

Key Update for OSCORE (KUDOS)

draft-ietf-core-oscore-key-update-13

Rikard Höglund, RISE
Marco Tiloca, RISE

IETF CoRE WG meeting – IETF 125 – March 20th, 2026

Recap of KUDOS

› Key Update for OSCORE (KUDOS)

- Renew the Master Secret and Master Salt; derive new Sender/Recipient keys
- No change to the ID Context or Sender/Recipient ID; can achieve Forward Secrecy
- Agnostic of the key establishment method originally used
- Loosely inspired by Appendix B.2 of OSCORE
- The peers update their current context CTX_OLD, deriving a new context CTX_NEW
- Redesigned in v-10 to use a more flexible and simpler approach

› Properties

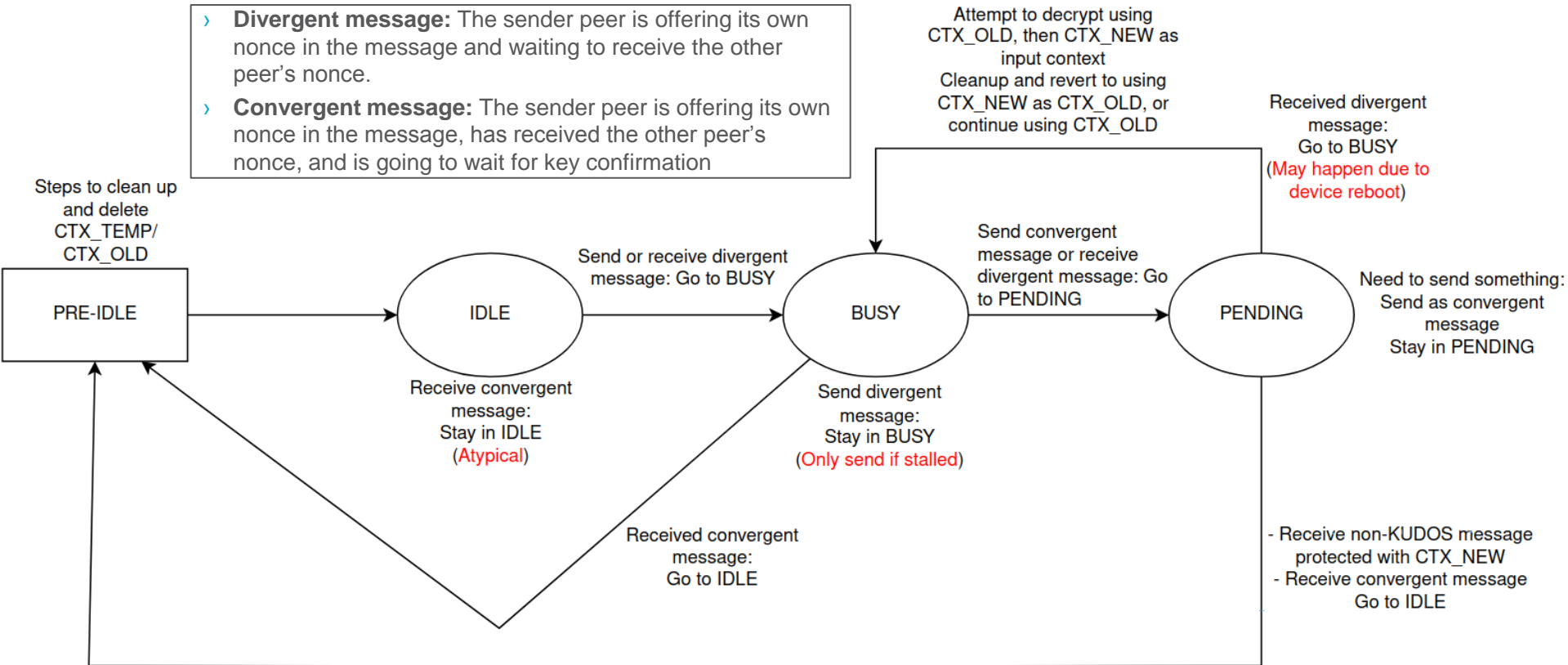
- Can be initiated by either peer
- It is robust against a peer rebooting and loss of state, avoiding the reuse of AEAD (nonce, key)
- It typically completes in one round trip by exchanging two OSCORE-protected CoAP messages
 - The two peers achieve mutual key confirmation in a following exchange, which is protected with the newly established OSCORE Security Context
- Flexible in terms of message flow; any CoAP message can be a KUDOS message

KUDOS States

- › **Three possible states: IDLE, BUSY, and PENDING**
 - Normally, the peer is in the IDLE state, i.e., in "equilibrium"
- › **A peer starts a KUDOS execution upon entering the BUSY state**
- › **A peer successfully completes a KUDOS execution by entering the IDLE state**
 - At which point the peer has the OSCORE Security Context CTX_NEW and has achieved key confirmation
- › **A peer can locally represent its current state using 2 bits**
 - (00) IDLE - The peer is not running KUDOS
 - (01) BUSY - The peer has not offered a nonce, but has received the nonce from the other peer
 - (10) BUSY - The peer has offered a nonce, but has not received the nonce from the other peer
 - (11) PENDING: The peer is running KUDOS, has offered its nonce, has received the nonce from the other peer, and is waiting for key confirmation

KUDOS State Machine

- › **Divergent message:** The sender peer is offering its own nonce in the message and waiting to receive the other peer's nonce.
- › **Convergent message:** The sender peer is offering its own nonce in the message, has received the other peer's nonce, and is going to wait for key confirmation



SCHC Compression of Extended OSCORE Option

› This was previously defined in draft-ietf-schc-8824-update

- Better to do in this document to have a better relation between the documents, which also affects normative/informative references

› Field Descriptors for SCHC

- the flag bits
- the Partial IV
- the kid context prepended by its size s
- the x byte
- the nonce
- the kid

› Instructions for handling of the ‘nonce’ and ‘x’ subfields

› IANA considerations (for "SCHC Compression of CoAP Fields" registry)

- Field: CoAP.option(9).x
- Field: CoAP.option(9).nonce

› Updated YANG data model including the extended format of the OSCORE

KUDOS Execution Example

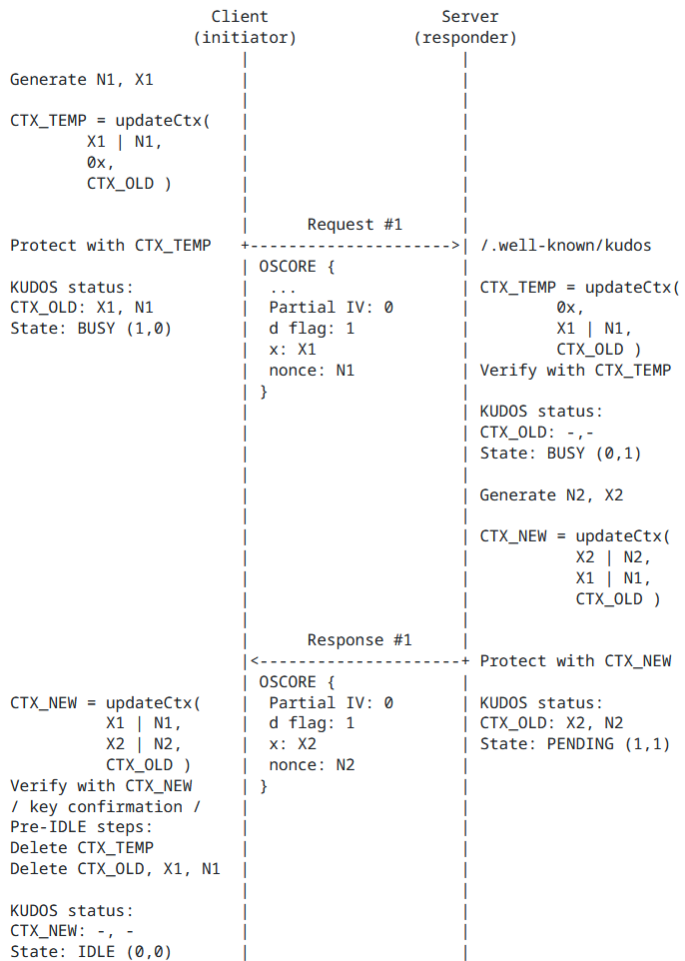
› Added further message flow example of a KUDOS execution

- Successful KUDOS Execution Initiated with a Request Message
- Successful KUDOS Execution Initiated with a Response Message
- Successful KUDOS Execution Initiated with a Request Message, with Non-capable Server that has Rebooted
- Successful KUDOS Execution Initiated with a Request Message, where the Client Executes KUDOS again after the first Execution
- Successful KUDOS Execution Initiated with a Request Message, where KUDOS Response #1 is Lost
- Successful KUDOS Execution Completed using two Request Messages
- Successful KUDOS Execution Initiated with a Request Message, Final Response is Lost

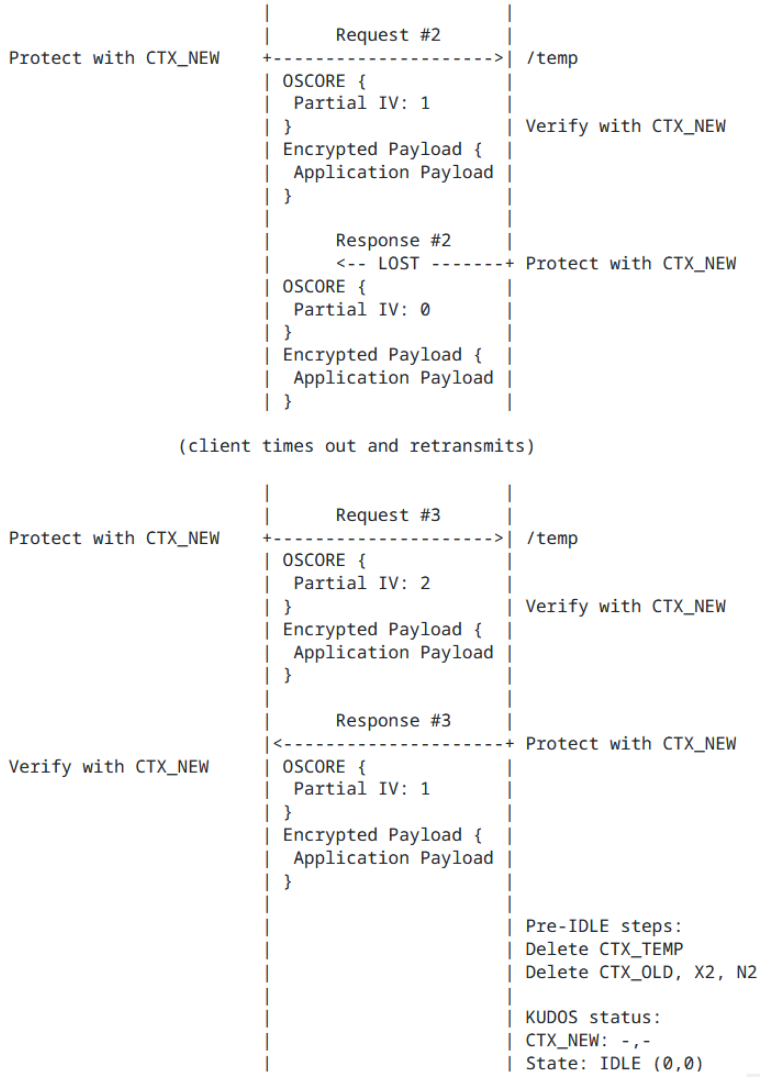
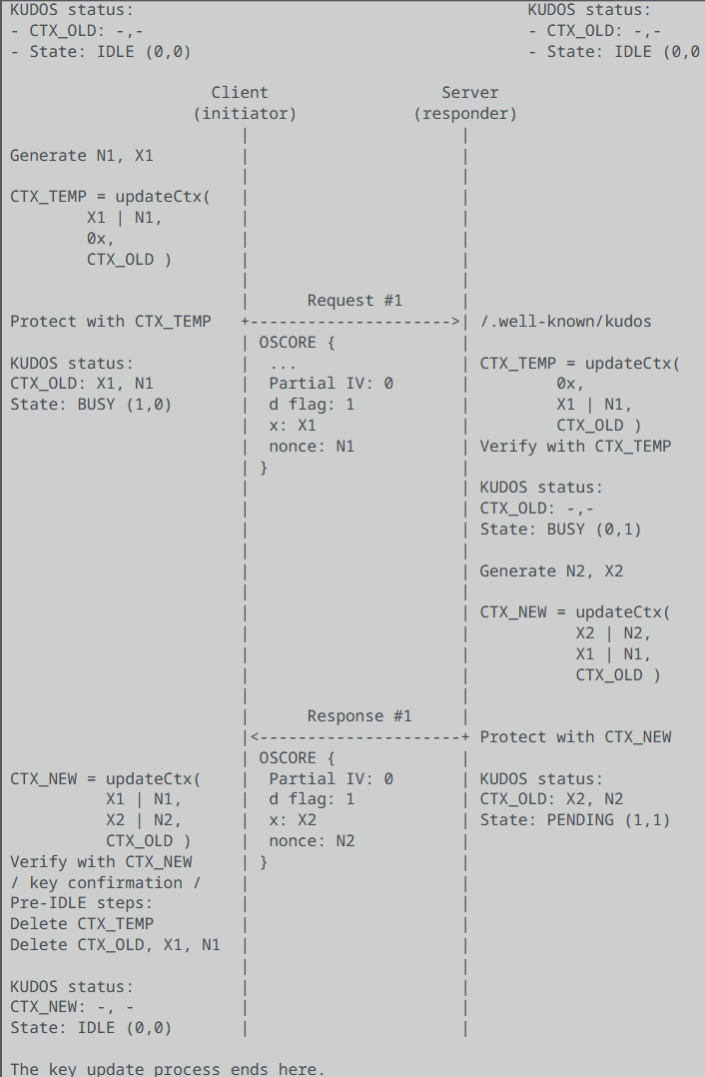
```

KUDOS status:
- CTX_OLD: -,-
- State: IDLE (0,0)
KUDOS status:
- CTX_OLD: -,-
- State: IDLE (0,0)

```



The key update process ends here.



Summary and Next Steps

› Design is in a stable state

- No major modifications planned

› KUDOS implementations

- Work is ongoing to update the implementation in Java [1] to be aligned with the latest design
 - › We also had interest from other IETF contributors regarding implementing the new design
- A later step is to also update the implementation in C for Contiki-NG

› Comments and reviews are welcome!

Thank you!

Comments/questions?

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

<https://github.com/core-wg/oscore-id-update>

<https://github.com/core-wg/oscore-key-limits>

Backup

Key Usage Limits & ID Update

› ID Update (draft-ietf-core-oscore-id-update)

- **Recap:** Method for updating peers' OSCORE Sender/Recipient IDs. Can be initiated by a client or by a server
- Submitted new version in January with main change to split overly long sections
- Next steps
 - Re-consider its design, to be in the same spirit of the new KUDOS design
 - Add more examples including failure cases

› Key Usage Limits (draft-ietf-core-oscore-key-limits)

- **Recap:** OSCORE-specific safe limits for Sender/Recipient Key usage
 - Safe number of encryptions using the Sender Key
 - Safe number of failed decryptions using the Recipient Key
- Next steps: Align to the document from CFRG as it develops [1]
 - There is ongoing discussion in the CFRG mailing list [2]

[1] <https://datatracker.ietf.org/doc/draft-irtf-cfrg-aead-limits/>

[2] <https://mailarchive.ietf.org/arch/msg/cfrg/CQ6MaMX1t96qxzxJK8cpTPVA46g/>