Key Update for OSCORE (KUDOS)

draft-ietf-core-oscore-key-update-03

Rikard Höglund, RISE
Marco Tiloca, RISE

IETF 115, CoRE WG, November 7th, 2022
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 (**)
  › Excessive use of the same key can enable breaking security properties of the AEAD algorithm*
    – Defining appropriate limits for OSCORE, for a variety of algorithms
    – Defining counters for key usage; message processing details; steps when limits are reached

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

** Candidates for splitting out (see later slides)

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

Extended OSCORE Option

Client-initiated rekeying

```
<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

<- 1 byte -> <----- s bytes -------> <- 1 byte -> <--- m + 1 bytes --->
| s (if any) | kid context (if any) | x (if any) | nonce (if any) |

```

'x' byte contains additional signaling flags
OSCORE flag bits

› Updates to bit registrations, based on discussion and agreement in [1]
  – As before bit 15, 'd', indicates a KUDOS message (presence of nonce and x)
  – Defined bit 0 for signaling a second flag byte (instead of bit 1)
    › No concrete other plan for bit 0 otherwise
    › This is an additional point about KUDOS updating RFC 8613
  – Changed the status of bit 1 from "Reserved" to "Unassigned"
  – Plan to soon request registration of bits 8, 16, 24, 32, 40, 48 as further extension bits

---

[1] https://mailarchive.ietf.org/arch/msg/core/x_lx5a4PV-XcrvmLEcTsC_CmoYs/
Method for context update

Previously `updateCtx()` had two paths for key update
- One based on EDHOC-KeyUpdate() (Method 1)
- One based on Extract and Expand (Method 2)
- Discussed at the CoRE interim meeting on 2022-09-28 [2]
  - Why not keep only Method 2?
  - No additional benefits from EDHOC-KeyUpdate

Now `updateCtx()` relies only on Expand
- Only one code path: simplified implementations
- Building internal value \( X_N \) for key derivation is easier
- No need for fallback or signaling in case EDHOC-KeyUpdate can't be used
- No need to support EDHOC or to think of EDHOC for the original key establishment

```
updateCtx(X, N, CTX_IN) {
  CTX_OUT    // The new Security Context
  MSECRET_NEW // The new Master Secret
  MSALT_NEW  // The new Master Salt
  X_cbor = bstr.cbor X // CBOR bstr wrapping of X
  N_cbor = bstr.cbor N // CBOR bstr wrapping of N
  X_N = X_cbor | N_cbor
  oscore_key_length = < Size of CTX_IN.MasterSecret in bytes >
  Label = "key update"
  MSECRET_NEW = KUDOS-Expand-Label(CTX_IN.MasterSecret, Label, X_N, oscore_key_length)
  = KUDOS-Expand(CTX_IN.MasterSecret, ExpandLabel, oscore_key_length)
  MSALT_NEW = N;
  < Derive CTX_OUT using MSECRET_NEW and MSALT_NEW, together with other parameters from CTX_IN >
  Return CTX_OUT;
}
```

New version of `updateCtx`

Not locked to HKDF anymore

› The updateCtx() function has been generalized
  – Previously, it used specifically HKDF-Expand()
  – Now it uses KUDOS-Expand()
    › Interface to the key derivation function used by OSCORE
  – This ensures flexibility and is future-proof

› If OSCORE uses an HKDF Algorithm …
  – KUDOS-Expand is mapped to HKDF-Expand
  – This would be the typical functionality of OSCORE today

\[\text{KUDOS-Expand}(\text{CTX\_IN.MasterSecret, ExpandLabel, oscore_key_length}) = \\
\text{HKDF-Expand}(\text{CTX\_IN.MasterSecret, ExpandLabel, oscore_key_length})\]

› A potential, future update to RFC 8613 that admits a different KDF for OSCORE …
  – … must define the mapping between that key derivation function and KUDOS-Expand()
Signal KUDOS support in EDHOC

- Defined an EDHOC EAD item for signaling KUDOS support
  - The sender peer indicates if it supports KUDOS and in which modes
  - Peers learn of each other’s KUDOS support during EDHOC execution

- Registered `ead_label` and defined values: ASK, NONE, FULL, PART
  - FULL or PART in EDHOC message_2 also asks the other peer to indicate whether it supports KUDOS in EDHOC message_3

**EAD items are optional data that can be exchanged during an EDHOC execution**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASK</td>
<td>h'' (0x40)</td>
<td>Used only in EDHOC message_1. It asks the recipient peer to specify in EDHOC message_2 whether it supports KUDOS.</td>
</tr>
<tr>
<td>NONE</td>
<td>h'00' (0x4100)</td>
<td>Used only in EDHOC message_2 and message_3. It specifies that the sender peer does not support KUDOS.</td>
</tr>
<tr>
<td>FULL</td>
<td>h'01' (0x4101)</td>
<td>Used only in EDHOC message_2 and message_3. It specifies that the sender peer supports KUDOS in FS mode and no-FS mode.</td>
</tr>
<tr>
<td>PART</td>
<td>h'02' (0x4102)</td>
<td>Used only in EDHOC message_2 and message_3. It specifies that the sender peer supports KUDOS in no-FS mode only.</td>
</tr>
</tbody>
</table>

Thoughts? Objections?
Further updates from IETF 114

› Forbid sending non-KUDOS messages during a KUDOS execution

› In the client-initiated version of KUDOS
  – The server’s Partial IV is included in its KUDOS response message
  – This prevents reusing the same pair (AEAD nonce, key)
  – Later open point on how to better make this a general rule for OSCORE

› Clarify what a CAPABLE and non-CAPABLE device must support
  – Not CAPABLE device MUST support no-FS mode
  – CAPABLE device MUST support FS mode and SHOULD support no-FS mode

› Restructured section about reasons for rekeying

› Improved retention policies of CTX_OLD
Open point: Partial IV in responses

- The Server MUST include a PIV in Response #1
- This prevents a reuse of the same pair (AEAD nonce, key) from the server, as otherwise shown in this table:

<table>
<thead>
<tr>
<th>Peer</th>
<th>Message</th>
<th>Nonce</th>
<th>Sender key from</th>
<th>Pair reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Request #1</td>
<td>A</td>
<td>CTX_1</td>
<td>No</td>
</tr>
<tr>
<td>Server</td>
<td>Response #1</td>
<td>A</td>
<td>CTX_NEW</td>
<td>No</td>
</tr>
<tr>
<td>Client</td>
<td>Request #2</td>
<td>A</td>
<td>CTX_NEW</td>
<td>No</td>
</tr>
<tr>
<td>Server</td>
<td>Response #2</td>
<td>A</td>
<td>CTX_NEW</td>
<td>YES</td>
</tr>
</tbody>
</table>

- This is now an ad-hoc fix for client-initiated KUDOS
  - The server-initiated version does not have this problem.
  - For simplicity, it can be a general update for OSCORE
  - If an OSCORE response is protected with a different Security Context than the corresponding request was unprotected with, the server MUST include its Sequence Number as Partial IV.
  - An exception is Appendix B.2 of RFC 8613, which does not have this problem by construction.

Objections?
Future structure of the document

› Content on AEAD limits – Section 2 and Appendix A
  – Split out into a separate, WG document?
  – From the 2022-09-28 CoRE interim meeting [2]: strong preference to split out.
  – Shall we confirm to do it?

› Method for updating the OSCORE Sender/Recipient IDs – Section 5
  – This can be run stand-alone or embedded in a KUDOS execution
  – Split out into a separate, WG document?
  – From the 2022-09-28 CoRE interim meeting [2]: mild preference or no opinion to split out.
  – We still need work on that section (mainly discuss examples and preserving observation)
  – Proposal: keep for now and bring it up again when the section is completed?

› If both splits happen, this documents would be focused on KUDOS

Main next steps

› Addressed open points from the previous slides
  – Document restructuring/split
  – General rule for Partial IV in responses across a key update

› Text discussing soft limits vs. hard limits
  – Based on feedback from Rafa Marin-Lopez

› OSCORE ID update examples
  – Textual description of provided examples
  – Preservation of ongoing Observation

› 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/GXsKO4wKdl3RTZnOZxOzRdIG9QI/
[2] https://mailarchive.ietf.org/arch/msg/core/ClwcSF0BUVxDas8BpgT0WY1yQrY/