Key Provisioning for Group Communication using ACE

Work in progress towards:

draft-ietf-ace-key-groupcomm-11

Francesca Palombini, Ericsson
Marco Tiloca, RISE

ACE Interim Meeting, February 11th, 2021

Minor fixes/additions

- 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

New format for ‘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”], []]

New format for ‘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

Comments? Objections?

Public Key encoding with no ID

- ‘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

Comments? Objections?

Error handling

- In 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.

- Suggestion to make error response more structured when possible
  - For example, 5.03 can mean anything if not clarified
  - Actually, the same applies to several other 4.xx responses

- Error responses can have a CBOR map as payload
  - {error: int, ?error_description: tstr}
  - Same ct application/ace-groupcomm+cbor
  - “ACE Groupcomm Errors” registry, for ‘error’ values

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?

Extended scope format (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

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

Extended scope format (2/2)

- From the last interim: 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

- Current proposal
  - Prepare the actual *scope*, just as usual
  - Signal the scope’s *semantics* as an integer
    - Registered by applications and application profiles
  - 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

Comments? Objections?

Should it be a separate document?

```
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 usedin a CBOR Sequence:
sequence = [semantics, scope]

extended_scope = #6.TBD_TAG(<< sequence >>)
```

Next steps

- Address comments and input from today
- Polish the Editor's copy on Github and submit v -11
- If no major issues remain after IETF 110, target WGLC

Thank you!