Key Update for OSCORE (KUDOS)
draft-ietf-core-oscore-key-update-02

Rikard Höglund, RISE
Marco Tiloca, RISE

CoRE WG interim, September 28th, 2022
Recap

› OSCORE (RFC8613) uses AEAD algorithms
  – Need to follow limits in number of encryptions and failed decryptions, before rekeying
  – Excessive use of the same key can enable breaking security properties of the AEAD algorithm

› (1) Key Update for OSCORE (KUDOS) ==> Today's main focus
  – Renew the Master Secret and Master Salt; derive new Sender/Recipient keys
  – No change to the ID Context; can achieve Perfect Forward Secrecy
  – Loosely inspired by Appendix B.2 of OSCORE

› (2) AEAD Key Usage Limits in OSCORE
  – Defining appropriate limits for OSCORE, for a variety of algorithms
  – Defining counters for key usage; message processing details; steps when limits are reached

*See also draft-irtf-cfrg-aead-limits
Rekeying procedure

- Key Update for OSCORE (KUDOS)
  - Client and server exchange nonces N1 and N2
  - UpdateCtx() function for deriving new OSCORE Security Context using the nonces
  - Extended OSCORE Option

‘x’ byte enriched with additional signaling flags
Open points for today

› Flag bits in the OSCORE Option
   – First byte
   – Bit 'd' in the new second byte

› Single method to update the key material

› No runtime "negotiation" of FS mode or no-FS mode

› Content about key usage limits

› Learning KUDOS support through EDHOC EAD items

› Where to define the update of OSCORE Sender/Recipient IDs
OSCORE flag bits

› Bit 15, namely 'd', has been registered
  – If set to 1, it is a KUDOS message

› Current situation: bits 0 and 1 are Reserved
  – Current text: define bit 1 for signaling a second flag byte (as intended by RFC 8613)

› Alternative approach discussed on the mailing list [1]
  – Define bit 0 for signaling a second flag byte
  – Change the status of bit 1 to "Unassigned"
  – No real plan for bit 0 otherwise --- Only old thoughts on an uncompressed COSE Object
  – Nice to have a consistent "extension pattern" through bits 0/8/16/24/...

› Ok with the alternative approach? If yes:
  – Do Early Allocation of bit 0?
  – Register bits 8/16/24/... already?

[1] https://mailarchive.ietf.org/arch/msg/core/x_1x5a4PV-XcvmLECtsC_CmoYs/
Single method for context update

Current method: `updateCtx()` has two internal paths for key update
- One based on EDHOC-KeyUpdate() (Method 1)
  - When EDHOC was used at first
- One based on a HKDF Extract and Expand (Method 2)
  - When EDHOC was not used at first
- Method 1 implies that the EDHOC session is still valid
  - Otherwise, need to dynamically fallback to Method 2

From IETF 114: then why not only Method 2?
- No additional benefits from EDHOC-KeyUpdate
- Building X_N becomes simpler

Proposed change: `updateCtx()` uses only Method 2

Objections?
"Negotiation" of FS/no-FS mode

- Mode currently signaled through the 'p' bit in the 'x' byte of the OSCORE Option
  - 'p' set to 0 ==> sender's wish to run KUDOS in FS mode (original mode)
  - 'p' set to 1 ==> sender's wish to run KUDOS in no-FS mode
  - If p = 0 in both KUDOS messages ==> use the FS mode
  - If p = 1 in both KUDOS messages ==> use the no-FS mode

- If the initiator uses p = 0 and the responder uses p = 1
  - Abort KUDOS; from now on, the initiator uses p = 1
  - The initiator might not know the responder's capabilities from the start

- Is the above possible, and thus an agreed fallback necessary? (issue #54)

- Does an OSCORE Security Context also have information:
  - On the other peer's support for KUDOS? (answer: "maybe")
  - If yes, also on the other peer's support for the FS mode? (answer: "maybe")
  - If no, should it? That pre-knowledge may not be possible
Split out update of OSCORE IDs?

› Defined method for updating the peers' OSCORE Sender/Recipient IDs
  – Based on earlier discussions on the mailing list [1][2] and on [3]
  – This procedure can be embedded in a KUDOS execution or run standalone
  – This procedure can be initiated by a client or by a server

› Properties
  – The sender indicates its new wished Recipient ID in the new Recipient-ID Option (class E)
  – Both peers have to opt-in and agree in order for the IDs to be updated
  – Changing IDs practically triggers derivation of new OSCORE Security Context

› From IETF 114: split out as a separate draft?
  – This is strictly related to OSCORE, but ...
  – ... not strictly related to KUDOS functionality
  – Thus the KUDOS draft can focus on KUDOS!

[1] https://mailarchive.ietf.org/arch/msg/core/GXsKO4wKdt3RTZnQZxOzRdIG9QI/
[2] https://mailarchive.ietf.org/arch/msg/core/C1wcSF0BUVxDas8BpgT0WY1yQrY/
Signal KUDOS support in EDHOC

› We can register EDHOC an EAD item for signaling KUDOS support
  – A peer learns if the other peer supports KUDOS (and which modes) during EDHOC execution

› Possible semantics:
  – Value 1 -> "Tell me about what you support"
  – Value 2 -> "I do not support KUDOS"
  – Value 3 -> "I support KUDOS in both modes; tell me about you if you haven't already"
  – Value 4 -> "I support KUDOS only in no-FS mode; tell me about you if you haven't already"

Should we do it? Comments?
Relocate content related to limits?

› Current structure of Section 2
  – Section 2.1 - Overview of key usage limits; specific values to follow --- This builds on [1]
  – Section 2.2 - Extensions of the OSCORE Security Context
    › 'exp' in the Common Context; limits and counters in Sender/Recipient Context
  – Section 2.3 - Extensions of the OSCORE message processing
    › On incrementing the counters and when stopping using the current keys

› How to proceed?

1. Keep as is
  › 2.1 ==> Appendix A
2. Move content to an Appendix
  › 2.1 ==> Appendix A
  › Appendix A ==> Appendix A.1
3. Move content to a new draft
  › The whole Section 2? Only part of it?

It was agreed to elaborate on limits and to have all this content in this same document [2]

Thank you!

Comments/questions?

https://github.com/core-wg/oscore-key-update