The Group Object Security for Constrained RESTful Environments (Group OSCORE) Profile of the Authentication and Authorization for Constrained Environments (ACE) Framework

draft-ietf-ace-group-oscore-profile-03

Marco Tiloca, RISE
Rikard Höglund, RISE
Francesca Palombini, Ericsson

IETF 121 meeting – Dublin – November 8th, 2024

Recap

- **Group OSCORE profile of ACE**
  - Enable access control for accessing resources at group members
  - Group OSCORE [1] is the security protocol used between C and RSs
  - The group joining must separately happen first, as defined in [2]
  - An access token is bound to the already existing Group OSCORE Security Context and to the authentication credential AUTH_CRED_C of the client C
- **Properties**
  - Proof-of-Possession of the client's private key
    - Achieved when verifying a first Group OSCORE request from the client
    - Both the group mode and pairwise mode of Group OSCORE are covered
  - The RS achieves Proof-of-Group-Membership for the exact client C
  - Mutual authentication, when completing a first Group OSCORE exchange

[1] https://datatracker.ietf.org/doc/draft-ietf-core-oscore-groupcomm/
[2] https://datatracker.ietf.org/doc/draft-ietf-ace-key-groupcomm-oscore/

Updates in v -03

- **Editorial fixes and improvements, including:**
  - Lowercase use of “client”, “resource server”, “authentication server”, and “access token”
  - Aligned section numbers of referred documents, especially *draft-ietf-core-oscore-groupcomm*
  - Amended definition of new entries in IANA registries

- **Clarification in Sections 2.2 and 3.3**
  - Possible to use the alternative workflow defined in *draft-ietf-ace-workflow-and-params*
  - Not further detailed in this document

- **Clarification in Section 4.3**
  - If an uploaded access token later becomes invalid, …
  - … the client MAY ask the AS for a new one for re-enabling access at the RS

Updates in v -03

- **Clarification in Section 4.5**
  - More details on the RS verifying access rights, when receiving a protected request
  - Spelled-out what the RS looks for and uses, to retrieve the stored access token to check

- **New Section 8 “Guidelines on Using Multiple Profiles”**
  - In parallel with this profile, the RS might use additional profiles
  - Goal: establish additional, one-to-one OSCORE associations with specific clients
    - E.g., OSCORE profile, or EDHOC and OSCORE profile
  - The client cannot negotiate a profile to use when requesting the access token, …
  - … but somehow the client can ask for a specific profile to use for that token
  - We suggest to use a particular value of ‘audience’ in the Access Token Request, e.g.:
    - audience=“rs1_GP_OSC” to use the Group OSCORE profile from this document
    - audience=“rs1_OSC” to use the OSCORE profile from RFC 9203

Updates in v -03

- **Prevented (very unlikely) ambiguities**
  - A client can be a member of multiple OSCORE groups under different Group Managers, and:
    - The same Group Identifier GID* is currently used in all those groups
    - The client currently uses the same OSCORE Sender ID SID* in all those groups
    - The client uses the same authentication credential AUTH_CRED_C in all those groups

- **What would happen**
  - The client can obtain two access tokens T1 and T2
  - Both T1 and T2 would be tied with (GID*, SID*, AUTH_CRED_C)
  - An RS in those groups would associate both T1 and T2 with multiple groups

- **Goal: the RS associates each access token with a single OSCORE group**

Updates in v -03 (continued)

- **Rationale of the solution**
  - Prevent such problematic access tokens from being issued and accepted altogether
    - While still keeping the AS agnostic of the OSCORE groups and the Group Managers (GMs)

- **Client side – Section 3.1**
  - The client must locally check if it has multiple ambiguous memberships (see previous slide)
  - If no, the client just asks for the access token as usual
  - If yes, the client must first make its memberships non ambiguous
    - Leave some groups, or change Sender ID in some groups (by re-joining or not)

- **RS side (simplified explanation) – Section 4.2**
  - The RS considers the groups such that: i) it is a member; ii) GID* from the access token is used as Group Identifier; iii) the audience in the access token is consistent with using that security group
  - The RS checks with the responsible GMs to confirm AUTH_CRED_C from the access token
  - The RS accepts the access token only if exactly one positive confirmation is received
  - Extended: access token associated with (GID*, SID*, AUTH_CRED_C, AUTH_CRED_GM)

Updates in v -03

- **Storing multiple access tokens per PoP-Key on the RS – Section 4.6**
  - When using this profile, it's quite likely that the RS might have to do that
  - A client C with AUTH_CRED_C including a PoP key …
  - … might obtain a Token T1 and a Token T2, both bound to AUTH_CRED_C

- **RFC 9200 gives recommendations on this topic**
  - An RS is recommended to store only one access token PoP key
  - At IETF 120, Tim Hollebeek asked to clarify about any possible security issue with this deviation

- **After re-checking, the original recommendation in RFC 9200:**
  - Is mostly about limiting storage and avoiding potentially complicated evaluations of access rights
  - Is simple, but overly restrictive. While in the same spirit, we define new recommendations for this profile.

When using this profile:

Store only one access token that is: i) bound to a specific PoP key; ii) intended to a specific audience; iii) related to a specific OSCORE group

When using another profile P in parallel:

Store only one access token that is: i) issued for the profile P; ii) bound to a specific PoP key; iii) intended to a specific audience; iv) related to a specific secure association for C-RS

Next Steps

- **Enable dynamic update of access rights**
  - Follow the roadmap in Section 3.3.1
  - Use new parameters defined in *draft-ietf-ace-workflow-and-params*
- **Add considerations on interleaving group rekeying**
  - A group rekeying can occur between the Access Token Request and the access token upload, …
  - … with consequent change of Group Identifier in that group
- **Add considerations on group members that are "silent servers"**
  - By definition, they cannot reply with responses protected with Group OSCORE
- **Consider setups with multiple application groups and security groups**
  - A client might need an access token spanning multiple application/security groups
  - Currently, the access token can only target one security group
- **Comments and reviews are welcome!**

Thank you!

Comments/questions?

https://github.com/ace-wg/ace-group-oscore-profile

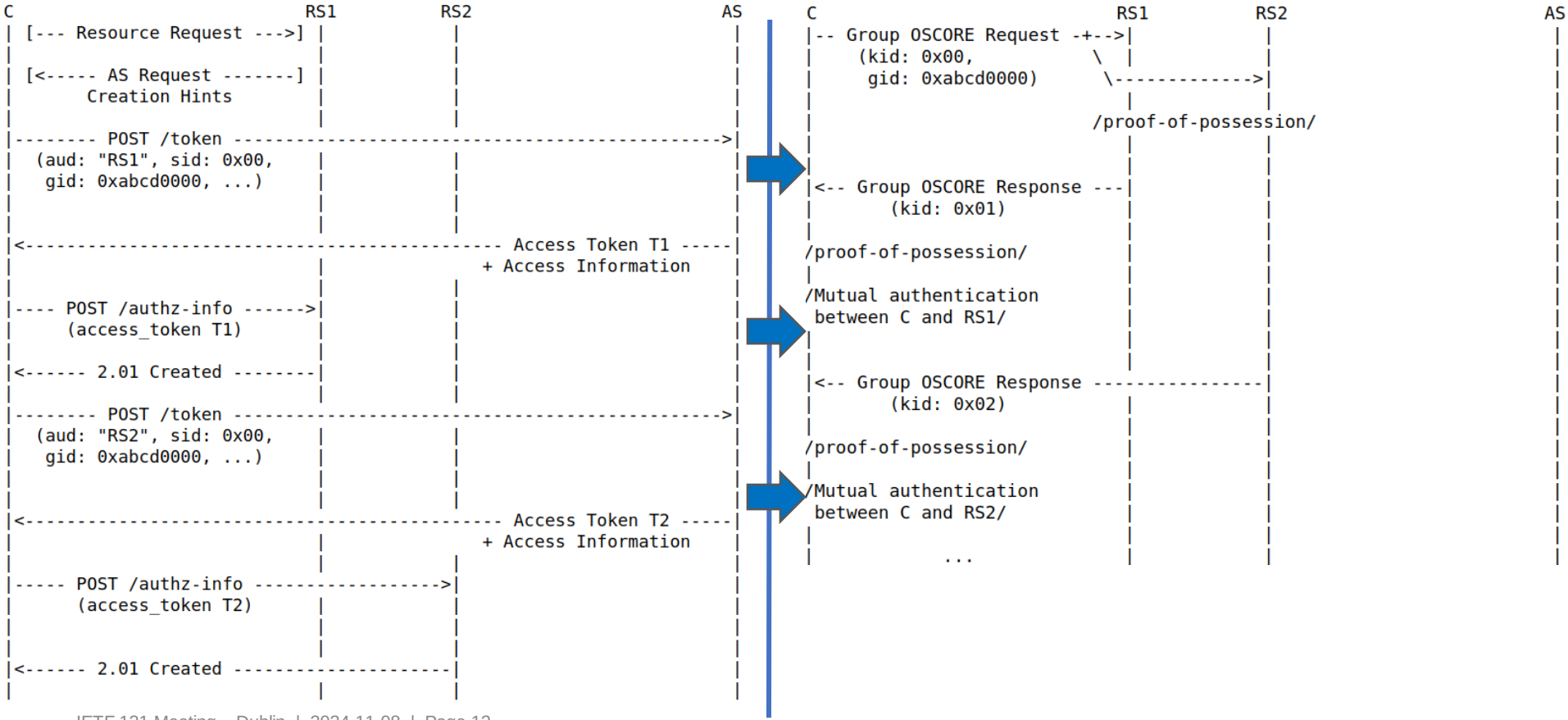
Backup

Applicability and Features

- **For application scenarios relying on group communication**
  - A client wants to access resources at multiple resource servers
  - Secure communication by using a shared set of keying material
  - Aims to enforce access control within the group, for resources at servers
- **Separation between group membership and access control**
  - Being a legitimate group member does not naturally imply access rights
  - The following two concepts are separate:
    - access to the secure group communication channel (through membership)
    - access control to the resource space provided by servers in the group - This draft
- **Follows the Zero-Trust paradigm [3]**
  - Focus on resource protection
  - Trust is never granted implicitly, but must be continually evaluated
  - Access control enforcement must be as granular as possible

[3] NIST-800-207: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

Overview - Protocol flow



Detailed message exchange (1/2)

- **The C-to-AS Access Token Request includes also:**
  - 'context_id': Group ID ('kid_context') of the OSCORE group
  - 'salt_input': Client Sender ID ('kid') in the OSCORE group
  - 'client_cred_verify': Client's proof-of-possession evidence
  - 'req_cnf': Client's auth. credential in the OSCORE group

- **Proof-of-possession evidence in 'client_cred_verify'**
  - Computed with the Client's private key used in the OSCORE group

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: 19 (application/ace+cbor)
Payload:
{
  / audience /        5 : "tempSensor4711",
  / scope /           9 : "read",
    e'context_id_param' : h'abcd0000',
    e'salt_input_param' : h'00',
  e'client_cred_verify' : h'c5a6...f100' / elided for brevity /,
  / req_cnf /         4 : {
    e'kccs' : {
      / sub / 2 : "42-50-31-FF-EF-37-32-39",
      / cnf / 8 : {
        / COSE_Key / 1 : {
          / kty /  1 : 2 / EC2 /,
          / crv / -1 : 1 / P-256 /,
          / x /   -2 : h'd7cc072de2205bdc1537a543d53c60a6
                         acb62eccd890c7fa27c9e354089bbe13',
          / y /   -3 : h'f95e1d4b851a2cc80fff87d8e23f22af
                         b725d535e515d020731e79a3b4e47120'
        }
      }
    }
  }
}
```

Access Token Request

- **What is the input to compute the proof-of-possession (PoP) evidence?**
  - If **(D)TLS** is used between C and AS ==> an exporter value (Section 7.5 of RFC 8446)
  - If **OSCORE** is used between C and AS ==> PRK = HMAC-Hash(x1 | x2, IKM)
    - x1 = Context ID of the C-AS OSCORE Security Context ;
    - x2 = Sender ID of C in the C-AS OSCORE Security Context;
    - IKM = OSCORE Master Secret of the C-AS OSCORE Security Context

Detailed message exchange (2/2)

- **Nothing special in the AS-to-C Access Token Response**

```
Header: Created (Code=2.01)
Content-Format: 19 (application/ace+cbor)
Payload:
{
  / access_token / 1 : h'8343a1010aa2044c...00',
  / ace_profile / 38 : e'coap_group_oscore',
  / expires_in /   2 : 3600
}
```

Access Token Response

- **The Access Token includes also:**
  - 'contextId_input' : Group ID of the OSCORE group
  - 'salt_input': Client Sender ID in the OSCORE group
  - 'cnf': Client's auth. credential in the OSCORE Group

- **Token POST to the RS and response from the RS**
  - RS checks C's auth. credential with the Group Manager (GM)
  - RS stores the access token and associates it with the quartet (Group ID; Sender ID; C's auth. Credential; GM's auth. Credential)
  - Another group member cannot impersonate C

```
{
  / aud /           3 : "tempSensorInLivingRoom",
  / iat /           6 : 1719820800,
  / exp /           4 : 2035353600,
  / scope /         9 : "temperature_g firmware_p",
  e'context_id_claim' : h'abcd0000',
  e'salt_input_claim' : h'00',
  / cnf /           8 : {
    e'kccs' : {
      / sub / 2 : "42-50-31-FF-EF-37-32-39",
      / cnf / 8 : {
        / COSE_Key / 1 : {
          / kty /  1 : 2 / EC2 /,
          / crv / -1 : 1 / P-256 /,
          / x /   -2 : h'd7cc072de2205bdc1537a543d53c60a6
                         acb62eccd890c7fa27c9e354089bbe13',
          / y /   -3 : h'f95e1d4b851a2cc80fff87d8e23f22af
                         b725d535e515d020731e79a3b4e47120'
        }
      }
    }
  }
}
```

Access Token