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

draft-ietf-ace-group-oscore-profile-01

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

IETF 119 meeting – Brisbane – March 19\textsuperscript{th}, 2024
Motivation

› Application scenarios relying on group communication
  - A Client can access a resource shared by multiple Resource Servers
  - Secure communication can be achieved using a shared set of keying material
  - Aims to enforce access control within the group, for resources at servers

› For very simple use cases (e.g., basic lighting control)
  - 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 advanced use cases
  - Different clients should have different access rights
  - A more fine-grained approach is necessary
Desired Properties

› 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 control to the secure group communication channel (through membership)
    › access control to the resource space provided by servers in the group - This draft

› Zero-Trust paradigm [1]
  – Focus on resource protection
  – Trust is never granted implicitly, but must be continually evaluated
  – Access control enforcement must be as granular as possible

Overview

› **Group OSCORE profile of ACE**
  - Enables access control for accessing resources at group members
  - Group OSCORE [2] used as security protocol between C and RSs
  - The group joining must separately happen first! (as defined in [3])
  - Access Token is bound to the already existing Group OSCORE Security Context and to the authentication credential of the Client

› **Properties**
  - Proof-of-Possession of the Client 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
  - Proof-of-Group-Membership for the exact Client
  - Mutual authentication, when completing a first Group OSCORE exchange

Overview - Protocol flow

--- Resource Request --->

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

POST /token
  (aud: "RS1", sid: 0x00,
   gid: 0xabcd0000, ...)

Access Token T1 ---+ Access Information

POST /authz-info
  (access_token T1)

2.01 Created

POST /token
  (aud: "RS2", sid: 0x00,
   gid: 0xabcd0000, ...)

Access Token T2 ---+ Access Information

POST /authz-info
  (access_token T2)

2.01 Created

--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)

RS1

RS2

AS

[--- Group OSCORE Request --->
  (kid: 0x00,
   gid: 0xabcd0000)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x01)

RS1

RS2

AS

[--- Group OSCORE Response ---
  (kid: 0x02)
Main Updates in v-01 (1/3)

› Deleting an Access Token does not delete the Group OSCORE Security Context
  – Clarification: if the Client or the RS deletes an Access Token (e.g., the Access Token has expired), it MUST NOT delete the related Group OSCORE Security Context.
  – The Group OSCORE context should not be deleted, as it is needed for Group OSCORE communication

› Renamed TLS Exporter Label for computing the C-AS PoP input
  – Changed name from EXPORTER-ACE-Sign-Challenge-Client-AS to EXPORTER-ACE-PoP-Input-Client-AS
  – More appropriate, as: focusing on the PoP input computation at that step; and irrespective of the signature or MAC computed later as PoP evidence
Main Updates in v-01 (2/3)

- Different computation of the C-AS PoP input when C and AS use (D)TLS 1.2 or 1.3
  - The key exporter takes different input for (D)TLS 1.2 and (D)TLS 1.3

- Revised computation of the C-AS PoP input when C and the AS use OSCORE
  - PoP input is the output PRK of an HKDF-Extract step: $PRK = HMAC-Hash(salt, IKM)$.
  - The used HKDF is the HKDF algorithm specified in the OSCORE Security Context
  - 'salt' takes $x1 \mid x2$
    - $x1$: CBOR byte string with value the OSCORE ID Context, or the CBOR simple value null (0xf6) if no OSCORE ID Context is used
    - $x2$: CBOR byte string with value the OSCORE Sender ID of the Client

- Nits and editorial improvements
Main Updates in v-01 (3/3)

› Removed quotation marks when using CBOR simple values
  – "null" (0xf6) → null (0xf6)

› Updated AS requirements for verifying the PoP evidence
  – The AS must support the format of authentication credential that is used in the OSCORE group
  – If the PoP evidence is a signature: The AS must support the signature algorithm and curve (when applicable) that are used in the OSCORE group
  – If the PoP evidence is a MAC: The AS must support the ECDH algorithm that is used as Pairwise Key Agreement Algorithm in the OSCORE group
  – Practically, not an issue: a C/RS supporting this profile is expected to be registered only at an AS that supports the right algorithms
Summary and Next Steps

› **ACE profile for secure group communication**
  - Fine-grained access control within an OSCORE group

› **Overall, the core of the profile is stable**
  - Refined in the latest version

› **Some planned next steps**
  - Avoid text strings as placeholders for to-be-registered integer abbreviations (use what is proposed in draft-bormann-cbor-e-ref)
  - Support Access Tokens that target RS:s in different OSCORE Groups
  - Revise text on profile requirements from RFC 9200 (see -workflow-and-params)
  - Guidelines to have a follow-up running of the OSCORE profile
Thank you!

Comments/questions?

https://github.com/ace-wg/ace-group-oscore-profile
Backup
Use cases for fine-grained control

› Group of smart locks
  – Some clients should only check the lock status (e.g. for a child's account)
  – Some clients can both check and change the lock status (e.g. for a parent's account)
  – The smart locks should be servers only, i.e. cannot lock/unlock each other

› Building automation (BACnet; with classes of clients)
  – 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
How to accomplish this?

› What about creating a security group for each different set of access rights?
  – It scales poorly and is hard to manage
  – Change of access rights ==> Need to join a different group and to rekey groups

› Better to do it using the ACE framework!
  – Access to secure group communication with draft-ietf-ace-key-groupcomm-oscore[2] OK
    › Provisioning of keying material to communicate in the group with Group OSCORE [3]
  – Fine-grained access to the resource space of the RSs in the group with … … ???

› Current transport profiles of ACE
  – None of them cover secure group communication between C and RSs
  – None of them uses Group OSCORE as security protocol between C and RSs

› The right transport profile is missing

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 auth. credential in the OSCORE group
- ‘client_cred_verify’: Client’s proof-of-possession evidence

Proof-of-possession evidence in ‘client_cred_verify’
- Computed with the private key in the OSCORE group

What is the proof-of-possession (PoP) input to compute the 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 (1/2)
Detailed message exchange (2/2)

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 auth. credential in the OSCORE Group

Token POST and response
- RS checks the auth. credential of C with the Group Manager
- RS stores {Token; Sender ID; Group ID; C’s auth. credential}
- Another group member cannot impersonate C