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

draft-tiloca-ace-group-oscore-profile-01

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

IETF 106, ACE WG, Singapore, November 19th, 2019
Motivation (1/3)

› 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
Motivation (2/3)

› 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, thanks Dave!)
  – 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 …
Motivation (3/3)

› 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
  – Builds on the OSCORE profile v -08
  – Admits two security protocols: OSCORE and Group OSCORE
  – Assumes that C and RS have already joined a same OSCORE group

› Outcomes
  – Pairwise OSCORE Security Context \(ctx\)
  – Token bound to both \(ctx\) and the Group OSCORE Security Context \(g_ctx\)
  – \(ctx\) is bound to \(g_ctx\), i.e. \(ctx\) derivation relies also on \(g_ctx\) parameters

› Properties
  – Proof-of-Possession of the OSCORE Master Secret in the Token
  – Server Authentication (through OSCORE or Group OSCORE)
  – Proof-of-Group-Membership for that exact Client (Token bound also to \(g_ctx\)
The C-to-AS Access Token Request includes also:
- ‘salt’: Sender ID (‘kid’) of the Client in the OSCORE group
- ‘context_id’: Group ID (‘kid_context’) of 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:
- Namesake parameters of the OSCORE Sec Ctx Object
- Same OSCORE Sec Ctx Object in the Access Token

The Access Token includes also:
- ‘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 can check 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 (thanks, Jim!)

Overview – Δs from OSCORE profile

```
Access Token Response

Header: Created (Code=2.01)
Content-Type: "application/ace+cbor"
Payload:
{
  "access_token": "h’a5037674656d7053656e73 ...’
  (remainder of access token omitted for brevity),
  "profile": "coap_group_oscore",
  "expires_in": 3600,
  "cnf": {
    "OSCORE_Security_Context": {
      "alg": "AES-CCM-16-64-128",
      "clientId": "b64’q4",
      "serverId": "b64’Qg",
      "ms": "h’f9af83e360e353e70888e1426bd94e6f’",
      "salt": "h’00’",
      "context_id": "h’abcd0000’"
    }
  }
}
```

```
Access Token

{ "aud": "tempSensorinLivingRoom",
  "iat": "1360189224",
  "exp": "1360289224",
  "scope": "temperature_g firmware_p",
  "cnf": {
    "OSCORE_Security_Context": {
      "alg": "AES-CCM-16-64-128",
      "clientId": "client’",
      "serverId": "server’",
      "ms": "h’f9af83e360e353e70888e1426bd94e6f’",
      "salt": "h’00’",
      "context_id": "h’abcd0000’"
    }
  }
}
```

```
Access Token

{ "client_cred": {
  "COSE_Key": {
    "kty": "EC2",
    "crv": "P-256",
    "x": "h’d7cc072de2205bdci1537a543d53c60a6acb62eccd890c7fa27c9e354089bbe13’",
    "y": "h’f95e1d4b551a2cc86ff87d8e23f22af7b725d535e515d020731e79a3e4f7129’"
  }
  }
}
```
Overview – Δs from OSCORE profile

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

› $Context\ ID = \langle Group\ ID\ of\ the\ OSCORE\ group\rangle | N1 | N2$
  – The $Group\ ID\ of\ the\ OSCORE\ group$ is also in the Access Token, as ‘context_id’

› $Salt = \langle Sender\ ID\ of\ C\ in\ the\ OSCORE\ group\rangle | N1 | N2 | \langle Master\ Salt\ in\ the\ OSCORE\ group\rangle$
  – The $Sender\ ID\ of\ C\ in\ the\ OSCORE\ group$ is also in the Access Token, as ‘salt’
  – The $Master\ Salt\ in\ the\ OSCORE\ group$ is known to C and RS as group members

› $Master\ Secret = \langle OSCORE\ Master\ Secret\rangle | \langle Master\ Secret\ of\ the\ OSCORE\ group\rangle$
  – The OSCORE Master Secret is in the Access Token, as ‘ms’ like in the OSCORE profile
  – 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 --->]  RS1  RS2  AS

<<<- 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, N1)

2.01 Created (N2) ------

Pairwise OSCORE Sec  Pairwise OSCORE Sec
Context Derivation/    Context Derivation/

IETF 106 | Singapore | ACE WG | 2019-11-19 | Page 9
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, ...)

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

POST /authz-info
(access_token, N1')

2.01 Created (N2')

/Pairwise OSCORE Sec Context Derivation/

RS1

RS2

AS

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

--- OSCORE Request ---
?

<--- OSCORE Response ---

-- Group OSCORE Request --
(kid: 0, gid: abcd0000)

<--- Group OSCORE Response ---
(kid: 1)

<--- Group OSCORE Response
(kid: 2)
...

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
Summary

› New ACE profile for secure group communication
  – Two security protocols: OSCORE and Group OSCORE
  – The pairwise context and group context are bound to each other
  – The Access Token is bound also to the group context

› Benefits
  – Enables Group OSCORE together with ACE-based access control
  – Builds on the OSCORE profile and its context derivation

› Need for document reviews
Thank you!
Comments/questions?

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