Key Update for OSCORE (KUDOS)

draft-ietf-core-oscore-key-update-06

Rikard Höglund, RISE
Marco Tiloca, RISE

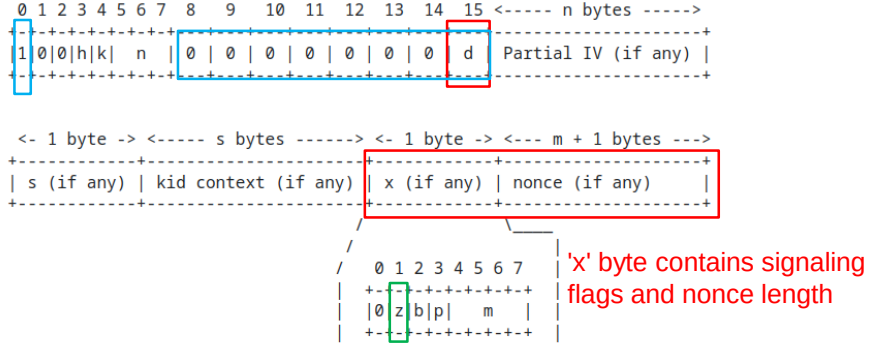
IETF 118 meeting – Prague – November 9th, 2023

Recap

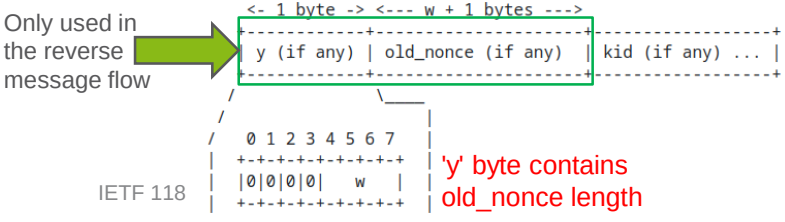
- (1) Key Update for OSCORE (KUDOS)
  - Renew the Master Secret and Master Salt; derive new Sender/Recipient keys
  - No change to the ID Context; can achieve Perfect Forward Secrecy
  - Loosely inspired by Appendix B.2 of OSCORE
- (2) AEAD Key Usage Limits in OSCORE
  - Was split into separate draft as of March 2023: *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

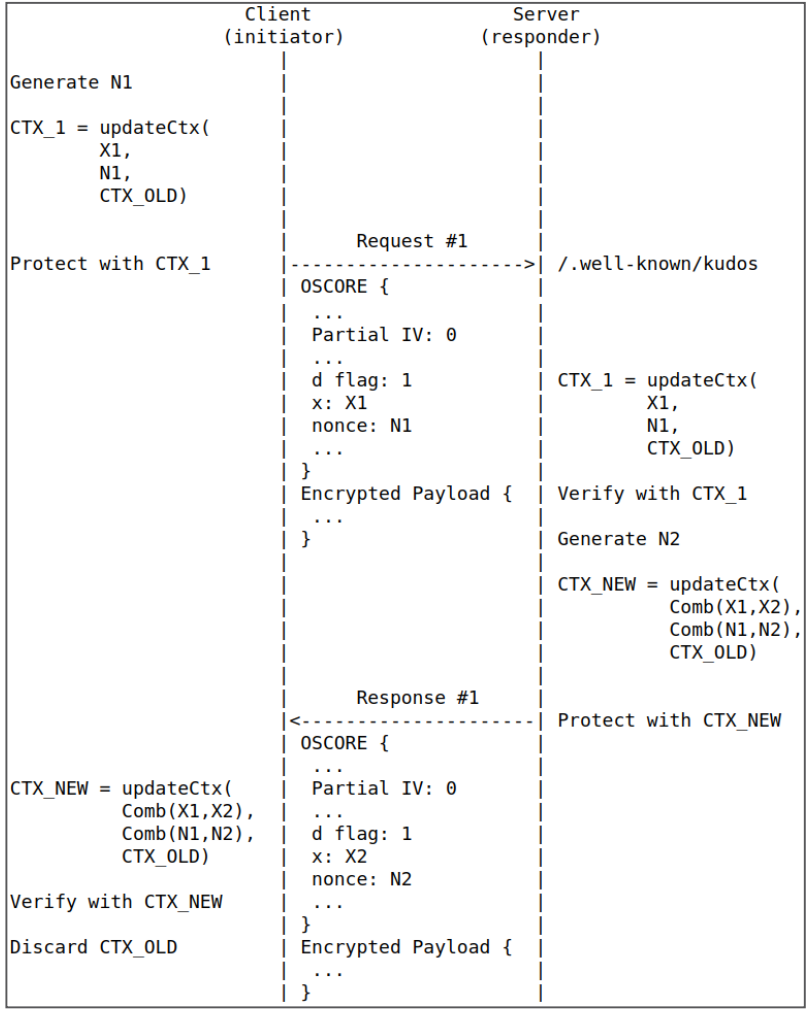
- **Key Update for OSCORE (KUDOS)**
  - Message exchange to share nonces N1 and N2
  - Nonces are placed in new fields in OSCORE CoAP option
  - *UpdateCtx()* function for deriving new OSCORE Security Context using the nonces and ‘x’ bytes
  - Extended OSCORE Option



Only used in the reverse message flow



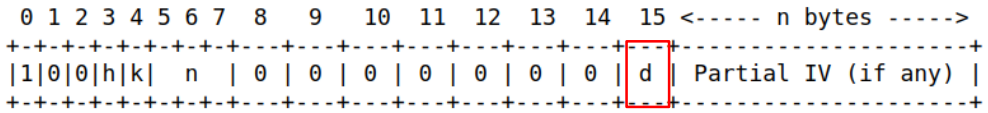
KUDOS forward message flow



Updates since IETF 117

- **Mandate support for both the forward and reverse message flow**
  - A peer that uses KUDOS MUST support both the forward message flow and the reverse message flow
  - Intent is to avoid possible deadlock situations
  - Practically if one flow is supported, supporting the other flow should not add much complexity

- **Clarify the definition of a KUDOS message**
  - A message is a KUDOS message*: if and only if the second byte of flags is present and the 'd' bit is set to 1*



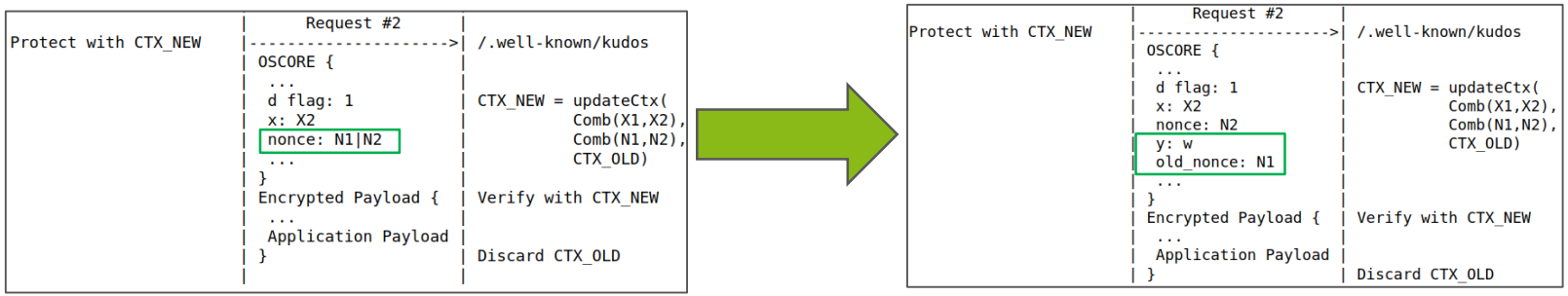
- **Add EDHOC and OSCORE profile of ACE as a method for rekeying**
  - Section 2 lists current methods for rekeying OSCORE
  - This section has now been expanded to also mention the EDHOC and OSCORE profile of ACE

Updates since IETF 117

- **Mandate support for no-FS mode on CAPABLE devices**
  - CAPABLE device: *Can store information in non-volatile memory (e.g., on disk)*
  - No-FS mode: *KUDOS execution mode that does not achieve forward secrecy*
  - We now mandate that CAPABLE devices MUST support both the FS and no-FS mode
    - This is to prevent mismatching of supported mode between peers and thus deadlocks
- **Elaborate on when KUDOS fails during selection of mode**
  - Peers select which mode to execute KUDOS in by using the signaling bit 'p'
  - Added clearer criteria for how to proceed when the indicated bits differ between the peers
  - E.g., how should a mismatch be signaled, and how should peers follow-up (retry)
- **Forbid using old keying material after reboot**
  - Explicitly state that a peer MUST perform a KUDOS execution after reboot before OSCORE communication commences (In order to avoid key and nonce reuse)
  - An exception is when a peer supports a means to do this safely, such as OSCORE Appendix B.1

Updates since IETF 117

- **Further extension of the OSCORE option**
  - To allow transporting N1 in the reverse flow's second KUDOS message
  - Extended the OSCORE option with the 'y' byte and *old_nonce* field
  - N1 must be included in the Request #2 to enable retrieving the correct KUDOS execution
  - Previously N1 and N2 were simply concatenated, which leaves ambiguity and causes issues when the nonce lengths differ



More flexible message flow

- **An execution of KUDOS is based on a request/response message pair**
  - Forward message flow: *KUDOS request* then *KUDOS response*
  - Reverse message flow: *KUDOS response* then *KUDOS request*

- **Shall we allow for more variations of message flows?**
  - Consider a scenario using the Resource Directory where both KUDOS messages are requests

1. EP -> RD: POST /.well-known/rd - **KUDOS message 1 (request)**
2. RD -> EP: GET /.well-known/core - **KUDOS message 2 (request)**
3. EP -> RD: response with payload of /.well-known/core
4. RD -> EP: response to the POST

Credits to Christian Amsüss for this scenario

- **Other scenarios are also possible**
  - Second KUDOS message is a response to a different request than the first KUDOS message
  - Could be the case where there are ongoing observations between the peers

Thoughts/feedback?

KUDOS messages as regular application messages

- **Currently KUDOS messages are 'special' types of messages**
  - For instance, in the forward message flow the client sends the first KUDOS message to */.well-known/kudos,* without any payload

- **Can we instead allow the client to initiate KUDOS with a 'normal' application message?**
  - That is, the client wishes to send an actual application request to the server. Then this message can also serve as a KUDOS message.
  - Implications would be:
    - KUDOS request messages can target any resource at the server
    - In the forward message flow, the client would send the application message that it currently wants to send as a KUDOS message
    - The caveat is that the server cannot be sure the request is fresh, thus it may want to respond with 4.01. This can in turn signal to the client to re-run KUDOS.

Thoughts or comments?

Usage of non-random nonces

- **Currently we always refer to the nonces as random values**
  - However, in some scenarios using counters instead can make sense
  - CAPABLE devices which can persist the context over a reboot may keep a nonce counter. This also ensures that the nonces are not re-used.

- **Shall we explicitly allow usage of counters as nonces?**
  - Currently we do not explicitly forbid, nor allow it
  - Taking RFC9203 (OSCORE ACE profile) as example: *The use of a 64-bit long random number as the nonce's value is RECOMMENDED in this profile.*

- **Proposed solution**
  - Non-CAPABLE devices: MUST use random values
  - CAPABLE devices: SHOULD use random values, but may use counters if it enforces measures to ensure their freshness and accepts the privacy implications

Thoughts or comments?

Splitting out OSCORE IDs update

- **Method for updating the OSCORE Sender/Recipient IDs – Section 5 & Appendix A**
  - The ID update procedure can be run stand-alone or embedded in a KUDOS execution
  - Fundamentally it is a separate procedure and not related to KUDOS
- **Shall we split this out into a separate, WG document?**
  - We raised this in the past but without a strong consensus to proceed
  - Combining Section 5 and Appendix A, this content currently spans around 16 pages
- **One benefit of doing this is to focus this document specifically on KUDOS**

Comments?

Summary and next steps

- **Address open points presented today**
  - More flexible message flow
  - KUDOS messages as regular application messages
  - Usage of non-random nonces
  - Splitting out OSCORE IDs update procedure

- **Implementation**
  - Plan to start an implementation soon

- **Proceed with work on open issues**
  - All are documented on the draft Github repository
  - https://github.com/core-wg/oscore-key-update/issues

- **Comments and reviews are welcome!**

Thank you!

Comments/questions?

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

Backup

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)

No.	C	U	N	R	Name	Format	Length	Default
TBD24					Recipient-ID	opaque	any	(none)

C=Critical, U=Unsafe, N=NoCacheKey, R=Repeatable

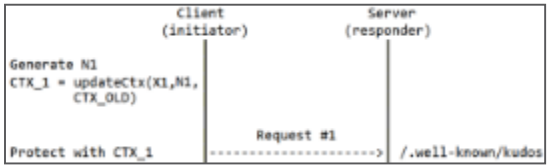
- Properties
  - The sender indicates its new wished Recipient ID in the new Recipient-ID Option (class E)
  - Both peers have to opt-in and agree in order for the IDs to be updated
  - Changing IDs practically triggers derivation of new OSCORE Security Context
  - Must not be done immediately following 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/GXsKO4wKdt3RTZnQZxOzRdIG9QI/
[2] https://mailarchive.ietf.org/arch/msg/core/ClwcSF0BUVxDas8BpgT0WY1yQrY/
[3] https://github.com/core-wg/oscore/issues/263#issue-946989659

KUDOS Request target

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



Forward message flow

- **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?