

Key Provisioning for Group Communication using ACE

[draft-ietf-ace-key-groupcomm-05](#)

Francesca Palombini, Ericsson
Marco Tiloca, RISE

IETF 107, ACE WG, Virtual, Apr 15, 2020

Quick Recap

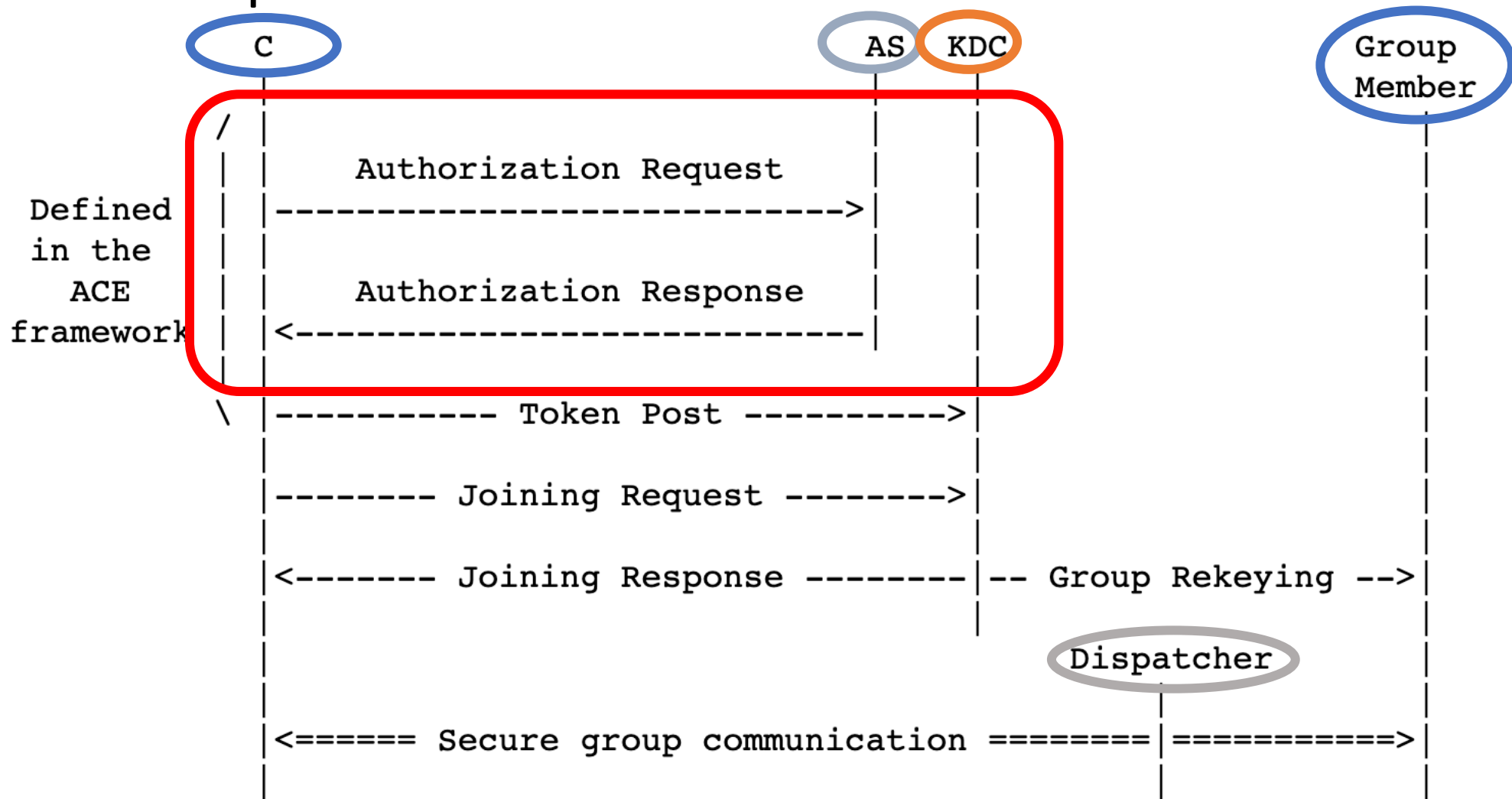
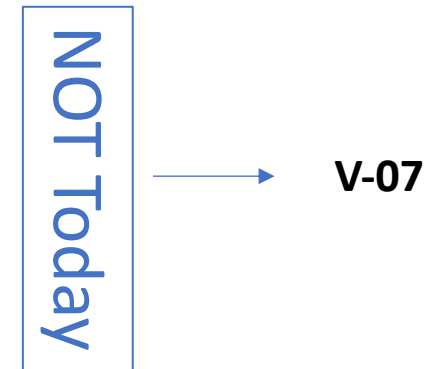
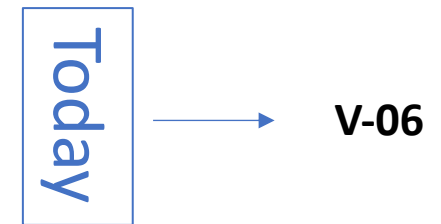


Figure 2: Message Flow Upon New Node's Joining




What's left?

- High level status update (and where to find in-detail updates)
- TODOs left from February's interim
- Updates that need ok from wg (already implemented)
 - "Scope question"
 - "Jim's proposal on legal requestor"
- Open github issues (including open points from latest review)

- Open points from mailing list posts
 - "Keeping the same key identifier for groups"
 - "Considerations on congestion control"
- Couple of open points to check from previous review (outdated?)



TODO left from February Interim

- Token to cover more than one group/topic at once – Issue #42 
- Continue including Peter's review (editorials and clarifications)
https://mailarchive.ietf.org/arch/msg/ace/_PDsf5rnGtVw6y3nSQTJun0t7P4 
- Continue including Jim's review – Issues #46 #51 #53
<https://mailarchive.ietf.org/arch/msg/ace/mR-qwWEhmmOFIVL2ToH5pc2GXqs> 

What happened since IETF 106 – status update

- Version -04 was submitted January 15th – Closed most points from IETF 106
 - Review from Jim: <https://mailarchive.ietf.org/arch/msg/ace/mR-qwWEhmmOFIVL2ToH5pc2GXqs/>
 - Presented at [January ACE interim](#)
 - Discussion at the [February ACE interim](#)
- Version -05 was submitted March 9th
 - Based on the review from Jim.
 - Review from Jim: <https://mailarchive.ietf.org/arch/msg/ace/gSF02hHymShiYkr1pxvqWTNUcP4/>
→ github issues
- Work in progress in PR v-06 on github:
 - <https://ace-wg.github.io/ace-key-groupcomm/v-06/draft-ietf-ace-key-groupcomm.html>
 - Most issues closed
 - Some open points left (later slides)

New issues in github

<https://github.com/ace-wg/ace-key-groupcomm/issues?page=1&q=is%3Aissue+is%3Aopen>

- New labels:
 - “editor done + wg review request”
 - “ongoing”
 - “question”
 - “PR v-06”

- 33 open / 51 closed
 - 27 editor done – please review, explicit “agree, ok to close” help keeping track.
 - 3 open
 - 3 question

Scope question from Jim

(Or how we are dealing with multi-scope tokens)

- Goal: 1 access token covers multiple groups at once. 1 token post for several groups.

- Access token response:

```
sign_info_res = [ + sign_info_entry ]
sign_info_entry = [ id : gid / [ + gid ],
                   sign_alg : int / tstr,
                   sign_parameters : any / nil,
                   sign_key_parameters : any / nil,
                   ? pub_key_enc_res = int / nil ]
```

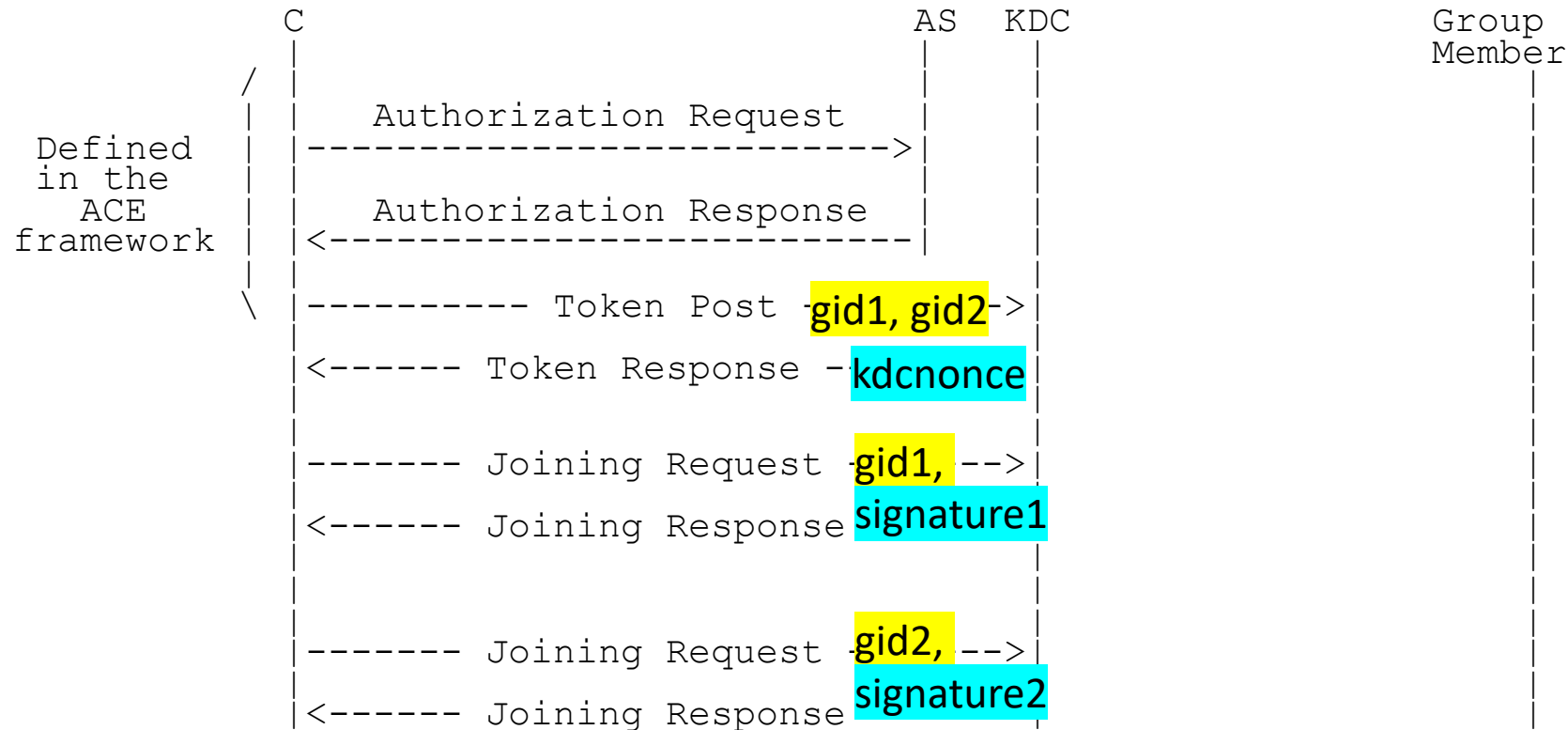
- New format for scope.

```
scp = [ gid , ? ( role / [ 2*role ] ) ]
scope = << [ + scp ] >>
```

- A node can join several groups if they use the same pub key encoding and signing algorithm.
- 1 Token Post followed by several Joining exchanges
- **/*! The same kdcnonce is used to join each different group. /*!**
 - kdcnonce is used as one of the input to the signature for the client to prove possession of its private key (also over client nonce)
 - ace-key-groupcomm-OSCORE does it differently, based on the existing sec association Client-KDC

Scope question

(Or how we are dealing with multi-scope tokens)



Jim's proposal on legal requestor

- Goal: role to be propagated along with the signature public key for each node from the KDC to the server during the joining process.
- Issue #65
- Addition of “peer_roles” parameter to reflect that:

```
pub_keys = <<[key1, key2, key3]>>  
peer_roles = [requester, requester, [requester, responder]]
```

Issue #60 - When are the public key and proof of possession required to be present

question

My main question is: what does this consideration applies to? It could apply to the joining process and to POST to pub-key (where the node post a new public key to the KDC). These are quite perpendicular though (1 -> joining, 2 -> pub-key, 3 -> not supported), so I am not sure what this adds... This could be covering the join – leave – rejoin with the same token process, that is not very well covered right now, is that something we need to expand on?

- No public key with the right parameters has been associated with the token - in this case both the key and proof are required**
*What we do say is that key and proof of possession are **required** during join process if KDC collects and distribute keys, and the client is a sender. So one key will always be associated with the token. 1 is a quite convoluted way to say: “always at joining” (which we already say...) or is this supposed to cover another case?*
- One public key with the right parameters has been associated with the token - in this case the following would seem to be fine**
Are we talking about after joining, posting a new pub key to the KDC? Then both are needed. Unless the key is the same...in which case it is useless to repost the same to pub-key.
 - a) neither are present,
 - b) just the key is present (proof already has been done),
 - c) both are present
- Multiple public keys with the right parameters have been associated with the token - In this case the following would seem to be fine**
The KDC only keeps one public key per node (see pub-key handler), this is not supported. In what case would the client want to associate more than one key? Is this something we want?
 - a) just the key is present (proof has already been done) or
 - b) both are present

Such a change would seem to maximize the number of times that the public key can be omitted and an even greater number of times the proof can be omitted. In the event that the proof is needed, the error would return a new server nonce to be used in the re-sent join message.

*I think right now we already cover the case where the key and proof **MUST** be sent in each section. What are we missing?*

Issue #83 - Section 4.1.7.1 - Group rekeying if a public key is replaced

question

In section 4.1.7.1 (POST to pub-key)

If a public key is replaced, does this constitute a reason that a key roll over is going to be required? Is this something that needs to be distributed proactively if it is allowed by the server since otherwise things may be really messed up?

The answer is not necessarily, but nothing stops the KDC to do a key rollover if it wants.

I think this should be already clear, do we need any specific clarifications?

Issue #88 - Section 7 - Expand the first paragraph.

question

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 [\[RFC8613\]](#).

This need to be explained a bit more. I know what this is talking about but my first thought was how can it be a replay if this is the first time I have seen it? This should probably also note the possible mitigation that it is limited by the time since the last time the key was rolled over.

So, the additional detail should cover:

- A mechanism like Appendix B.2 of OSCORE are useful in case of lost of context and reboot, so that the recipient can be on the safe side.
- If the group is rekeying, the time window between the end of the rekeying and a next loss of context is safe to go. A group rekeying puts the recipients on the safe side.

Is that it?

Issue #77 - Section 4.1.3.1 - FETCH of public keys based on node roles

I would like to have the ability to do a fetch based on the roles of the client associated with the public key. That is I would like to retrieve only those public keys associated with a requestor.

Right now: FETCH to /ace-group/GROUPNAME/pub_key with payload:
[node1, node2, node3]

Proposal: FETCH to /ace-group/GROUPNAME/pub_key with payload:

[[role1, role2], [node1, node2, node3]] → this means I want all keys for the nodes having roles “role2” OR “role2”, and for the nodes “node1”, “node2”, “node3”

NOTES

- this is OK: [[], [node1, node2, node3]]
- this is also OK (combination of roles): [[[role1, role2]], []] and means I want all keys for the nodes having roles “role2” AND “role2”

Issue #84 - Section 4.3 - Two expiration times for the key material

In section 4.3 - I don't know if this is worth while thinking about or not. There may want to be two different expiration values that are defined in the system. When you stop sending using the key material and when you stop receiving using the keying material. These values would need to be contingent on a "key revocation" event as well. The delta could be setup as a new group parameter with a default value set by the document.

- “Contingent on a key revocation event”?
- Easy to add.

Issue #90 - Section 8 - Admit encoding in JSON

In term of the name, what about doing the encoding in JSON?

Are you talking about registering ace-groupcomm+json? We do not define that, and all the ace-groupcomm parameters are defined only for CBOR.

Easy to do.

Is this something we want to add?

Issue #91 - Role/functionality of "proxy signature checker"

Editor done

You do not need to be a member of the group (i.e. having done the join process) to get the public keys, so there is no need to define such a role. As long as you get access to the pub-key resource (by getting authorized by the AS), you can request those.

- Added considerations about that in the text.
- Application profiles can still define a role for those (ace-key-groupcomm-OSCORE)
- Anything we need to add?

Any other issue you'd like to bring up?

(in particular from the editor done if you think the resolution is not sufficient)

Plan forward

- Get ok for all editor done issues
- Include all the issues above
- Merge PR
- Submit v-06

- Work on the rest of the points (interim?)
- Submit v-07