally, the client MUST derive the complete pairwise OSCORE Security Context following Section 3.2.1 of [I-D.ietf-core-object-security].

From then on, when communicating with the RS to access the resources as specified by the authorization information, the Client MUST use the newly established pairwise OSCORE Security Context or the Group OSCORE Security Context of the OSCORE Group where both the Client and the RS are members.

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

Then, the Client uses this pairwise OSCORE Security Context to send requests to RS protected with OSCORE. In the first request sent to the RS, the Client MAY include the kid context if the application needs to, with value the ID Context, i.e. N1 concatenated with N2 concatenated with GID. Besides, the Client uses the Group OSCORE Security Context for protecting unicast requests to the RS, or multicast requests to the OSCORE group including also the RS.
4.4. OSCORE Setup - Resource Server Side

After validation of the Access Token as defined in Section 4.2 and after sending the 2.01 (Created) response, the RS performs the following actions, in order to set up and fully derive the pairwise OSCORE Security Context created to communicate with the Client.

- The RS MUST set the ID Context of the pairwise OSCORE Security Context as the concatenation of: the nonce N1; the nonce N2; and the Group Identifier GID of the OSCORE group. The concatenation occurs in this order: ID Context = N1 | N2 | GID, where | denotes byte string concatenation.

- The RS MUST set the Master Salt of the pairwise OSCORE Security Context as the concatenation of MSalt and GMSalt, where: i) MSalt is the Sender ID that the Client has in the OSCORE group, as specified in the ‘salt’ parameter in the OSCORE_Security_Context object of the ‘cnf’ claim, included in the Access Token received from the Client (see Section 4.1); while ii) GMSalt is the (optional) Master Salt in the Group OSCORE Security Context, which is known to the RS as a member of the OSCORE group. The concatenation occurs in this order: Master Salt = MSalt | GMSalt, where | denotes byte string concatenation.

- The RS MUST set the Master Secret of the pairwise OSCORE Security Context to the concatenation of MSec and GMSec, where: i) MSec is the value of the ‘ms’ parameter in the OSCORE_Security_Context object of the ‘cnf’ claim, included in the Access Token received from the Client (see Section 4.1); while ii) GMSec is the Master Secret of the Group OSCORE Security Context, which is known to the RS as a member of the OSCORE group.

- The RS MUST set the Sender ID of the pairwise OSCORE Security Context from the corresponding parameter received from the Client in the Access Token (see Section 4.1), i.e. to the value of the ‘serverId’ parameter in the OSCORE_Security_Context object of the ‘cnf’ claim.

- The RS MUST set the Recipient ID of the pairwise OSCORE Security Context from either what it indicated in the 2.01 (Created) response if included (see Section 4.2 of [I-D.ietf-ace-oscore-profile]), or from the corresponding parameter received from the Client in the Access Token (see Section 4.1), i.e. to the value of the ‘clientId’ parameter in the OSCORE_Security_Context object of the ‘cnf’ claim.

- The RS MUST set the AEAD Algorithm, HKDF, and Replay Window from the corresponding parameters received from the Client in the
Access Token (see Section 4.1), if present in the OSCORE_Security_Context object of the ‘cnf’ claim. In case these parameters are omitted, the default values are used as described in Section 3.2 of [I-D.ietf-core-object-security].

Finally, the RS MUST derive the complete pairwise OSCORE Security Context following Section 3.2.1 of [I-D.ietf-core-object-security].

Once having completed the derivation above, the RS MUST associate the authorization information from the Access Token with the just established pairwise OSCORE Security Context.

Furthermore, the RS MUST associate the authorization information from the Access Token with the pair (GID, MSalt), where GID is the Group Identifier of the OSCORE Group and MSalt is the Sender ID that the Client has in that OSCORE group. The RS MUST keep this association up-to-date over time.

Then, the RS uses this pairwise OSCORE Security Context to verify requests from and send responses to the Client protected with OSCORE, when this Security Context is used. If OSCORE verification fails, error responses are used, as specified in Section 8 of [I-D.ietf-core-object-security]. Besides, the RS uses the Group OSCORE Security Context to verify (multicast) requests from and send responses to the Client protected with Group OSCORE. If Group OSCORE verification fails, error responses are used, as specified in Section 6 of [I-D.ietf-core-oscore-groupcomm]. Additionally, for every incoming request, if OSCORE or Group OSCORE verification succeeds, the verification of access rights is performed as described in Section 4.5.

After the expiration of the Access Token related to a pairwise OSCORE Security Context and to a Group OSCORE Security Context, the RS MUST NOT use the pairwise OSCORE Security Context and MUST respond with an unprotected 4.01 (Unauthorized) error message. Also, 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.

4.5. 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, the RS processes it according to [I-D.ietf-core-object-security]. If an RS receives a Group OSCORE-protected request from a Client, the RS processes it according to [I-D.ietf-core-oscore-groupcomm].
If the OSCORE or Group OSCORE verification succeeds, and the target resource requires authorization, the RS retrieves the authorization information from the Access Token associated to the pairwise OSCORE Security Context and to the Group OSCORE Security Context. Then, the RS MUST verify that the authorization information covers the resource and the action requested by the Client.

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

5. Secure Communication with the 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 communicate 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 the scope of 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

The Client and the RS MUST follow what is defined in Section 6 of [I-D.ietf-ace-oscore-profile] about discarding the pairwise OSCORE 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 2.1 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, in order to establish a new pairwise OSCORE Security Context and bind it to the Group OSCORE Security Context obtained upon re-joining the group.

7. CBOR Mappings

The new parameters and claims defined in this document MUST be mapped to CBOR types as specified in Figure 9, using the given integer abbreviation for the map key.

```
<table>
<thead>
<tr>
<th>Parameter name</th>
<th>CBOR Key</th>
<th>Value Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>context_id</td>
<td>TBD1</td>
<td>bstr</td>
</tr>
<tr>
<td>salt</td>
<td>TBD2</td>
<td>bstr</td>
</tr>
</tbody>
</table>
```

Figure 9: CBOR mappings for new parameters.

8. 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 ACE framework also apply to this profile.

This specification inherits the security considerations from the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile]. Also, the general security considerations about OSCORE [I-D.ietf-core-object-security] hold for this document, as to the specific use of OSCORE according to this profile.

Furthermore, the general security considerations about Group OSCORE [I-D.ietf-core-oscore-groupcomm] hold for this document, 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, for group requests and responses. To ensure source authentication of messages, Group OSCORE uses digital countersignatures that group members embed in their own transmitted messages.
9. 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 ACE framework also apply to this profile.

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

This profile also inherits the privacy considerations from the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

10. IANA Considerations

This document has the following actions for IANA.

10.1. ACE Profile Registry

IANA is asked to enter the following value into the "ACE Profile Registry" defined in Section 8.7 of [I-D.ietf-ace-oauth-authz].

- Profile name: coap_group_oscore
- Profile Description: Profile to secure communications between constrained nodes using the Authentication and Authorization for Constrained Environments framework by: i) establishing a Pairwise OSCORE Security Context and enabling OSCORE communication between two members of an OSCORE group; ii) 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.
- Profile ID: TBD (value between 1 and 255)
- Change Controller: IESG
- Specification Document(s): [[this document]]

10.2. OAuth Parameters Registry

IANA is asked to enter the following values into the "OAuth Parameters Registry" defined in Section 11.2 of [RFC6749].

- Name: "context_id"
- Parameter Usage Location: token request
10.3. OAuth Parameters CBOR Mappings Registry

IANA is asked to enter the following values into the "OAuth Parameters CBOR Mappings Registry" defined in Section 8.9 of [I-D.ietf-ace-oauth-authz].

- **Name**: "context_id"
  - **CBOR Key**: TBD1
  - **Change Controller**: IESG
  - **Reference**: Section 3.1.1 of [[this document]]

- **Name**: "salt"
  - **CBOR Key**: TBD2
  - **Change Controller**: IESG
  - **Reference**: Section 3.1.2 of [[this document]]

11. References

11.1. Normative References

[I-D.dijk-core-groupcomm-bis]

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

[I-D.ietf-ace-oauth-params]

[I-D.ietf-ace-oscore-profile]

[I-D.ietf-ace-object-security]

[I-D.ietf-ace-oscore-groupcomm]


11.2. Informative References

[I-D.ietf-ace-dtls-authorize]

[I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", draft-ietf-ace-key-groupcomm-oscore-02 (work in progress), July 2019.

[I-D.ietf-ace-mqtt-tls-profile]

[I-D.ietf-tls-dtls13]

[I-D.tiloca-core-oscore-discovery]

[RFC5246]

[RFC6347]

[RFC7390]
Appendix A. Profile Requirements

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

- (Optional) discovery process of how the Client finds the right AS for an RS it wants to send a request to: Not specified.

- Communication protocol the Client and the RS must use: CoAP.

- Security protocol(s) the Client and RS must use: OSCORE, i.e. establishment of a pairwise OSCORE Security Context and exchange of secure messages; and/or Group OSCORE, i.e. exchange of secure messages by using a pre-established Group OSCORE Security Context.

- How the Client and the RS mutually authenticate: Implicitly by possession of a common OSCORE Security Context (when using OSCORE); and/or explicitly, by possession of a common Group OSCORE Security Context and usage of digital countersignatures (when using Group OSCORE).

- Content-format of the protocol messages: "application/ace+cbor".

- Proof-of-Possession protocol(s) and how to select one; which key types (e.g. symmetric/asymmetric) supported: OSCORE algorithms; pre-established symmetric keys.

- Profile identifier: coap_group_oscore

- (Optional) how the RS talks to the AS for introspection: HTTP/CoAP (+ TLS/DTLS/OSCORE).

- How the client talks to the AS for requesting a token: HTTP/CoAP (+ TLS/DTLS/OSCORE).

- How/if the authz-info endpoint is protected: Security protocol above.
(Optional) other methods of token transport than the authz-info endpoint: no.

Acknowledgments

The authors sincerely thank Goeran Selander for his comments and feedback.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC.

Authors’ Addresses

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

Email: marco.tiloca@ri.se

Rikard Hoeglund
RISE AB
Isafjordsgatan 22
Kista SE-16440 Stockholm
Sweden

Email: rikard.hoglund@ri.se

Ludwig Seitz
RISE AB
Scheelevagen 17
Lund SE-22370 Lund
Sweden

Email: ludwig.seitz@ri.se

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

Email: francesca.palombini@ericsson.com