

Key Provisioning for Group Communication Using ACE

draft-ietf-ace-key-groupcomm-11

Francesca Palombini, Ericsson
Marco Tiloca, RISE

IETF 110, ACE WG, March 12th, 2021

Recap

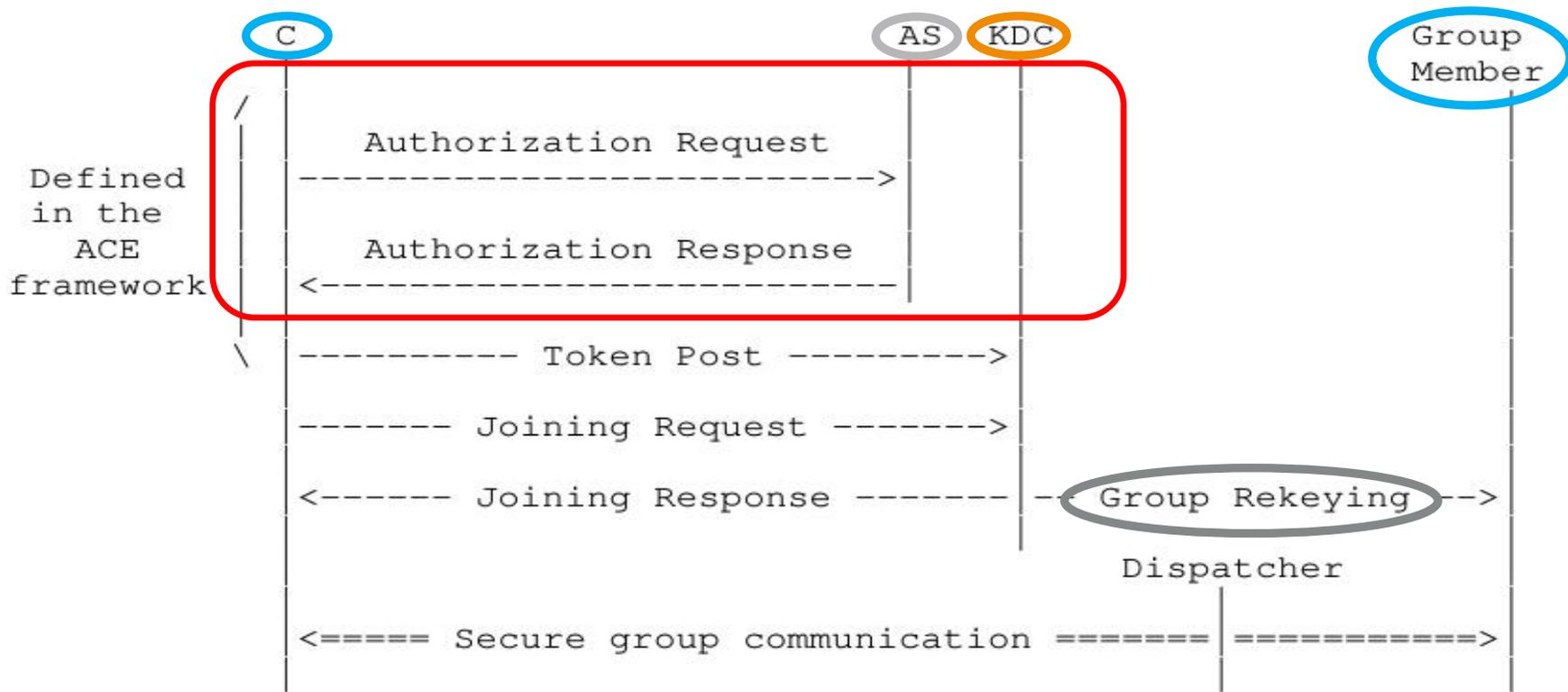


Figure 2: Message Flow Upon New Node's Joining

Updates

- › Addressed all remaining open points
 - From IETF 109 and the latest interim meetings
- › Editorial cleanup and simplifications
- › Renumbering of mandatory and optional requirements
- › 'control_path' parameter renamed to 'control_uri'
- › CoAP methods are just examples of possible operations in groups
- › Possible to observe `ace-group/GROUPNAME/nodes/NODENAME` at the KDC
 - Pro: get an unsolicited 4.04 (Not Found) in case of eviction from the group
 - Non prescriptive suggestion to observe with No-Response: 2, if supported
 - › Avoid 2.xx notifications, as mostly overlapping with notifications from `ace-group/GROUPNAME`

Updates – ‘get_pub_keys’ (1/2)

- › ‘get_pub_keys’: null / [**inclusion-flag**, [roles-filter], [IDs-filter]]
 - New ‘inclusion-flag’
 - › True = Get the public keys of the nodes that have their ID in IDs-filter (if non empty)
 - › False = Get the public keys of the nodes that do **not** have their ID in IDs-filter
- › Kept the rule that ‘roles-filter’ and ‘IDs-filter’ cannot be both empty
- › ‘IDs-filter’ is empty → inclusion-flag = true
- › In the POST request to ace-group/GROUPNAME (Joining Request)
 - Target all group members → ‘get_pub_keys’ : null
 - Target group members with certain roles → ‘get_pub_keys’ : [true, [“role1”, “role2”], []]

Updates – ‘get_pub_keys’ (2/2)

- › In the FETCH request to ace-group/GROUPNAME/pub-key
 - Target members with certain roles
 - › ‘get_pub_keys’ : [true, [“role1”, “role2”], []]
 - Target members with any role and **with** certain IDs
 - › ‘get_pub_keys’ : [true, [], [0x01, 0x7b]]
 - Target members with any role and **without** certain IDs
 - › ‘get_pub_keys’ : [false, [], [0x01, 0x7b]]
 - Target members with certain roles **and/or** **with** certain IDs
 - › ‘get_pub_keys’ : [true, [“role1”, “role2”], [0x01, 0x7b]]
 - Target members with certain roles and **at the same time** **without** certain IDs
 - › ‘get_pub_keys’ : [false, [“role1”, “role2”], [0x01, 0x7b]]
- › Target all group members → GET request to ace-group/GROUPNAME/pub-key

Updates – Pub key encoding

- › ‘pub_keys’ includes public keys of group members in:
 - The Joining Response from ace-group/GROUPNAME
 - The response from ace-group/GROUPNAME/pub-key
- › If COSE Keys are used, ‘kid’ specifies the ID of the associated group members
- › If using a different key wrapper that can’t embed node identifiers ...
 - We have to provide node identifiers in a separate parameter
- › Added an optional parameter ‘peer_identifiers’, for responses with ‘pub_keys’
 - CBOR array, with elements corresponding to elements of ‘pub_keys’, in the same order
 - Used only where the public key encoding does not embed the node identifier

Updates – Error handling

- › Discussion started for the PUT handler of ace-group/GROUPNAME/nodes/NODENAME
 - Return 4.00 (Bad Request), if the payload is not empty as expected.
 - Return 5.03 (Service Unavailable) if a new individual key material (e.g., OSCORE Sender ID) cannot be assigned at the moment – But 5.03 can really mean anything, if not clarified ...
- › Now error responses are more structured, when possible
 - This applies to several 4.xx responses
- › Error responses can have a CBOR map as payload
 - {error: int, ?error_description: tstr}
 - Same ct application/ace-groupcomm+cbor
 - New “ACE Groupcomm Errors” registry, for ‘error’ values
 - Six error types defined in this document

- | | |
|---|--|
| 0 | Operation permitted only to group members |
| 1 | Request inconsistent with the current roles |
| 2 | Public key incompatible with the group configuration |
| 3 | Invalid proof-of-possession signature |
| 4 | No available node identifiers |
| 5 | Group-membership terminated |
| 6 | Group deleted |

Comments? Objections?

Updates – Extended scope (1/2)

The KDC may act also as RS for other resources, accessible via other applications.

C → KDC : POST /authz-info , with ‘scope’ as a CBOR byte string in the Token

How does the KDC know the semantics of scope at this point ?

- How does the KDC know how to parse and interpret the scope from the Token?
- How does the KDC know which possible application profile of ACE should be used?
 - › Etc: for ace-key-groupcomm , the CBOR byte string wraps CBOR array, which contains ...
- Arguable workaround: use different values of “audience” as a hint
 - › This requires ad-hoc agreement between each pair RS-AS (e.g. upon RS registration)

!/ General problem for RSs supporting several applications and application profiles !/

Updates – Extended scope (2/2)

- › From interim meetings: try to draft an extended format of scope, combining:
 - A high-level signaling of “typed scope”, through a single CBOR tag
 - A detailed signaling of the exact scope type, through an integer
- › Optional and only for the ‘scope’ claim in the Token
- › Not limited to application profiles of this document or to applications using group communication
- › Current proposal
 - Prepare the **actual scope**, just as usual
 - Signal the **scope’s semantics** as an integer
 - › Registered by applications and application profiles
 - › Registry “ACE Scope Semantics” defined in this document
 - Build a CBOR **sequence** : [semantics, scope]
 - Wrap the sequence in a CBOR byte string and tag it
 - Include the result in the ‘scope’ claim of the Token

```
gname = tstr
permissions = uint . bits roles

roles = &(
  Requester: 1,
  Responder: 2,
  Monitor: 3,
  Verifier: 4
)

scope_entry = AIF_Generic<gname, permissions>
scope = << [ + scope_entry ] >>
semantics = int
; This defines an array, the elements
; of which are to be used in a CBOR Sequence:
sequence = [semantics, scope]
extended_scope = #6.TBD_TAG(<< sequence >>)
```

Comments? Objections?

Next steps

- › Address possible feedback from today
- › Ready for WGLC ?

Thank you!

Comments/questions?

<https://github.com/ace-wg/ace-key-groupcomm>