Key Update for OSCORE (KUDOS)

draft-ietf-core-oscore-key-update-05

Rikard Höglund, RISE
Marco Tiloca, RISE

IETF 117 meeting – San Francisco – July 25th, 2023
Recap

› (1) Key Update for OSCORE (KUDOS)
  – 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
  › This content now lives in the new Informational document draft-ietf-core-oscore-key-limits

› (3) Update of OSCORE Sender/Recipient IDs
  – Exchanging desired new Recipient ID through a new CoAP Option
Rekeying procedure

Key Update for OSCORE (KUDOS)
- Message exchange to share nonces N1 and N2
- Nonces are placed in new field in OSCORE CoAP option
- UpdateCtx() function for deriving new OSCORE Security Context using the nonces and ‘x’ bytes
- Extended OSCORE Option

KUDOS forward message flow

'x' byte contains additional signaling flags
Updates since IETF 116

› Extended considerations on minimum size of nonces N1 & N2
  – Each peer produces a random nonce value (N1 or N2)
  – Use of an 8 byte nonce is now RECOMMENDED
    – Smaller nonce size might be acceptable (e.g., in constrained devices)
  – Extended security considerations on collision risk

› Clarified what has to be written to non-volatile memory
  – To be able to use the forward-secrecy mode, also after reboot
  – What peers MUST store: the immutable parts of the OSCORE Security Context
  – Possible to exclude: the Common IV, Sender Key, and Recipient Key, which can be re-derived

› General improvements and handling of corner-cases
  – E.g., note on retransmitting Request #1 if KUDOS execution fails in the reverse message flow
Updates since IETF 116

› Rekeying when Using SCHC with OSCORE
  – Considerations on how use of SCHC effects KUDOS executions and their cadence
  – Partial IV compression results in smaller IV space which necessitates more frequent rekeying
  – If SCHC context rules are updated, that endpoint must perform a rekeying

› Revised OSCORE IDs update procedure run stand alone
  – Specific criteria for success/failure
  – Revised examples, for both forward and reverse execution flow
  – Added step-by-step narration of the examples

› OSCORE IDs update procedure run integrated in KUDOS
  – Section 5.0: extended guidelines for KUDOS integration
  – New Appendix A: examples for both forward and reverse execution flow
Updates since IETF 116

Recipient-ID Option used for the OSCORE IDs update

<table>
<thead>
<tr>
<th>No.</th>
<th>C</th>
<th>U</th>
<th>N</th>
<th>R</th>
<th>Name</th>
<th>Format</th>
<th>Length</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Recipient-ID</td>
<td>opaque</td>
<td>any</td>
<td>(none)</td>
</tr>
</tbody>
</table>

- C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable
- In general, the option value can now have an arbitrary length
- However, in the context of this document, the option value specifies the new OSCORE Recipient ID that the sender endpoint intends to use.
- Thus, its maximum length is equal to the maximum length of OSCORE Sender/Recipient IDs
Splitting out OSCORE IDs update

Method for updating the OSCORE Sender/Recipient IDs – Section 5
- This procedure can be run stand-alone or embedded in a KUDOS execution
- But fundamentally it is a separate procedure and not related to KUDOS

Shall we also split this out into a separate, WG document?
- From the 2022-09-28 CoRE interim meeting [2]: mild preference or no opinion to split out.
- Conclusion from later discussions: "bring it up again when all the content is included" (which is now!)
- This content has been greatly expanded and now includes all the main information

If we split out also the OSCORE IDs update procedure, this documents would become specifically focused on KUDOS

Thoughts? Objections?

Open Point: Registry for ‘x’ byte

- **Current structure of ‘x’ byte (in the extended OSCORE option)**
  - \(m\): nonce size in bytes minus 1
  - \(p\): peer indicates its wish to run KUDOS in FS/no-FS mode
  - \(b\): peer indicates its wish to preserve ongoing observations
  - Bits 0-1: reserved for future use

- **Defining a registry for the ‘x’ byte (to become ‘x’ field)**
  - Can aid in future extensibility
  - Bit 0 can be "Reserved" (e.g., for future extensions of ‘x’)
  - Bit 1 and 8-63 (?) can be "Unassigned"
  - Thoughts or comments?
Open Point: KUDOS Request target

› What resource should KUDOS Requests target?
   - Should the client target a KUDOS resource, or just any resource?

› Option 1
   - The client must send Requests to a dedicated KUDOS resource (that doesn't produce a payload or act on requests).
   - **Downside**: This may require that the KUDOS resource interacts with methods within the OSCORE-related code. Alternatively, the OSCORE-related code can be aware of which resources are "KUDOS resources".

› Option 2 (like in OSCORE Appendix B.2)
   - The client's Requests can target any resource (existing or not)
   - The server cannot act on this request (in the forward flow)
   - The client must ignore any payload in KUDOS Responses.
   - **Downside**: Likely requires modifications to the OSCORE library itself, not sufficient to just implement a new standalone resource

Thoughts or comments?
Open Point: Unset Notif. Number

› KUDOS allows for retaining ongoing observations after rekeying
  – A consequence of rekeying is creating a new OSCORE Security Context
  – Peers will start over from Sender Sequence Number (SSN) 0 after a rekeying

› Notification Number
  – OSCORE Observe notifications rely on the Notification Number for ordering and anti-replay
  – Practically the Partial IV is used as Notification Number
  – Thus, if peers resets their SSN to 0, the notification with PIV 0 will be rejected by other peer

› Unsetting the Notification Number
  – Setting it to 0 indicates that Partial IV 0 was already received ==> Incorrect replay detection
  – Proposal: The Notification Number must be set as uninitialized on new Context creation

Thoughts or comments?
Summary and next steps

- Address open points from the previous slides
  - Splitting out OSCORE IDs update procedure
  - Creating registry for ‘x’ byte
  - Considerations on Notification Number
  - Mandating to target KUDOS resource with Requests

- Continue processing open issues
  - All documented on the draft Github repository

- Comments and reviews are welcome!
Thank you!

Comments/questions?

https://github.com/core-wg/oscore-key-update
Update of Sender/Recipient IDs

- Method for updating 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
  - Content moved from old appendix to document body and improved (Section 5)

- 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
  - Must not be done immediately following a reboot (e.g., KUDOS must be run first)
  - Offered Recipient ID must be not used yet under (Master Secret, Master Salt, ID Context)
  - Received Recipient ID must not be used yet as own Sender ID under the same triple

- Examples are provided in Sections 5.1.1 and 5.1.2

[1] https://mailarchive.ietf.org/arch/msg/core/GXsKO4wKdt3RTZnOZxOzRd/lG9Q/1
[2] https://mailarchive.ietf.org/arch/msg/core/CcwcSE0/UxvDadS8BogT0WY1yOryY