

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

F. Palombini
Ericsson AB
M. Tiloca
RISE AB
November 04, 2019

Key Provisioning for Group Communication using ACE draft-ietf-ace-key-groupcomm-03

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 May 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	2
1.1.	Terminology	3
2.	Overview	4
3.	Authorization to Join a Group	6
3.1.	Authorization Request	6
3.2.	Authorization Response	7
3.3.	Token Post	8
4.	Keying Material Provisioning and Group Membership Management	11
4.1.	Interface at KDC	12
4.2.	Joining Exchange	21
4.3.	Retrieval of Updated Keying Material	22
4.4.	Retrieval of New Keying Material	24
4.5.	Retrieval of Public Keys for Group Members	24
4.6.	Retrieval of Group Policies	25
4.7.	Retrieval of Keying Material Version	25
4.8.	Group Leaving Request	26
5.	Removal of a Node from the Group	26
6.	ACE Groupcomm Parameters	27
7.	Security Considerations	28
7.1.	Update of Keying Material	29
7.2.	Block-Wise Considerations	29
8.	IANA Considerations	30
8.1.	ACE Authorization Server Request Creation Hints Registry	30
8.2.	ACE Groupcomm Parameters Registry	30
8.3.	ACE Groupcomm Key Registry	31
8.4.	ACE Groupcomm Profile Registry	32
8.5.	ACE Groupcomm Policy Registry	32
8.6.	Sequence Number Synchronization Method Registry	33
8.7.	Expert Review Instructions	34
9.	References	34
9.1.	Normative References	34
9.2.	Informative References	35
Appendix A.	Requirements on Application Profiles	37
Appendix B.	Document Updates	39
B.1.	Version -02 to -03	39
B.2.	Version -01 to -02	39
B.3.	Version -00 to -01	40
	Acknowledgments	41
	Authors' Addresses	41

[1.](#) Introduction

This document expands the ACE framework [[I-D.ietf-ace-oauth-authz](#)] to define the message exchanges 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, by defining a number of details such as the exact group communication protocol and security protocols used. The specific list of details a profile needs to define is in [Appendix A](#).

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. 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", "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 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:

- o Transport profile, to indicate a profile of ACE as per Section 5.6.4.3 of [[I-D.ietf-ace-oauth-authz](#)]. 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, etc. Transport profiles of ACE include, for instance, [[I-D.ietf-ace-oscore-profile](#)], [[I-D.ietf-ace-dtls-authorize](#)] and [[I-D.ietf-ace-mqtt-tls-profile](#)].
- o Application profile, that defines how applications enforce and use supporting security services they require. These services may include, for instance, provisioning, revocation and (re-)distribution of keying material. An application profile may define specific procedures and message formats.

2. Overview

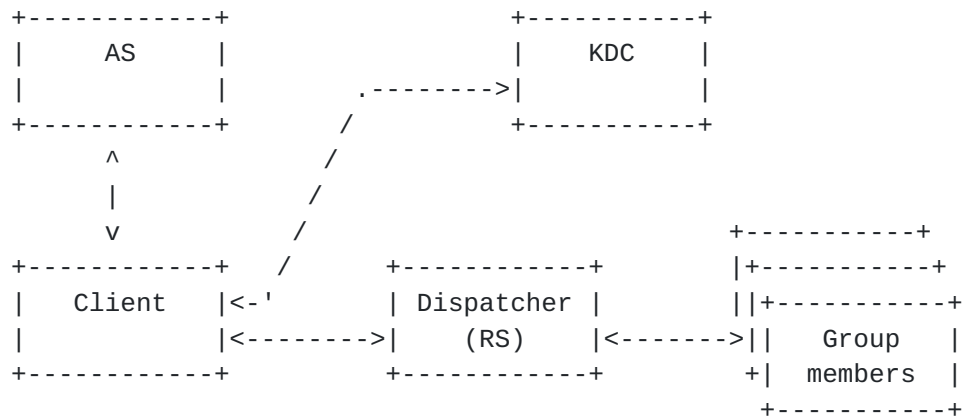


Figure 1: Key Distribution Participants

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

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

- o 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](#)).
- o A node to leave the group of for the KDC to remove a current member of the group ([Section 5](#)).
- o Retrieving keying material as a current group member ([Section 4.3](#) and [Section 4.4](#)).
- o Renewing and re-distributing the group keying material (rekeying) upon a membership change in the group ([Section 4.8](#) and [Section 5](#)).

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

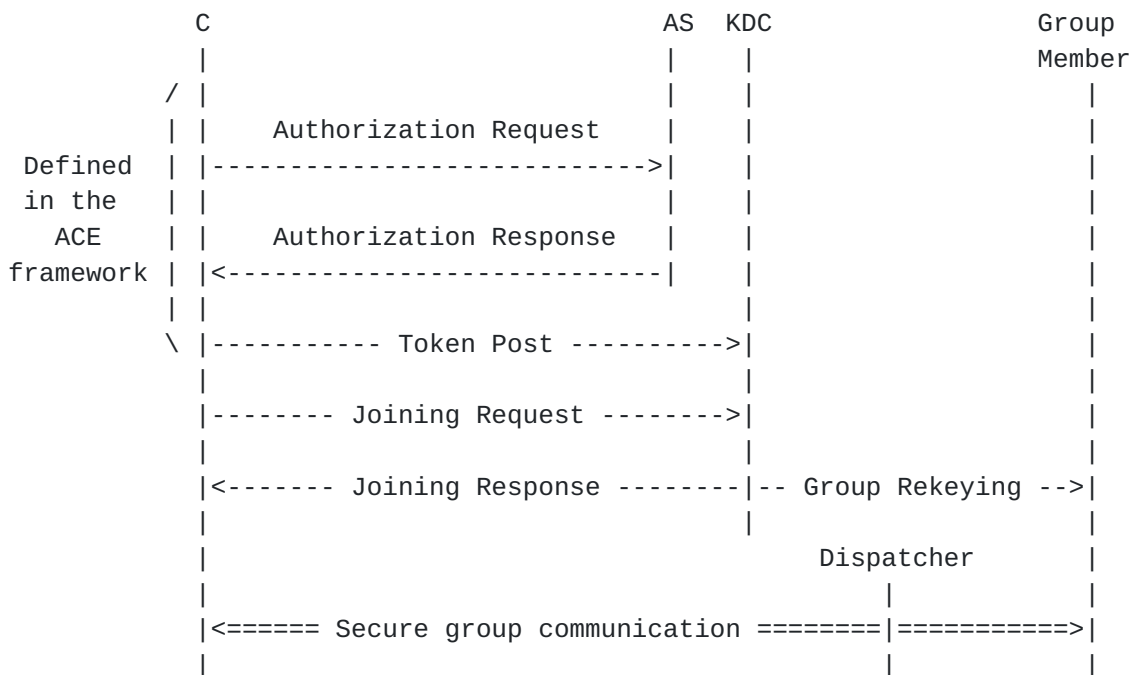


Figure 2: Message Flow Upon New Node's Joining

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 Joining Request and Joining 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. This 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, as defined by 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). The transport profile of ACE also defines a number of details such as the communication and security protocols used with the KDC (see [Appendix C](#) of [[I-D.ietf-ace-oauth-authz](#)]).

Figure 3 gives an overview of the exchange described above.



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

- o '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 (REQ1) and of the role identifiers (REQ2) is application specific, and part of the requirements for the application profile.

- o 'audience', with an identifier of a KDC.
- o '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.
- o Other additional parameters as defined in [\[I-D.ietf-ace-oauth-authz\]](#), if necessary.

As in [\[I-D.ietf-ace-oauth-authz\]](#), these parameters are included in the payload, which is formatted as a CBOR map. The Content-Format "application/ace+cbor" defined in Section 8.14 of [\[I-D.ietf-ace-oauth-authz\]](#) is used.

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:

- o 'access_token', containing the proof-of-possession access token.
- o '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 the same as in [Section 3.1](#).

- 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.

As in [\[I-D.ietf-ace-oauth-authz\]](#), these parameters are included in the payload, which is formatted as a CBOR map. The Content-Format "application/ace+cbor" is used.

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 want 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". The payload of the message MUST be formatted as a CBOR map, including the access token and the following parameters:

- o '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.
- o '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.

```
sign_info_req = nil
```

```
pub_key_enc_req = nil
```

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 the parameter 'rsnonce' defined in [Section 3.3.3](#), specifying a dedicated nonce N_S generated by the KDC. The Client may use this nonce for proving 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.

- o 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 (REQ3).
- o 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 (REQ4). If no parameters of the signature algorithm are specified, 'sign_parameters' MUST be encoded as the CBOR simple value Null.
- o 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 (REQ5). If no parameters of the key used with the signature algorithm are specified, 'sign_key_parameters' MUST be encoded as 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. Its acceptable values are taken from the "CWT Confirmation Method" Registry defined in [[I-D.ietf-ace-cwt-proof-of-possession](#)]. It is REQUIRED of the application profiles to define specific values to use for this parameter (REQ6).

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

```
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.

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.

In this specification and in application profiles building on it, this parameter is used to ask and retrieve from the KDC information about the signature algorithm and related parameters used in the group.

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.

In this specification and in application profiles building on it, this parameter is used to ask and retrieve from the KDC information about the encoding of public keys used in the group.

3.3.3. 'rsnonce' Parameter

The 'rsnonce' 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 a nonce generated by the RS and provided to the Client. Its exact content is application specific.

In this specification and in application profiles building on it, this parameter is used to provide a nonce that the Client may use to prove possession of its own private key in the Joining Request ((see the 'client_cred_verify' parameter in [Section 4](#))).

4. Keying Material Provisioning and Group Membership Management

This section defines the interface available at the KDC. Moreover, this section specifies how the clients can use this interface to join a group, leave a group, retrieve new keying material or policies.

During the first exchange with the KDC ("Joining"), the Client sends a request to the KDC, specifying the group it wishes to join (see [Section 4.2](#)). 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 other members of the group. Whenever used, the Content-Format in messages containing a payload is set to application/cbor.

TODO: Do we need to define a new Content-Format cbor+ace-groupcomm?

When the Client is already a group member, the Client can use the interface at the KDC to perform the following actions:

- o The Client can (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 described in [Section 4.3](#).
- o The Client can retrieve a new individual key, or new input material to derive it. This is described in [Section 4.4](#).
- o The Client can (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 described in [Section 4.5](#).
- o The Client can (re-)get the policies currently enforced in the group. This is described in [Section 4.6](#).
- o The Client can (re-)get the version number of the keying material currently used in the group. This is described in [Section 4.7](#).
- o The Client can request to leave the group. This is further discussed in [Section 4.8](#).

Upon receiving a request from a Client, the KDC MUST check that it is storing a valid access token from that Client for the group identifier associated to the endpoint. If that is not the case, i.e. the KDC does not store a valid access token or this is not valid for that Client for the group identifier at hand, the KDC MUST respond to the Client with a 4.01 (Unauthorized) error message.

[4.1](#). Interface at KDC

The KDC is configured with the following resources:

- o /ace-group : this resource is fixed and indicates that this specification is used. Other applications that run on a KDC implementing this specification MUST NOT use this same resource.
- o /ace-group/gid : one sub-resource to /ace-group is implemented for each group the KDC manages. These resources are identified by the group identifiers of the groups the KDC manages (in this example,

the group identifier has value "gid"). These resources support GET and POST method.

- o /ace-group/gid/pub-key : this sub-resource is fixed and supports GET and POST methods.
- o /ace-group/gid/policies: this sub-resource is fixed and supports the GET method.
- o /ace-group/gid/ctx-num: this sub-resource is fixed and supports the GET method.
- o /ace-group/gid/node: this sub-resource is fixed and supports GET and POST methods.

The details for the handlers of each resource are given in the following sections. These endpoints are used to perform the operations introduced in [Section 4](#). Note that the url-path given here are default names: implementations are not required to use these names, and can define their own instead.

[4.1.1](#). **ace-group**

No handlers are implemented for this resource.

[4.1.2](#). **ace-group/gid**

This resource implements GET and POST handlers.

[4.1.2.1](#). **POST Handler**

The POST handler adds the public key of the client to the list of the group members' public keys and returns the symmetric group keying material for the group identified by "gid".

The handler expects a request with payload formatted as a CBOR map which MAY contain the following fields, which, if included, MUST have the corresponding values:

- o '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_cred', 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 (OPT1).
- o 'cnonce', as defined in Section 5.1.2 of [\[I-D.ietf-ace-oauth-authz\]](#), and including a dedicated nonce N_C generated by the Client. This parameter MUST be present if the 'client_cred' parameter is present.
- o 'client_cred_verify', encoded as a CBOR byte string. This parameter MUST be present if the 'client_cred' parameter is present. This parameter contains a signature computed by the Client over N_S concatenated with N_C, where N_S is the nonce received from the KDC in the 'rsnonce' parameter of the 2.01 (Created) response to the token POST request (see [Section 3.3](#)), while N_C is the nonce generated by the Client and specified in the 'cnonce' parameter above. The Client computes the signature by using its own private key, whose corresponding public key is either directly specified in the 'client_cred' parameter or included in the certificate specified in the 'client_cred' parameter.
- o 'pub_keys_repos', can be present if a certificate is present in the 'client_cred' 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.

The handler verifies that the group identifier of the /ace-group/gid path 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 request is not formatted correctly (e.g. unknown fields present), the handler MUST respond with 4.00 (Bad Request) error message.

If verification succeeds, the handler adds the public key indicated in "client_cred" to the list of public keys stored for the group identified by "gid". The handler returns a 2.01 (Created) message containing the symmetric group keying material, the group policies

and all the public keys of the current members of the group, if the KDC manages those and the Client requested them. The payload of the response is formatted as a CBOR map which MAY contain the following fields, which, if included, MUST have the corresponding values:

- 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.
- o 'num', containing the version number of the keying material for the group communication, formatted as an integer. The initial version MUST be set to 0 at the KDC.

The exact format of the 'key' value MUST be defined in applications of this specification (REQ7), as well as accepted values of 'kty' by the application (REQ8). Additionally, documents specifying the key format MUST register it in the "ACE Groupcomm Key" registry defined in [Section 8.3](#), including its name, type and application profile to be used with.

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

Figure 4: Key Type Values

Optionally, the 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 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 (OPT2). The specific format of the identifiers of group members MUST be specified in the application profile (REQ8).
- o '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 5. Application profiles that build on this document MUST specify the exact content format of included map entries (REQ9).

Name	CBOR label	CBOR type	Description	Reference
Sequence Number Synchronization Method	TBD1	tstr/int	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	[[this document]]
Key Update Check Interval	TBD2	int	Polling interval in seconds, to check for new keying material at the KDC	[[this document]]

Figure 5: ACE Groupcomm Policies

- o '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 (OPT3).

Specific application profiles that build on this document MUST specify how exactly the keying material is used to protect the group communication (REQ10).

CBOR labels for these fields are defined in [Section 6](#).

[4.1.2.2](#). GET Handler

The GET handler returns the symmetric group keying material for the group identified by "gid".

The handler expects a GET request.

The handler verifies that the group identifier of the /ace-group/gid path 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 verification succeeds, the handler returns a 2.05 (Content) message containing the symmetric group keying material, the group policies and all the public keys of the current members of the group. The payload of the response is formatted as a CBOR map which MUST contain the parameters 'kty', 'key' and 'num' specified in [Section 4.1.2.1](#).

The payload MAY also include the parameters 'profile', 'exp' and 'mgt_key_material' parameters specified in [Section 4.1.2.1](#).

[4.1.3](#). ace-group/gid/pub-key

This resource implements GET and POST handlers.

[4.1.3.1](#). POST Handler

The POST handler receives identifiers of group members for the group identified by "gid" and returns the public keys of such group members.

The handler expects a request with payload formatted as a CBOR map. The payload of this request is a CBOR Map that MUST contain the following fields:

- o 'get_pub_keys', whose value is a non-empty CBOR array. Each element of the array is the identifier of a group member for the group identified by "gid". The specific format of public keys as well as identifiers of group members MUST be specified by the application profile (REQ11, REQ8).

The handler verifies that the group identifier of the /ace-group/gid path 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.

The handler verifies that the 'get_pub_keys' parameter is not an empty CBOR Array. If verification fails, the KDC MUST treat the request as malformed and respond with a 4.00 (Bad Request) error message.

If verification succeeds, the handler identifies the public keys of the current group members for which the identifier matches with one of those indicated in the request. Then, the handler returns a 2.05 (Content) message response with payload formatted as a CBOR map containing only the 'pub_keys' parameter from [Section 4.1.2.1](#), which encodes the list of public keys of those group members including the respective member identifiers. If the KDC does not store any public key associated with the specified member identifiers, the handler

returns a response with payload formatted as a CBOR byte string of zero length. The specific format of public keys as well as of identifiers of group members is specified by the application profile (REQ11, REQ8).

The handler MAY enforce one of the following policies, in order to handle possible identifiers that are included in the 'get_pub_keys' parameter of the request but are not associated to any current group member. Such a policy MUST be specified by the application profile (REQ12)

- o The KDC silently ignores those identifiers.
- o 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.

4.1.3.2. GET Handler

The handler expects a GET request.

The handler verifies that the group identifier of the /ace-group/gid path 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 verification succeeds, the handler returns a 2.05 (Content) message containing the public keys of all the current group members, for the group identified by "gid". The payload of the response is formatted as a CBOR map containing only the 'pub_keys' parameter from [Section 4.1.2.1](#), which encodes the list of public keys of all the group members including the respective member identifiers. If the KDC does not store any public key for the group, the handler returns a response with payload formatted as a CBOR byte string of zero length. The specific format of public keys as well as of identifiers of group members is specified by the application profile (REQ11, REQ8).

4.1.4. ace-group/gid/policies

This resource implements a GET handler.

4.1.4.1. GET Handler

The handler expects a GET request.

The handler verifies that the group identifier of the /ace-group/gid path 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 verification succeeds, the handler returns a 2.05 (Content) message containing the list of policies for the group identified by "gid". The payload of the response is formatted as a CBOR map including only the parameter 'group_policies' defined in [Section 4.1.2.1](#) and specifying the current policies in the group. If the KDC does not store any policy, the payload is formatted as a zero-length CBOR byte string.

The specific format and meaning of group policies MUST be specified in the application profile (REQ13).

[4.1.5.](#) **ace-group/gid/ctx-num**

This resource implements a GET handler.

[4.1.5.1.](#) **GET Handler**

The handler expects a GET request.

The handler verifies that the group identifier of the /ace-group/gid path 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 verification succeeds, the handler returns a 2.05 (Content) message containing an integer that represents the version number of the symmetric group keying material. This number is incremented on the KDC every time the KDC updates the symmetric group keying material. The payload of the response is formatted as a CBOR integer.

[4.1.6.](#) **ace-group/gid/node**

This resource implements GET and POST handlers.

[4.1.6.1.](#) **POST Handler**

The POST handler removes the node from the group, for the group identified by "gid".

The handler expects a request with payload formatted as a CBOR map. The payload of this request is a CBOR Map that MAY contain only the 'scope' field as specified in [Section 4.1.2.1](#).

The handler verifies that the group identifier of the `/ace-group/gid` path 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 request contained a 'scope' field, the handler MUST extract the roles for that client. If the value is such that the KDC cannot extract all the necessary information to understand and process it correctly (e.g. unrecognized roles), the KDC MUST respond with a 4.00 (Bad Request) error message.

If verification succeeds, the handler removes the client from the group identified by "gid", for specific roles if roles were specified in the 'scope' field, or for all roles. That includes removing the public key of the client if the KDC keep tracks of that. Then, the handler returns a 2.05 (Content) message with empty payload.

[4.1.6.2.](#) GET Handler

The handler expects a GET request.

The handler verifies that the group identifier of the `/ace-group/gid` path 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 verification succeeds, the handler returns a 2.05 (Content) message containing newly-generated individual keying material for the Client, or information enabling the Client to derive it. The payload of the response is formatted as a CBOR map. The specific format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label, MUST be specified in the application profile (REQ14) and registered in [Section 8.2](#).

[4.2.](#) Joining Exchange

Figure 6 gives an overview of the Joining exchange between Client and KDC, when the Client first joins a group.



Figure 6: Message Flow of First Exchange for Group Joining

If not previously established, the Client and the KDC MUST first establish a pairwise secure communication channel (REQ15). This can be achieved, for instance, by using a transport profile of ACE. The Joining exchange 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.

The secure communication protocol is REQUIRED to establish the secure channel by using the proof-of-possession key bound to the access token. As a result, the 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.

To join the group, the Client sends a CoAP POST request to the /ace-group/gid endpoint at the KDC, where gid is the group identifier of the group to join, formatted as specified in [Section 4.1.2.1](#). This group identifier is the same as the 'scope' value of the Authorization Request/Response, or it can be retrieved from it.

If the application requires backward security, the KDC MUST generate new group keying material and securely distribute it to all the current group members, upon a new node's joining the group. To this end, the KDC uses the message format of the Joining Response (see [Section 4.1.2.1](#)). Application profiles may define alternative ways of retrieving the keying material, such as sending separate requests to different resources at the KDC ([Section 4.1.2.2](#), [Section 4.1.3.2](#), [Section 4.1.4.1](#)). After distributing the new group keying material, the KDC MUST increment the version number of the keying material.

[4.3. Retrieval of Updated Keying Material](#)

A node stops using the group keying material upon its expiration, according to what indicated by the KDC with the 'exp' parameter in a Joining response, or to a pre-configured value. Then, if it wants to continue participating in the group communication, the node has to request new updated keying material from the KDC.

The Client may need to request the latest group keying material also upon receiving messages from other group members without being able to retrieve the material 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. To this end, the Client sends a CoAP GET request to the /ace-group/gid endpoint at the KDC, formatted as specified in [Section 4.1.2.2](#).

Note that policies can be set up so that the Client sends a Key Re-Distribution 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-oscure-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 (OPT4).

The same Key Distribution Request could also be sent by the Client without being triggered by a failed decryption of a message, if the Client wants to be sure 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 already has in memory.

Figure 7 gives an overview of the exchange described above.

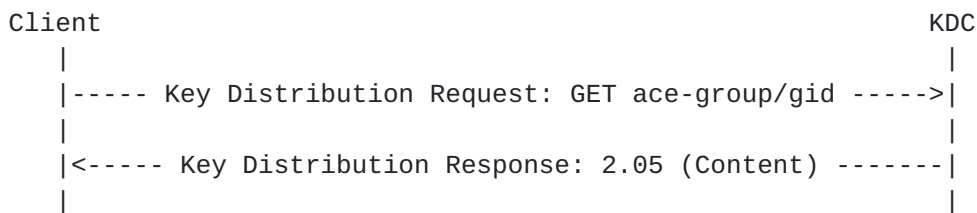


Figure 7: Message Flow of Key Distribution Request-Response

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

- o Can make the ace-group/gid resource Observable, and send notifications to Clients when the keying material is updated.
- o Can send the Key Distribution Response as one or multiple multicast requests to the members of the group, using secure rekeying schemes such as [[RFC2093](#)][RFC2094][[RFC2627](#)].
- o Can send unicast requests to each Client over a secure channel, with the same payload as the Key Distribution Response.
- o 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 distribution have different security properties and require different security associations.

4.4. Retrieval of New Keying Material

Beside possible expiration and depending on what part of the keying material is no longer eligible to be used, 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. To this end, the client performs a Key Renewal Request/Response exchange with the KDC, that is a CoAP GET request to the /ace-group/gid/node endpoint at the KDC, where gid is the group identifier, and formatted as defined in [Section 4.1.6.2](#).

Figure 8 gives an overview of the exchange described above.



Figure 8: Message Flow of Key Renewal Request-Response

Note the difference between the Key Distribution Request and the Key Renewal Request: while the first one only triggers distribution (the renewal might have happened independently, e.g. because of expiration), the second one triggers the KDC to produce new individual keying material for the requesting node.

4.5. 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, by sending a CoAP GET or POST request to the /ace-group/gid/pub-key endpoint at the KDC, where gid is the group identifier, and formatted as defined in [Section 4.1.3.2](#) and [Section 4.1.3.1](#).

Figure 9 and Figure 10 give an overview of the exchanges described above.

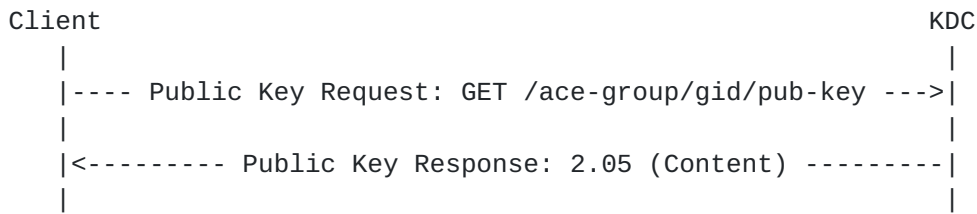


Figure 9: Message Flow of Public Key Exchange to Request All Members Public Keys

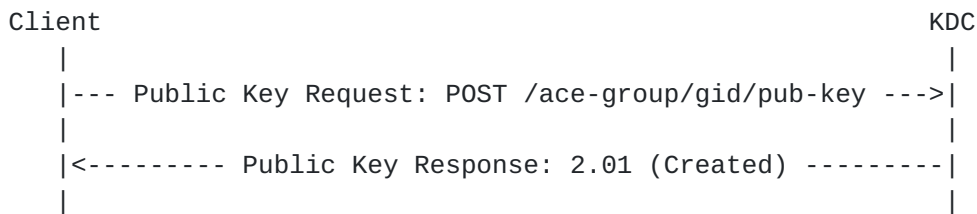


Figure 10: Message Flow of Public Key Exchange to Request Specific Members Public Keys

4.6. Retrieval of Group Policies

A node in the group can contact the KDC to retrieve the current group policies, by sending a CoAP GET request to the /ace-group/gid/policies endpoint at the KDC, where gid is the group identifier, and formatted as defined in [Section 4.1.4.1](#)

Figure 11 gives an overview of the exchange described above.

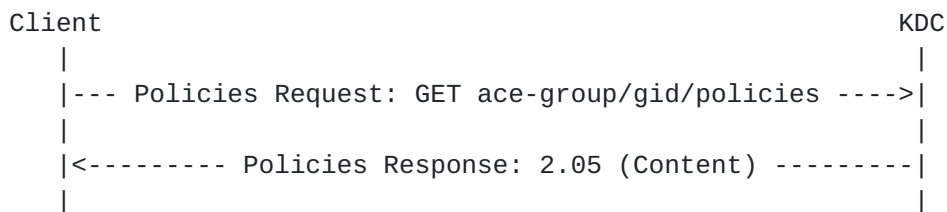


Figure 11: Message Flow of Policies Request-Response

4.7. Retrieval of Keying Material Version

A node in the group can contact the KDC to request information about the version number of the symmetric group keying material, by sending a CoAP GET request to the /ace-group/gid/ctx-num endpoint at the KDC, where gid is the group identifier, formatted as defined in [Section 4.1.5.1](#). In particular, the version is incremented by the KDC every time the group keying material is renewed.

Figure 12 gives an overview of the exchange described above.

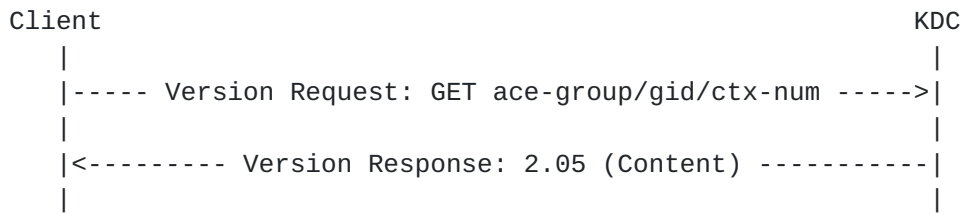


Figure 12: Message Flow of Version Request-Response

4.8. Group Leaving Request

A node can actively request to leave the group. In this case, the Client sends a CoAP POST request to the endpoint `/ace-group/gid/node` at the KDC, where `gid` is the group identifier, formatted as defined in [Section 4.1.6.1](#)

Alternatively, a node may be removed by the KDC, without having explicitly asked for it. This is further discussed in [Section 5](#).

5. Removal of a Node from the Group

This section describes the different scenarios according to which a node ends up being removed from the group.

If the application requires forward security, the KDC MUST generate new group keying material and securely distribute it to all the current group members but the leaving node, using the message format of the Key Distribution Response (see [Section 4.3](#)). Application profiles may define alternative message formats. Once distributed the new group keying material, the KDC MUST increment the version number of the keying material.

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 still has a 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 might decide to keep track of nodes with valid access tokens, before deleting all information about the leaving node.

A node may be evicted from the group in the following cases.

1. The node explicitly asks to leave the group, as defined in [Section 4.8](#).
2. The node has been found compromised or is suspected so.

3. The node's authorization to be a group member is expired. 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.

6. ACE Groupcomm Parameters

This specification defines a number of fields used during the second part of the message exchange, after the ACE Token POST exchange. The table below summarizes them, and specifies the CBOR key to use instead of the full descriptive name.

Name	CBOR Key	CBOR Type	Reference
scope	TBD	array	Section 4.1.2.1
get_pub_keys	TBD	array	Section 4.1.2.1
client_cred	TBD	byte string	Section 4.1.2.1
cnonce	TBD	byte string	Section 4.1.2.1
client_cred_verify	TBD	byte string	Section 4.1.2.1
pub_keys_repos	TBD	array	Section 4.1.2.1
kty	TBD	int / byte string	Section 4.1.2.1
key	TBD	see "ACE Groupcomm Key" Registry	Section 4.1.2.1

num	TBD	int	Section
			4.1.2.1
profile	TBD	int	Section
			4.1.2.1
exp	TBD	int / float	Section
			4.1.2.1
pub_keys	TBD	byte string	Section
			4.1.2.1
group_policies	TBD	map	Section
			4.1.2.1
mgt_key_material	TBD	byte string	Section
			4.1.2.1
get_pub_keys	TBD	array	Section
			4.1.3.1
+-----+-----+-----+-----+			

7. 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 \[RFC8613\]](#).

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 minum 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.

7.1. 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.

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.

7.2. 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 bandwidth resending the blocks again and again, which might be problematic.

8. IANA Considerations

This document has the following actions for IANA.

8.1. 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](#)].

- o Name: sign_info
- o CBOR Key: TBD (range -256 to 255)
- o Value Type: any
- o Reference: [[This specification]]
- o Name: pub_key_enc
- o CBOR Key: TBD (range -256 to 255)
- o Value Type: integer
- o Reference: [[This specification]]
- o Name: rsnonce
- o CBOR Key: TBD (range -256 to 255)
- o Value Type: byte string
- o Reference: [[This specification]]

8.2. 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 8.7](#).

The columns of this Registry are:

- o 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.
- o 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.
- o 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.
- o Reference: This contains a pointer to the public specification for the item.

This Registry has been initially populated by the values in [Section 6](#). The Reference column for all of these entries refers to sections of this document.

[8.3](#). 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 8.7](#).

The columns of this Registry are:

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

- o 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 4. The specification column for all of these entries will be this document.

8.4. 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 8.7](#). 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:

- o Name: The name of the application profile, to be used as value of the profile attribute.
- o Description: Text giving an overview of the application profile and the context it is developed for.
- o 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.
- o Reference: This contains a pointer to the public specification of the abbreviation for this application profile, if one exists.

8.5. 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 8.7](#). 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:

- o Name: The name of the group communication policy.
- o 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.
- o CBOR type: the CBOR type used to encode the value of this group communication policy.
- o Description: This field contains a brief description for this group communication policy.
- o 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 5.

8.6. 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 8.7](#). 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:

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

8.7. 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:

- o Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. The zones tagged as private use are intended for testing purposes and closed environments, code points in other ranges should not be assigned for testing.
- o Specifications are required for the standards track range of point assignment. Specifications should exist for specification required ranges, but early assignment before a specification is available is considered to be permissible. Specifications are needed for the first-come, first-serve range if they are expected to be used outside of closed environments in an interoperable way. When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for.
- o Experts should take into account the expected usage of fields when approving point assignment. The fact that there is a range for standards track documents does not mean that a standards track document cannot have points assigned outside of that range. The length of the encoded value should be weighed against how many code points of that length are left, the size of device it will be used on, and the number of code points left that encode to that size.

9. References

9.1. Normative References

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

Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", [draft-ietf-ace-cwt-proof-of-possession-11](#) (work in progress), October 2019.

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

Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", [draft-ietf-ace-oauth-authz-25](#) (work in progress), October 2019.

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

Seitz, L., "Additional OAuth Parameters for Authorization in Constrained Environments (ACE)", [draft-ietf-ace-oauth-params-05](#) (work in progress), March 2019.

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

Tiloca, M., Selander, G., Palombini, F., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", [draft-ietf-core-oscore-groupcomm-05](#) (work in progress), July 2019.

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

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

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

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

Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", [draft-dijk-core-groupcomm-bis-01](#) (work in progress), July 2019.

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

Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)", [draft-ietf-ace-dtls-authorize-08](#) (work in progress), April 2019.

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

Sengul, C., Kirby, A., and P. Fremantle, "MQTT-TLS profile of ACE", [draft-ietf-ace-mqtt-tls-profile-01](#) (work in progress), October 2019.

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

Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "OSCORE profile of the Authentication and Authorization for Constrained Environments Framework", [draft-ietf-ace-oscore-profile-08](#) (work in progress), July 2019.

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

Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", [draft-ietf-core-coap-pubsub-09](#) (work in progress), September 2019.

[RFC2093] Harney, H. and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification", [RFC 2093](#), DOI 10.17487/RFC2093, July 1997, <<https://www.rfc-editor.org/info/rfc2093>>.

[RFC2094] Harney, H. and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture", [RFC 2094](#), DOI 10.17487/RFC2094, July 1997, <<https://www.rfc-editor.org/info/rfc2094>>.

[RFC2627] Wallner, D., Harder, E., and R. Agee, "Key Management for Multicast: Issues and Architectures", [RFC 2627](#), DOI 10.17487/RFC2627, June 1999, <<https://www.rfc-editor.org/info/rfc2627>>.

[RFC7390] Rahman, A., Ed. and E. Dijk, Ed., "Group Communication for the Constrained Application Protocol (CoAP)", [RFC 7390](#), DOI 10.17487/RFC7390, October 2014, <<https://www.rfc-editor.org/info/rfc7390>>.

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

- [RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", [RFC 7959](#), DOI 10.17487/RFC7959, August 2016, <<https://www.rfc-editor.org/info/rfc7959>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, [RFC 8259](#), DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", [RFC 8613](#), DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/info/rfc8613>>.

Appendix A. Requirements on Application Profiles

TODD: fix req numbers in the text.

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

- o REQ1: Specify the encoding and value of the identifier of group or topic of 'scope' (see [Section 3.1](#)).
- o REQ2: Specify the encoding and value of roles of 'scope' (see [Section 3.1](#)).
- o REQ3: Optionally, specify the acceptable values for 'sign_alg' (see [Section 3.3](#)).
- o REQ4: Optionally, specify the acceptable values for 'sign_parameters' (see [Section 3.3](#)).
- o REQ5: Optionally, specify the acceptable values for 'sign_key_parameters' (see [Section 3.3](#)).
- o REQ6: Optionally, specify the acceptable values for 'pub_key_enc' (see [Section 3.3](#)).
- o REQ7: Specify the exact format of the 'key' value (see [Section 4.1.2.1](#)).
- o REQ8: Specify the acceptable values of 'kty' (see [Section 4.1.2.1](#)).
- o REQ9: Specify the format of the identifiers of group members (see [Section 4.1.2.1](#)).

- o REQ10: Optionally, specify the format and content of 'group_policies' entries (see [Section 4.1.2.1](#)).
- o REQ11: Specify the communication protocol the members of the group must use (e.g., multicast CoAP).
- o REQ12: 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.
- o REQ13: Specify and register the application profile identifier (see [Section 4.1.2.1](#)).
- o REQ14: Optionally, specify the encoding of public keys, of 'client_cred', and of 'pub_keys' if COSE_Keys are not used (see [Section 4.1.2.1](#)).
- o REQ15: Specify policies at the KDC to handle id that are not included in get_pub_keys (see [Section 4.1.3.1](#)).
- o REQ16: Specify the format and content of 'group_policies' (see [Section 4.1.2.1](#)).
- o REQ17: Specify the format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label (see [Section 4.1.6.2](#)).
- o REQ18: Specify how the communication is secured between Client and KDC. Optionally, specify transport profile of ACE [[I-D.ietf-ace-oauth-authz](#)] to use between Client and KDC (see [Section 4.2](#)).
- o OPT1: Optionally, specify the encoding of public keys, of 'client_cred', and of 'pub_keys' if COSE_Keys are not used (see [Section 4.1.2.1](#)).
- o OPT2: 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](#)).
- o OPT3: Optionally, specify the format and content of 'mgt_key_material' (see [Section 4.1.2.1](#)).
- o OPT4: Optionally, specify policies that instruct clients to retain unsuccessfully decrypted messages and for how long, so that they can be decrypted after getting updated keying material (OPT4).

Appendix B. Document Updates

RFC EDITOR: PLEASE REMOVE THIS SECTION.

B.1. Version -02 to -03

- o Exchange of information on the countersignature algorithm and related parameters, during the Token POST ([Section 3.3](#)).
- o Restructured KDC interface, with new possible operations ([Section 4](#)).
- o Client PoP signature for the Joining Request upon joining ([Section 4.1.2.1](#)).
- o Revised text on group member removal ([Section 5](#)).
- o Added more profile requirements (Appendix A).

B.2. Version -01 to -02

- o Editorial fixes.
- o Distinction between transport profile and application profile ([Section 1.1](#)).
- o New parameters 'sign_info' and 'pub_key_enc' to negotiate parameter values for signature algorithm and signature keys ([Section 3.3](#)).
- o New parameter 'type' to distinguish different Key Distribution Request messages ([Section 4.1](#)).
- o New parameter 'client_cred_verify' in the Key Distribution Request to convey a Client signature ([Section 4.1](#)).
- o Encoding of 'pub_keys_repos' ([Section 4.1](#)).
- o Encoding of 'mgt_key_material' ([Section 4.1](#)).
- o Improved description on retrieval of new or updated keying material ([Section 6](#)).
- o Encoding of 'get_pub_keys' in Public Key Request ([Section 7.1](#)).
- o Extended security considerations (Sections [10.1](#) and [10.2](#)).
- o New "ACE Public Key Encoding" IANA Registry ([Section 11.2](#)).

- o New "ACE Groupcomm Parameters" IANA Registry ([Section 11.3](#)), populated with the entries in [Section 8](#).
- o New "Ace Groupcomm Request Type" IANA Registry ([Section 11.4](#)), populated with the values in [Section 9](#).
- o 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](#)).
- o Improved list of requirements for application profiles (Appendix A).

B.3. Version -00 to -01

- o 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:
Carsten Bormann, Rikard Hoeglund, Ben Kaduk, John Mattsson, Daniel Migault, 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

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

Email: francesca.palombini@ericsson.com

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

Email: marco.tiloca@ri.se

