Key Management for OSCORE Groups in ACE

draft-ietf-ace-key-groupcomm-oscore-02

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

IETF 105, ACE WG, Montreal, July 25th, 2019
Recap

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

› Build on \textit{draf-ietf-ace-key-groupcomm}
  – Agnostic of the ACE profile used by C and GM

› Out of Scope:
  – Authorizing access to resources at group members
  – Actual secure communication in the OSCORE group
Selected updates from -01

› Review from Jim (-01) and Ludwig (-02) – Thanks a lot!

› Renaming
  – Roles: “requester”, “responder”, “monitor”
  – Profile name: “group_oscore_app”

› Consistency with ace-key-groupcomm
  – ‘type’ parameter in any request to a Join Resource
  – Renamed and revised parameter ‘signed_info’

› Provisioning & checking of public keys at the GM
  – Consistency with signature parameters and expected key encoding
  – Check for possible public key already owned for that joining node
Selected updates from -01

› Agreement on signatures
  – ‘sign_info’, i.e. signature algorithm and parameters
  – ‘pub_key_enc’, i.e. encoding of public keys
  – Used in Token POST response and/or Join Response

› Proof-of-possession of private key
  – The Client gets a nonce in response to the Token POST
  – The Client signs the nonce with its own private key
  – The signature is included in ‘client_cred_verify’ of the Join Request
Open point #1

› The Client has to agree with the GM about
  – Countersignature algorithm and parameters
  – Countersignature key parameters
  – Countersignature key encoding, e.g. COSE_Key

› We are defining three approaches
  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 Join Response
  3. Early group discovery with the CoRE RD and link target attributes [1]

› Do we agree on ... ?
  – Keeping all the three approaches
  – Avoid recommending/mandating some

[1] draft-tiloca-core-oscore-discovery
Open point #2

› We are admitting one public key encoding
  – COSE Key, from RFC 8152
  – Registered in “ACE Public Key Encoding Values” [2]

› Right now, we have no more encodings to register

› Do we agree on admitting possible future encodings?
  – What would be a good registration policy?

[2] draft-ietf-ace-key-groupcomm
Open point #3

› Proof-of-possession of the Client’s private key
  – The Client gets a nonce in response to the Token POST, as ‘cnonce’
  – The Client signs the nonce with its own private key
  – The signature is included in ‘client_cred_verify’ of the Join Request

› Signing process
  – Now referring to COSE
  – In fact, it is fine to just sign a byte stream

› Proposal to sign more data, and avoid oracle:
  – Add a further client-generated nonce in the Join Request
  – The signature in the Join Request covers both nonces

› Do we agree that nothing more is needed to be signed?

As also addressed in ace-key-groupcomm
Open point #4

Section 7 “Group Rekeying Process”

– In order to rekey the OSCORE group, the Group Manager distributes a new Group ID of the group and a new OSCORE Master Secret for that group. When doing so, the Group Manager may take a best effort to preserve the same unchanged Sender IDs for all group members.

Should it be required (MUST/SHOULD) instead?
– Pros: avoid side effects on public key retrieval and signature verification

Reasons to keep it best effort
– Pros: flexible refactoring of Sender ID space, e.g. if many nodes leave
– ???

Note: a node can ask for individual rekeying
– E.g., the sequence number wraps-around
– The GM may assign a new Sender ID, rather than rekeying the whole group
Implementation

› RISE: ongoing development in Californium:
  – Build on the ACE implementation
  – Aligned with -01, i.e. basic functionalities
  – Work in progress to support -02 and different ACE profiles
  – https://bitbucket.org/lseitz/ace-java/

› Other ongoing implementations:
  – From Peter van der Stok
  – From Jim

› Early tests during the Hackhathon
  – Exchange of Join Request/Response over OSCORE
Summary

› Latest major updates
  – Parameters for agreements on signature information
  – Proof-of-possession of Clients’ private keys, i.e. sign a nonce

› Open points to address
  – Which agreement methods for signature information?
  – Other public key encodings than “COSE_Key”?
  – More data to protect/involve during PoP of private keys
  – Preservation of same Sender IDs after a group rekeying

› Next steps
  – Simplify/shorten the document
  – Process comments from Ludwig
  – Get more reviews and run interop tests
Thank you!

Comments/questions?

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

Structure of the Join 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)