Key Update for OSCORE (KUDOS)
draft-hoeglund-core-oscore-key-limits-02

Rikard Höglund, RISE
Marco Tiloca, RISE

IETF 112, CoRE WG, November 8th, 2021
Recap

› OSCORE (RFC8613) uses AEAD algorithms to provide security
  – Need to follow limits in key usage and number of failed decryptions, before rekeying
  – Excessive use of the same key can enable breaking security properties of the AEAD algorithm
  – Reference draft-irtf-cfrg-aead-limits-03

› (1) Study of AEAD limits and their impact on OSCORE
  – Defining appropriate limits for OSCORE, for a variety of algorithms
  – Defining counters for key usage; message processing details; steps when limits are reached
  – Taking into account John Mattsson's input at the April CoRE interim [1]

› (2) Defined a new method for rekeying OSCORE (KUDOS)
  – Loosely inspired by Appendix B.2 of OSCORE
  – Goal: renew the Master Secret and Master Salt; derive new Sender/Recipient keys from those
  – Achieves Perfect Forward Secrecy

Recap on AEAD limits
- Discussed in draft-irtf-cfrg-aead-limits-03
- Limits key use for encryption (q) and invalid decryptions (v)
- This draft defines fixed values for ‘q’, ‘v’, and ‘l’ and from those calculate CA & IA probabilities
  - IA & CA probabilities must be acceptably low

Now explicit size limit of protected data to be sent in a new OSCORE message
- The probabilities are influenced by ‘l’, i.e., maximum message size in cipher blocks
- Implementations should not exceed ‘l’, and it has to be easy to avoid doing so
- New text: the total size of the COSE plaintext, authentication Tag, and possible cipher padding for a message may not exceed the block size for the selected algorithm multiplied with ‘l’

New table (Figure 3) showing values of ‘l’ not just in cipher blocks but actual bytes
Key limits (2/3)

- Increased value of ‘l’ (message size in blocks) for algos except AES_128_CCM_8
  - Increasing ‘l’ from $2^8$ to $2^{10}$ should maintain secure CA and IA probabilities
  - draft-irtf-cfrg-aead-limits mentions aiming for CA & IA lower than $2^{-50}$
    - They have added a table in that document with calculated ‘q’ and ‘v’ values

$q = 2^{20}$, $v = 2^{20}$, and $l = 2^{10}$

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>IA probability</th>
<th>CA probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEAD_AES_128_CCM</td>
<td>$2^{-64}$</td>
<td>$2^{-66}$</td>
</tr>
<tr>
<td>AEAD_AES_128_GCM</td>
<td>$2^{-97}$</td>
<td>$2^{-89}$</td>
</tr>
<tr>
<td>AEAD_AES_256_GCM</td>
<td>$2^{-97}$</td>
<td>$2^{-89}$</td>
</tr>
<tr>
<td>AEAD_CHACHA20_POLY1305</td>
<td>$2^{-73}$</td>
<td>-</td>
</tr>
</tbody>
</table>

- Intent is to increase 'q', 'v' and/or 'l' further. Should we?
  - Since we are well below $2^{-50}$ for CA & IA currently
Key limits (3/3)

- Updated table of ‘q’, ‘v’ and ‘l’ for AES_128_CCM_8
  - Added new value for ‘v’, still leaving CA and IA less than 2^-50
  - Is it ideal to aim for CA & IA close to 2^-50 as defined in the CRFG document?

<table>
<thead>
<tr>
<th>‘q’, ‘v’ and ‘l’</th>
<th>IA probability</th>
<th>CA probability</th>
<th>‘q’, ‘v’ and ‘l’</th>
<th>IA probability</th>
<th>CA probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>q=2^20, v=2^20, l=2^8</td>
<td>2^-44</td>
<td>2^-70</td>
<td>q=2^20, v=2^20, l=2^6</td>
<td>2^-44</td>
<td>2^-74</td>
</tr>
<tr>
<td>q=2^15, v=2^20, l=2^8</td>
<td>2^-44</td>
<td>2^-80</td>
<td>q=2^15, v=2^20, l=2^6</td>
<td>2^-44</td>
<td>2^-84</td>
</tr>
<tr>
<td>q=2^10, v=2^20, l=2^8</td>
<td>2^-49</td>
<td>2^-90</td>
<td>q=2^10, v=2^20, l=2^6</td>
<td>2^-44</td>
<td>2^-94</td>
</tr>
<tr>
<td>q=2^20, v=2^15, l=2^8</td>
<td>2^-49</td>
<td>2^-70</td>
<td>q=2^20, v=2^15, l=2^6</td>
<td>2^-49</td>
<td>2^-74</td>
</tr>
<tr>
<td>q=2^15, v=2^15, l=2^8</td>
<td>2^-49</td>
<td>2^-80</td>
<td>q=2^15, v=2^15, l=2^6</td>
<td>2^-49</td>
<td>2^-84</td>
</tr>
<tr>
<td>q=2^10, v=2^15, l=2^8</td>
<td>2^-49</td>
<td>2^-90</td>
<td>q=2^10, v=2^15, l=2^6</td>
<td>2^-49</td>
<td>2^-94</td>
</tr>
</tbody>
</table>

- [Updated table of ‘q’, ‘v’ and ‘l’ for AES_128_CCM_8](#)
Key update (1/4)

- Defined a new method for rekeying OSCORE
  - Key Update for OSCORE (KUDOS) - Named procedure
  - Client and server exchange two nonces R1 and R2
  - UpdateCtx() function for deriving new OSCORE Security Context using the nonces
  - Current Sec Ctx (to renew) ==> Intermediate Sec Ctx ==> New Sec Ctx

- Properties
  - Can be initiated by either the client or server
  - Completes in one round-trip (after that, the new Security Context can be used)
  - Only one intermediate Security Context is derived
  - The ID Context does not change
  - Robust and secure against peer rebooting
  - Compatible with prior key establishment using the EDHOC protocol
No more R1 in the Response #1 for the **client-initiated** rekeying
- Just like in OSCORE Appendix B.2
- Simply not needed: Response #1 correlates to Request #1 through the CoAP Token
Key update (3/4)

› Recommendations on minimum length of R1 and R2 values
  – R1 and R1 | R2 are used as nonces
  – Motivation is based on similar considerations for Appendix B.2 in RFC8613
  – We now recommend minimum 8 bytes, *is this sufficient?*
  – Further text needs to be added as in Appendix B.2. e.g. mentioning the birthday paradox

› Currently MUST terminate ongoing observations after rekeying (derived CTX_NEW)
  – Possible to keep them ongoing for a price, i.e. admitting an earlier use of large Partial IVs
  – Possible solution: after a rekeying, the client considers PIV* as the highest req_piv among all the ongoing observations. Then, when the client starts the first new observation, the SSN jumps to PIV*+1, thus every observation request has a PIV greater than PIV*.
  – Drawback: Big jumps in PIV, i.e., faster consumption and larger communication overhead
  – (More complicated solutions like reserving some PIVs in a bit-map is also possible)
  – *Is it worth keeping observations ongoing across a rekeying? Plan is to not keep observations*
Added and discussed 6TiSCH as use case

- 6TiSCH uses OSCORE Appendix B.2 to handle failure events
- If the 6TiSCH JRC severely fails, it can use Appendix B.2 with the pledges (RECOMMENDED)
- The new key update procedure is a good replacement, especially for 6TiSCH
- Among its intrinsic advantages compared to Appendix B.2, it preserves the ID Context across rekeying
  - 6TiSCH uses ID Context as pledge identifier, meaning that:
    - A key update would not change pledge identifier, which remains unchanged in the long run
    - The JRC does not need anymore to do a remapping between new ID Context and pledge identifier
    - ID Contexts and pledge identifiers can be used as intended at setup/deploy time

The update to RFC8613 includes also “deprecating and replacing” its Appendix B.2

- Ok with this?
More general updates

› Improved Table of Content structure
  – Key Limits
  – Current rekeying methods
  – New rekeying methods
    › Building blocks
    › Client-initiated procedure
    › Server initiated procedure
    › Policies
    › Discussion

› Editorial improvements
  – Terminology harmonization
  – Alignment to most recent EDHOC interfaces
  – Use of RFC8126 terminology in IANA considerations
  – Updated title to Key Update for OSCORE (KUDOS) - Feedback on title?
Next steps

› Address open points, including:
  – Material to save to disk to support rebooting
  – Reuse applicable considerations from OSCORE Appendix B.2
  – Update security considerations
  – Further refinement of key limits

› The document foundation and the key update protocol are stable

› Plan to implement

› WG adoption?
Thank you!
Comments/questions?

OSCORE Option update

- OSCORE Option: defined the use of flag bit 1 to signal presence of flag bits 8-15
- Defined flag bit 15 -- 'd' -- to indicate:
  - This is a OSCORE key update message
  - "id detail" is specified (length + value); used to transport a nonce for the key update

![Diagram of OSCORE option value, including 'id detail']