Group OSCORE Profile of the Authentication and Authorization for Constrained Environments Framework
draft-tiloca-ace-group-oscore-profile-02

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

IETF ACE WG, Virtual Interim, May 18th, 2020
Motivation

› Application scenarios with group communication
  – Group OSCORE provides security also over multicast
  – What about access control for resources at group members?

› For very simple use cases
  – Straightforward and plain access control may be just fine
  – Joining the security group is enough to access resources
  – Any group member can do anything at any other group members’ resource

› For more complicated use cases
  – Different clients should have different access rights
  – Creating (many) more groups poorly scales and is hard to manage
    › Changing access rights means changing group and perform rekeying
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

› Use ACE to enforce fine-grained access control. However …
Problem

› Every current profile of ACE
   – Does not cover secure group communication between C and RSs
   – Relies on a single security protocol between C and RS

› OSCORE profile
   – C and RS must use OSCORE
   – The Token is bound to the OSCORE Security Context
   – Group OSCORE is simply not admitted

› We cannot use Group OSCORE and ACE-based access control of resources
Contribution

› New Group OSCORE profile of ACE
   – Group OSCORE as security protocol between C and RS
   – ACE-based access control among group members
     › The group joining has to happen first
   – The Access Token is bound also to the group context

› Properties
   – Proof-of-Possession of the client signature key
     › Use the Client’s public key to verify the signature of group messages (*)
   – Proof-of-Group-Membership for the exact Client
     › Token bound to the group context

(*) In the group mode of Group OSCORE
Updates from -01

› Clarified event timeline – Requested by Ben at IETF 106
  – Nodes have to join the OSCORE group first
    › That requires access control at the Group Manager
    › Out of scope for this document, defined in ace-key-groupcomm-oscore
  – This profile focuses on access control among current group members

› Simplified profile – Thanks Göran!
  – Current document body: Group OSCORE as only security protocol
  – The Client’s public key used in the group acts as actual PoP key
  – Message format and examples adapted accordingly

› New Appendix – “Dual mode”
  – Essentially the document body of -01, building on the OSCORE profile
  – Both OSCORE and Group OSCORE are used as security protocol
  – A newly established OSCORE context is bound to the group context
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

Header: Created (Code=2.01)
Content-Type: "application/ace+cbor"
Payload:

```
{  "access_token" : h’a5037674656d7053656e73 ...’
   (remainder of CWT omitted for brevity),
   "profile" : "coap_group_oscore",
   "expires_in" : 3600,
}
```

Access Token Response

```
{
   "aud" : "tempSensorInLivingRoom",
   "iat" : "1360189224",
   "exp" : "1360289224",
   "scope" : "temperature_2_firmware",
   "cnf" : {
      "COSE_Key" : {
         "kty" : "EC2",
         "crv" : "P-256",
         "x" : "h’d7cc072de2205bdc1537a543d53c6a6ac62eccd890c7fa27c9e354089bb13’,
         "y" : "h’f95e1d4b851a2cc80fff87d8e23f22af725d535e515d020731e79a3b4e47120’
      },
     "salt_input" : h’00’,
     "contextId_input" : h’abcd0000’
}
```

Access Token
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 Information ------]

----- POST /token ----->
(aud: RS1, sid: 0, gid: abcd0000, ... )

<------------------------ Access Token + RS Information ---------------------->
(aud: RS1, sid: 0, gid: abcd0000, ... )

---- 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
C – \{RS1, RS2\}

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

› Proof-of-possession achieved when receiving a Group OSCORE message
  – Signature verification, using the Client’s public key from the Access Token
Summary

› New ACE profile for secure group communication
  – Group OSCORE as security protocol
  – ACE-based access control among group members
  – The Access Token is bound also to the group context
  – Appendix: “Dual mode” for OSCORE + Group OSCORE

› Next steps
  – Align with latest Group OSCORE (PoP through signature/pairwise mode)
  – Guidelines on later running the OSCORE profile with the same RS

› Need for document reviews
Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-ace-group-oscore-profile
Backup

“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 = 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
The AS-to-C Access Token Response includes also:
- Same OSCORE Sec Ctx Object in 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
- 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
Overview – Δs from OSCORE profile

Derivation of the pairwise OSCORE Security Context $ctxt$
- Extended parameters, through more concatenations
- Use also information related to the OSCORE Group

Context ID = $\text{GID} | \text{N1} | \text{N2} | \text{CID}$
- 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 = $\text{SaltInput} | \text{MSalt} | \text{N1} | \text{N2} | \text{GMsalt}$
- 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

<table>
<thead>
<tr>
<th>C</th>
<th>RS1</th>
<th>RS2</th>
<th>AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>[--- Resource Request --&gt;]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[---- AS Information -----]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POST /token</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(aud: RS1, <strong>sid: 0, gid: abcd0000</strong>, ...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access Token + RS Information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(aud: RS1, <strong>sid: 0, gid: abcd0000</strong>, ...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POST /authz-info</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(access_token, N1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.01 Created (N2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/Pairwise OSCORE Sec</td>
<td>/Pairwise OSCORE Sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context Derivation/</td>
<td>Context Derivation/</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
- **0**: Sender ID ('kid') of C in the OSCORE group
- **abcd0000**: Group ID ('kid_context') of the OSCORE group
C – RS2 pairing

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

C

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

RS1

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

RS2

--- POST /authz-info ---
(access_token, N1)

AS

<--- 2.01 Created (N2)

/Pairwise OSCORE Sec Context Derivation/

Pairwise OSCORE Sec Context Derivation/
C – \{RS1, RS2\}

C can access RS1 and RS2 resources, as per the posted Access Token, using OSCORE or Group OSCORE.

<table>
<thead>
<tr>
<th>C</th>
<th>RS1</th>
<th>RS2</th>
<th>AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>------ OSCORE Request -------&gt;</td>
<td>------ OSCORE Response ------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>?(abcd0000, N1, N2)</td>
<td>------ Group OSCORE Request -------&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------&gt;</td>
<td>(kid: 0, gid: abcd0000) ------ Group OSCORE Response ------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------ Group OSCORE Response ------</td>
<td>(kid: 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------ Group OSCORE Response ------</td>
<td>(kid: 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------ Group OSCORE Response ------</td>
<td>...(continued)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{abcd0000}: \text{Group ID (‘kid_context’) of the OSCORE group} \]

\[ \text{0}: \text{Sender ID (‘kid’) of C in the OSCORE group} \]