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

draft-tiloca-ace-group-oscore-profile-04

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

IETF 109, ACE WG, November 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
  – \textbf{Any} group member can do \textit{anything} at \textit{any} other group members’ resource

› For more advanced 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
Problem

› In general, two logically separated domains of access control
  – To the secure group communication channel ➔ draft-ietf-ace-key-groupcomm-oscore
  – To the resource space provided by servers in the group ➔ Can we use ACE?

› Current profiles of ACE
  – Do not cover secure group communication between C and RSs
  – Rely on a single security protocol between C and RS

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

› Missing profile to use Group OSCORE and access control to the resource space
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 to the Group OSCORE Security Context

› Properties
  – Proof-of-Possession of the client signature key
    › Achieved when verifying a first Group OSCORE request from the client
    › Both the group mode and pairwise mode of Group OSCORE are covered
  – Proof-of-Group-Membership for the exact Client
    › Token bound to the group context
  – Mutual authentication, when completing a first exchange
Updates since -02

› 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 OSCORE Security Context
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 \( PRK = HMAC-Hash(x_1 \mid x_2, IKM) \)
  - \( x_1 = \text{Context ID of the C-AS context} \); \( x_2 = \text{Sender ID of C in the C-AS context} \)
  - \( IKM = \text{OSCORE Master Secret of the C-AS context} \)
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

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

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 – \{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 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
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

› Latest revisions addressed comments from Ben and Göran (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

“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 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 – $\Delta$s from OSCORE profile

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

  - **Context ID** = $GID | N1 | N2 | CID$
    - 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** = $SaltInput | MSalt | N1 | N2 | GMsalt$
    - 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** = $MSec | GMsec$
    - 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 − {RS1, RS2}

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

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