Group OSCORE Profile of the Authentication and Authorization for Constrained Environments Framework

draft-tiloca-ace-group-oscore-profile-05

Marco Tiloca, RISE
Rikard Höglund, RISE
Ludwig Seitz, Combitech
Francesca Palombini, Ericsson

IETF 110, ACE WG, March 12th, 2021
Recap

In application scenarios with group communication:
- Message protection $\rightarrow$ Group OSCORE, i.e. *draft-ietf-core-oscore-groupcomm*
- ACE for group joining and key provisioning $\rightarrow$ *draft-ietf-ace-key-groupcomm-oscore*
- What about access control for accessing resources at group members?

In general, they are two logically separated domains of access control

Once group members, different clients may have different access rights
- Parents/children that can/cannot open smart locks (which in turn cannot open other locks …)
- High- and low-privileged devices in BACnet, e.g. light switch and fire panel
- Creating (many) more groups poorly scales and is hard to manage
  - Changing access rights means changing group and perform rekeying

Missing profile to use Group OSCORE and access control to the resource space
New Group OSCORE profile of ACE

Group OSCORE as security protocol between C-RSs as group members
- The group joining has to happen first
- The Access Token is bound to the Group OSCORE Security Context
- PoP key: the public key that the Client uses in the OSCORE group

Properties
- Proof-of-Possession of the client signature key
  - Achieved when verifying a first Group OSCORE request from the client
- Proof-of-Group-Membership for the exact Client
  - Token bound to the group context
- Mutual authentication, when completing a first exchange

Appendix A – “Dual mode (Group OSCORE & OSCORE)”
- Both OSCORE and Group OSCORE are used as security protocol
- A newly established OSCORE context is bound to the Group OSCORE Security Context
Updates from -05

› Alignment with the latest updates of the OSCORE profile
  – Terminology, considerations and phrasing of security properties

› Token request for update of access rights
  – Explicit definition was required only for the “Dual Mode” in the Appendix
    › The Group OSCORE context is a separate responsibility of the Group Manager
  – It mirrors the OSCORE profile, i.e. a new pairwise OSCORE context is not established

› Addressed comments on v-04 from Christian Amsüss – Thanks!
  – Clarifications on the RS processing
    › Keep an up-to-date association between a Token and the Group-OSCORE-related information of the Client
  – What happens if the Client changes the public key used in the OSCORE group?
Updates from -05

- Association between Token and Group-OSCORE information
  - Token ↔ (GID, SID, Pub_Key)
    - GID: Group ID of the OSCORE group; it changes when the group is rekeyed
    - SID: Sender ID of the Client in the OSCORE group; the Client can get a new one
    - Pub_Key: Public Key used by the Client in the OSCORE group; it can change (see below)
  - As group member, the RS can track changes in GID and SID

- Change of Client’s public key in the OSCORE group (*)
  - The client asks the AS for a new Token, as bound to (GID, SID, Pub_Key_NEW)
  - The client re-runs the profile with the RS
  - The RS replaces the old Token and tuple with the new ones

(*) In the “Dual Mode”, the Client proceeds as when requesting an update of access rights
Summary

› New ACE profile for secure group communication
  – Group OSCORE as security protocol
  – ACE-based access control among group members
  – Appendix: “Dual mode” with Group OSCORE and OSCORE

› The latest revision addresses comments from Christian (thanks!)

› Next step
  – Guidelines on later running the OSCORE profile with the same RS in the group

› Need for document reviews
Thank you!

Comments/questions?
Backup
Use cases

› Simple groups of smart locks
  – Some clients should only check the lock status
  – Some clients can both check and change the lock status
  – The smart locks should be servers only, i.e. cannot lock/unlock each other

› Building automation (BACnet)
  – Light switch (Class C1): issue only low-priority commands
  – Fire panel (Class C2): issue all commands, set/unset high-priority level
  – C1 cannot override C2 commands, until C2 relinquishes high-priority control
  – Goal 1: limit execution of high-priority commands to C2 clients only
  – Goal 2: prevent a compromised C1 client to lock-out normal control
Protocol overview

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
- ‘req_cnf’: Client’s public key in the OSCORE group
- ‘client_cred_verify’: Client’s signature

Signature in ‘client_cred_verify’
- Computed with the signing key in the OSCORE group

What does the Client sign?
- If (D)TLS is used between C and AS, sign an exporter value (Section 7.5 of RFC 8446)
- If OSCORE is used between C and AS, sign PRK = HMAC-Hash(x1 | x2, IKM)
  - x1 = Context ID of the C-AS context ; x2 = Sender ID of C in the C-AS context
  - IKM = OSCORE Master Secret of the C-AS context
Protocol overview (ctd.)

- The AS-to-C Access Token Response includes also:
  - ‘profile’: “coap_group_oscore”

- The Access Token includes also:
  - ‘cnf’: Client’s Public Key in the Group
  - ‘salt_input’: Sender ID of C in the group
  - ‘contextId_input’: Group ID of the group

Token POST and response
- RS checks the public key of C with the Group Manager
- RS stores
  - Access Token;
  - Group ID; Sender ID of C in the group; C Public Key
- Another group member cannot impersonate C
C – RS1 pairing

0: Sender ID (‘kid’) of C in the OSCORE group
abd0000: Group ID (‘kid_context’) of the OSCORE group

C
[--- Resource Request ---->]
[<--- AS Information ------]

POST /token
(aud: RS1, sid: 0, gid: abd0000, ... )

Access Token + RS Information
(aud: RS1, sid: 0, gid: abd0000, ... )

POST /authz-info
(access_token)

2.01 Created
C – RS2 pairing

0: Sender ID (‘kid’) of C in the OSCORE group
abcd0000: Group ID (‘kid_context’) of the OSCORE group

--- POST /token ---
(aud: RS2, sid: 0, gid: abcd0000, ...)

<------------------- Access Token + RS Information ----->
(aud: RS2, sid: 0, gid: abcd0000, ...)

--- POST /authz-info ---
(access_token)

<--- 2.01 Created ---
C – {RS1, RS2}

- Sender ID (‘kid’) of C in the OSCORE group
- Group ID (‘kid_context’) of the OSCORE group

C can access RS1 and RS2 resources, as per the posted Access Token

- Proof-of-possession achieved when verifying the first Group OSCORE request
  - Group mode: signature verification, using the Client’s public key from the Access Token
  - Pairwise mode: message decryption, with the pairwise key derived from C and RS asymmetric keys
“Dual mode”
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’: Client’s public key in the OSCORE group
- ‘client_cred_verify’: Client’s signature

Signature in ‘client_cred_verify’
- Computed with the signing key in the OSCORE group

What does the Client sign?
- If (D)TLS is used between C and AS, sign an exporter value (Section 7.5 of RFC 8446)
- If OSCORE is used between C and AS, sign \( PRK = \text{HMAC-Hash}(x_1 | x_2, \text{IKM}) \)
  - \( x_1 \) = Context ID of the C-AS context ; \( x_2 \) = Sender ID of C in the C-AS context
  - \( \text{IKM} \) = OSCORE Master Secret of the C-AS context

Access Token Request

Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
```
{
  "audience" : "tempSensor4711",
  "acore" : "read",
  "context_id" : "h'abcd0000",
  "salt_input" : "h'00",
  "client_cred" : { "COSK\_Key" : { "kty" : "EC2", "crv" : "P-256", "x" : "h'd7cc072de2205bdc1537a543d53c60a6ac62ecc890c7fa27c9e354089b6e13'", "y" : "h'f95e1d4b861a20c80ff87d8e23f22af725d35e515d020731e79a3b4e47120" }, "client_cred_verify" : "h'...", (signature content omitted for brevity), }
```

The AS-to-C Access Token Response includes also:
- Same OSCORE Security Context Object of the Access Token

The Access Token includes also:
- ‘salt_input’: Client Sender ID in the OSCORE group
- ‘contextId_input’: Group ID of the OSCORE group
- ‘client_cred’: Client’s public key in the OSCORE Group

Token POST and response
- Exchange of nonces N1 and N2 as in the OSCORE profile
- Negotiation of C’s and RS’ IDs, as in the OSCORE profile
- RS checks the public key of C with the Group Manager
- RS stores {Access Token; Sender ID; Group ID; C Public Key}
- Another group member cannot impersonate C

Overview – Δs from OSCORE profile

Header: Created (Code=2.01)
Content-Type: "application/ace+cbor"
Payload:
{
"access_token": h’8343a1010aa2044c53 ...
(remainder of CWT omitted for brevity),
"profile": "coap_group_oscore",
"expires_in": 3600,
"cnf": {
"oso": {
"alg": "AES-CCM-16-64-128",
"id": h’01’,
"ms": h’f9af3836b3e353e78888e1426b94e6f’,
"salt": h’1122’,
"contextId": h’99’
}
}

Access Token Response

{"aud": "tempSensorInLivingRoom",
"iat": "1360189224",
"exp": "1360289224",
"scope": "temperature_g_firmware_p",
"cnf": {
"oso": {
"alg": "AES-CCM-16-64-128",
"id": h’01’,
"ms": h’f9af3836b3e353e78888e1426b94e6f’,
"salt": h’1122’,
"contextId": h’99’
}

"salt_input": h’00’,
"contextId_input": h’abced0000’,
"client_cred": {
"CODE_Key": {
"kty": EC2,
"crv": P-256,
"x": h’d7ce072de2205bdc1537a543d53c60a6acb62e0c8d90c7fa27c9e354089b8e13’,
"y": h’f95e1d4b851a2cc80ff87d9e23f22af725d53ae15d62c731e79a3b4e47120’
}
}
Overview – $\Delta$s from OSCORE profile

Derivation of the pairwise OSCORE Security Context $\text{ctx}$
- Extended parameters, through more concatenations
- Use also information related to the OSCORE Group

**Context ID** = $\text{GID} | N1 | N2 | \text{CID}$
- The Group ID of the OSCORE group is also in the Access Token, as ‘contextId_input’
- The context identifier indicated in the Access Token, in the ‘contextId’ field of ‘osc’

**Salt** = $\text{SaltInput} | \text{MSalt} | N1 | N2 | \text{GMsalt}$
- The Sender ID of C in the OSCORE group is also in the Access Token, as ‘salt’
- The Salt indicated in the Access Token, in the ‘salt’ field of ‘osc’
- The Master Salt in the OSCORE group is known to C and RS as group members

**Master Secret** = $\text{MSec} | \text{GMsec}$
- The OSCORE Master Secret in the Access Token, in the ‘ms’ field of ‘osc’
- The Master Secret of the OSCORE group is known to C and RS as group members
C – RS1 pairing

0: Sender ID (‘kid’) of C in the OSCORE group

abcd0000: Group ID (‘kid_context’) of the OSCORE group

--- Resource Request ---->

[<--- AS Request ------]
Creation Hints

POST /token

(aud: RS1, _sid: 0, gid: abcd0000, ...)

Access Token + RS Information

(aud: RS1, _sid: 0, gid: abcd0000, ...)

POST /authz-info

(access_token, N1, ID1)

--- 2.01 Created (N2, ID2) ---

/Pairwise OSCORE Sec /Pairwise OSCORE Sec
Context Derivation/ Context Derivation/
C – RS2 pairing

0: Sender ID (‘kid’) of C in the OSCORE group

abcd0000: Group ID (‘kid_context’) of the OSCORE group
C can access RS1 and RS2 resources, as per the posted Access Token, using OSCORE or Group OSCORE.