Key Management for OSCORE Groups in ACE

draft-ietf-ace-key-key-groupcomm-oscore-01

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

IETF 104, ACE WG, Prague, March 29, 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 *ace-key-groupcomm*

› Out of Scope:
  – Authorizing access to resources at group members
  – Actual secure communication in the OSCORE group
Status

› Adopted in December 2018

› Version -00 as simple adopted repost

› Version -01 mostly updates:
  – Format of the Join Response, Group Manager ---> Client
  – Agreement on countersignature algorithm / parameters
  – Related IANA registrations
Updates from v-00

New structure for 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’, countersignature algorithm
  - ‘cs_params’, countersignature parameters (if present)
- ‘profile’, “coap_group_oscore”
- ‘exp’, lifetime of the derived OSCORE Context
- ‘pub_keys’, public keys of group members (if present)
- ...

Defined in ace-key-groupcomm together with IANA Registry

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

Defined in the OSCORE Profile

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

Defined in the OSCORE Profile

Defined in ace-key-groupcomm together with IANA Registry
Updates from v -00

› Upon joining the group, the Client:
  – Provides its own public key, but ...
  – May miss details about countersigning in the OSCORE group

› The Client needs to know before actually joining
  – Three approaches are described

› Approach #1 – Blind attempt
  – The Join Request includes the public key in the preferred format
  – The Group Manager may reply with the new ‘key info’ parameter
    › ‘sign_alg’ and ‘sign_parameters’ (optional)
  – The Client sends a new Join Request, considering ‘key info’
Updates from v -00

› Approach #2 – Negotiation upon Token POST
  – The Client MAY ask for information, including ‘key_info’
    › POST request uses “application/ace+cbor”
    › ‘key_info’ encodes the CBOR simple value Null
  – The reply from the Group Manager includes ‘key info’
    › ‘sign_alg’ and ‘sign_parameters’ (optional)
    › MUST if ‘key_info’ was in the POST request, MAY otherwise
  – The Client sends the Join Request, considering ‘key info’

› Approach #3 – Learn upon discovering the OSCORE Group
  – E.g., using the CoRE RD as in draft-tiloca-core-oscore-discovery
Implementation

› Ongoing development in Californium

› Build on the ACE implementation:
  – https://bitbucket.org/lseitz/ace-java/branch/oscore-joining

› Status:
  – Complete interaction C – AS, with structured ‘scope’
  – Work in progress on the Join Response content
Summary

› 1. Updated structure of the Join Response
   – Extended the OSCORE Security Context Object
   – Specific instance of ‘kty’ and ‘profile’ from draft-ietf-ace-key-groupcomm

› 2. Agreement on countersignature algorithm and parameters
   – Blind attempt upon sending the Join Request
   – Negotiation during the Token POST
   – Contextual with OSCORE group discovery (e.g. through CoRE RD)

› Feedback/comments?
   – Is this a good direction?
   – Are all three agreement methods needed and good to go?
Thank you!

Comments/questions?

https://github.com/ace-wg/ace-key-groupcomm-oscore