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

```
 0 1 2 3 4 5 6 7  8   9   10  11  12  13  14  15 <----- n bytes ----->
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+---------------------+
|0|1|0|h|k|  n  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | d | Partial IV (if any) |
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+---------------------+

 <- 1 byte -> <----- s bytes ------> <- 1 byte -> <--- m + 1 bytes --->
+------------+----------------------+---------------------------------+
| s (if any) | kid context (if any) | x (if any) | nonce (if any)     |
+------------+----------------------+------------+--------------------+
                                    /             \____
                                   /                   |
                                  /   0 1 2 3 4 5 6 7  |
+------------------+             |  +-+-+-+-+-+-+-+-+  |
| kid (if any) ... |             |  |0|0|b|p|   m   |  |
+------------------+             |  +-+-+-+-+-+-+-+-+  |
```

'x' byte enriched with additional signaling flags

Client-initiated rekeying

```
                     Client                Server
                  (initiator)           (responder)
                        |                     |
Generate N1             |                     |
                        |                     |
CTX_1 =                 |                     |
  updateCtx(X1, N1,     |                     |
            CTX_OLD)    |                     |
                        |                     |
                        |      Request #1     |
Protect with CTX_1      |-------------------->|
                        | OSCORE Option:      | CTX_1 =
                        |   ...               |  updateCtx(X1, N1,
                        |   d flag: 1         |            CTX_OLD)
                        |   X1                |
                        |   Nonce: N1         | Verify with CTX_1
                        |   ...               |
                        |                     | Generate N2
                        |                     |
                        |                     | CTX_NEW =
                        |                     |  updateCtx(Comb(X1,X2),
                        |                     |            Comb(N1,N2),
                        |                     |            CTX_OLD)
                        |                     |
                        |      Response #1    |
                        |<--------------------| Protect with CTX_NEW
CTX_NEW =               | OSCORE Option:      |
 updateCtx(Comb(X1,X2), |   ...               |
           Comb(N1,N2), |                     |
              CTX_OLD)  |   d flag: 1         |
                        |   X2                |
Verify with CTX_NEW     |   Nonce: N2         |
                        |   ...               |
Discard CTX_OLD         |                     |
                        |                     |

// The actual key update process ends here.
// The two peers can use the new Security Context CTX_NEW.

                        |                     |
                        |      Request #2     |
Protect with CTX_NEW    |-------------------->|
                        |                     | Verify with CTX_NEW
                        |                     |
                        |                     | Discard CTX_OLD
                        |                     |
                        |      Response #2    |
                        |<--------------------| Protect with CTX_NEW
Verify with CTX_NEW     |                     |
```

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**

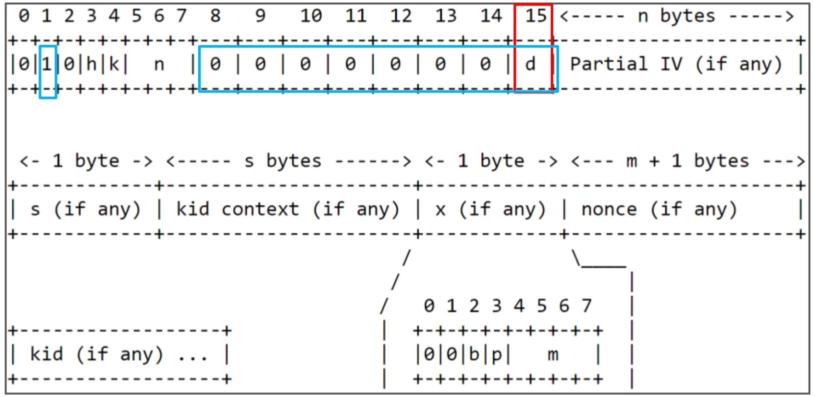
- **No runtime "negotiation" of FS mode or no-FS mode**

- **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



```
 0 1 2 3 4 5 6 7  8   9   10  11  12  13  14  15 <----- n bytes ----->
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+--------------------+
|0|1|0|h|k|  n  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | d | Partial IV (if any) |
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+--------------------+


 <- 1 byte -> <----- s bytes ------> <- 1 byte -> <--- m + 1 bytes --->
+------------+---------------------+------------+---------------------+
| s (if any) | kid context (if any) | x (if any) | nonce (if any)      |
+------------+---------------------+------------+---------------------+
                                  /              \____
                                 /                    |
                                /   0 1 2 3 4 5 6 7   |
+------------------+           |   +-+-+-+-+-+-+-+-+  |
| kid (if any) ... |           |   |0|0|b|p|   m   |  |
+------------------+           |   +-+-+-+-+-+-+-+-+  |
```

- 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 bytes (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/...

Possible to add to the already present update to RFC 8613

- 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_Ix5a4PV-XcrvmLECtsC_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

```
if <the original Security Context was established through EDHOC> {
  // METHOD 1

  // Update the EDHOC key PRK_out, and use the
  // new one to update the EDHOC key PRK_exporter
  (new PRK_out, new PRK_exporter) = EDHOC-KeyUpdate(X_N)

  MSECRET_NEW = EDHOC-Exporter(0, h'', oscore_key_length)
    = EDHOC-KDF(new PRK_exporter, 0, h'', oscore_key_length)

  oscore_salt_length = < Size of CTX_IN.MasterSalt in bytes >

  MSALT_NEW = EDHOC-Exporter(1, h'', oscore_salt_length)
    = EDHOC-KDF(new PRK_exporter, 1, h'', oscore_salt_length)

}
else {
  // METHOD 2

  Label = "key update"

  MSECRET_NEW = HKDF-Expand-Label(CTX_IN.MasterSecret, Label,
                                  X_N, oscore_key_length)
              = HKDF-Expand(CTX_IN.MasterSecret, HkdfLabel,
                            oscore_key_length)

  MSALT_NEW = N;
}
```

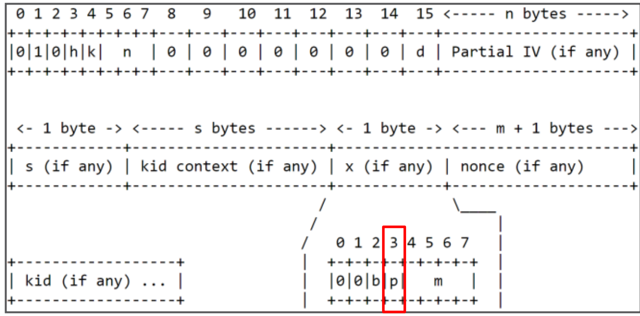
- **From IETF 114: then why not only Method 2?**
  - No additional benefits from EDHOC-KeyUpdate
  - Building X_N becomes simpler

- 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



```
 0 1 2 3 4 5 6 7  8   9   10  11  12  13  14  15 <----- n bytes ----->
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+---------------------+
|0|1|0|h|k|  n  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | d | Partial IV (if any) |
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+---------------------+


 <- 1 byte -> <----- s bytes ------> <- 1 byte -> <--- m + 1 bytes --->
+------------+----------------------+------------+---------------------+
| s (if any) | kid context (if any) | x (if any) | nonce (if any)      |
+------------+----------------------+------------+---------------------+
                                   /              \____
                                  /                    |
                                 /   0 1 2 3 4 5 6 7   |
+-------------------+            |   +-+-+-+-+-+-+-+-+ |
| kid (if any) ...  |            |   |0|0|b|p|   m   | |
+-------------------+            |   +-+-+-+-+-+-+-+-+ |
```

- 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)

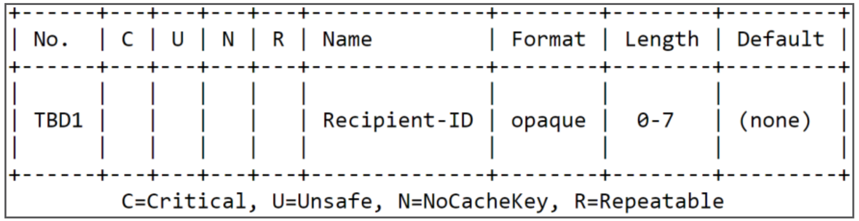
- Does an OSCORE Security Context also have information:
  - On the other peer's support for KUDOS? (answer: "maybe")
  - If yes, also on the other peer's support for the FS mode? (answer: "maybe")
  - If no, should it? That pre-knowledge may not be possible

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!



```
+------+---+---+---+---+--------------+--------+--------+---------+
| No.  | C | U | N | R | Name         | Format | Length | Default |
+------+---+---+---+---+--------------+--------+--------+---------+
|      |   |   |   |   |              |        |        |         |
| TBD1 |   |   |   |   | Recipient-ID | opaque |  0-7   | (none)  |
|      |   |   |   |   |              |        |        |         |
+------+---+---+---+---+--------------+--------+--------+---------+
        C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable
```

[1] https://mailarchive.ietf.org/arch/msg/core/GXsKO4wKdt3RTZnQZxOzRdIG9QI/
[2] https://mailarchive.ietf.org/arch/msg/core/ClwcSF0BUVxDas8BpgT0WY1yQrY/
[3] https://github.com/core-wg/oscore/issues/263#issue-946989659

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. 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]

[1] https://datatracker.ietf.org/doc/draft-irtf-cfrg-aead-limits/
[2] https://datatracker.ietf.org/doc/minutes-interim-2021-core-04-202104281600/

Thank you!

Comments/questions?

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