Key Management for OSCORE Groups in ACE

draft-ietf-ace-key-groupcomm-oscore-03

Marco Tiloca, RISE
Jiye Park, Universität Duisburg-Essen
Francesca Palombini, Ericsson

IETF 106, ACE WG, Singapore, November 19\textsuperscript{th}, 2019
Recap

› Message content and exchanges for:
  – Provisioning keying material to joining nodes and groups (rekeying)
  – Joining an OSCORE group through its Group Manager (GM)
  – More operations for current members at the GM

› Builds on *draft-ietf-ace-key-groupcomm*
  – Agnostic of the ACE transport profile used by C and GM

› Out of Scope:
  – Authorizing access to resources at group members
  – Actual secure communication in the OSCORE group
Open points raised at IETF 105

› Three approaches for C-GM agreement on countersignatures **[ALL ADMITTED]**
  1. Ask during the Token POST, with ‘sign_info’ and ‘pub_key_enc’
  2. Trial & error, with ‘sign_info’ and ‘pub_key_enc’ in a Joining Response
  3. Early group discovery with the CoRE RD and link target attributes

› Encoding of public keys **[SOLVED]**
  – Admitting COSE_Key, future alternatives may be considered
  – No need to created a new registry for encoding signaling

› Proof-of-possession of client’s private key **[SOLVED]**
  – Sufficient to sign a challenge from the GM plus a self-generated nonce
  – Signature included in the Joining Request

› When rekeying the group, the GM **[SOLVED]**
  – MUST preserve the same unchanged Sender IDs for all group members
Selected updates from -02

› Review from Ludwig (-02) – Thanks a lot!

› Simple “group name”
  – Invariant identifier of the OSCORE group
  – Replaces the old zeroed-epoch OSCORE Group ID
  – No more relation with the OSCORE group ID

› Join Resource ➔ Group-Membership resource
  – This is not only about joining anymore
  – Example path /group-oscore/N\_NAME

› Clarifications on the GM behavior
  – Handling of public keys, e.g. compatibility checks
  – Actions upon a node’s joining/leaving, e.g. (de)allocation of Sender ID
Selected updates from -02

› Aligned with the RESTification in `ace-key-groupcomm`

**Response to Token POST**

› `'pub_key_enc' = 1 ("COSE_Key")`
  – From the “CWT Confirmation Method” registry
  – Future new encodings are possible

› `'rs_nonce'`
  – Challenge to sign for the client. **Recommend a size of 8 bytes?**
  – If the Token was conveyed in a DTLS handshake, can ‘rs_nonce’ be a TLS exporter?
Selected updates from -02

Joining Request: POST to /group-oscore/NAME

› Added a client-generated ‘cnonce’
  – Recommend a size of 8 bytes?

› Signature ‘client_cred_verify’
  – Computed over ‘rsnonce’ | ‘cnonce’
  – Computed with the same signing key used in the OSCORE group

Joining Response

› Public keys of group members in ‘pub_keys’
  – The key owner is identified by the Sender ID in the OSCORE group
  – That Sender ID is included in the ‘kid’ field of the respective public key
Selected updates from -02

Req updated material: GET to /group-oscore/NAME

› E.g., failed processing of (many incoming messages); expired material

Req new material: GET to /group-oscore/NAME/node

› E.g., the Sender Sequence Number has wrapped around

› The Group Manager can:
  – Provide a new Sender ID, from which a new Sender Context is derived
  – Respond with an error, and rekey the whole group instead
Selected updates from -02

Req leaving:  POST to /group-oscore/NAME/node

› Like in case of forced eviction, the Group Manager
  – Free up the Sender ID value
  – Delete the public key, unless used in other groups

Req pub keys:  /group-oscore/NAME/pub-key

› GET request   Retrieve all public keys in the group
› POST request   Retrieve the keys of the specified members
  – The Group Manager silently ignores non recognized identifiers
› In the POST request and in the response to GET/POST
  – The key owner is identified by the Sender ID in the OSCORE group
  – That Sender ID is included in the ‘kid’ field of the respective public key
Implementation

› RISE: ongoing development in Californium
   – Build on the ACE implementation
   – Completed joining process, aligned with v -03
   – Support for both the DTLS and OSCORE profile
   – https://bitbucket.org/lseitz/ace-java/

› Other ongoing implementations:
   – From Peter van der Stok, for libcoap (C)
   – From Jim
Summary

› Latest major updates
  – RESTification according to ace-key-groupcomm
  – Use a simple “group name”, unrelated to a (zeroed-epoch) Group ID
  – Clarification on the GM: handling of public keys, local processing, ...

› Open points
  – Size of exchanged ‘rsnonce’ and ‘cnonce’ → 8 bytes ?
  – What replaces ‘rsnonce’, if the DTLS handshake transports the Token?

› Next steps
  – Continue the RESTification redesign
  – Implement post-joining operations
  – Get more reviews and run interop tests
Thank you!
Comments/questions?

https://github.com/ace-wg/ace-key-groupcomm-oscore
Backup
Joining Response message

› Structure of the Joining Response message

– ‘kty’, “Group_OSCORE_Security_Context object”

– ‘k’, Group_OSCORE_Security_Context object
  › ‘ms’, OSCORE Master Secret
  › ‘clientID’, Sender ID of the joining node (if present)
  › ‘hkdf’, KDF algorithm (if present)
  › ‘alg’, AEAD algorithm (if present)
  › ‘salt’, OSCORE Master Salt (if present)
  › ‘contextID’, Group ID
  › ‘rpl’, Replay Window Type and Size (if present)

› ‘cs_alg’, signature algorithm
  › ‘cs_params’, signature parameters (if present)
  › ‘cs_key_params’, signature key parameters (if present)
  › ‘cs_key_enc’, public key encoding (if present)

– ‘profile’, “coap_group_oscore_app”

– ‘exp’, lifetime of the derived OSCORE Context

– ‘pub_keys’, public keys of group members (if present)

– ‘num’, current version of the group keying material

Defined in ace-key-groupcomm

Extends the CBOR-encoded OSCORE Security Context Object of the OSCORE profile

Defined in the OSCORE Profile

Defined here and added to “OSCORE Security Context Parameters” Registry

Defined in ace-key-groupcomm