Key Update for OSCORE (KUDOS)

draft-ietf-core-oscore-key-update-11

Rikard Höglund, RISE
Marco Tiloca, RISE

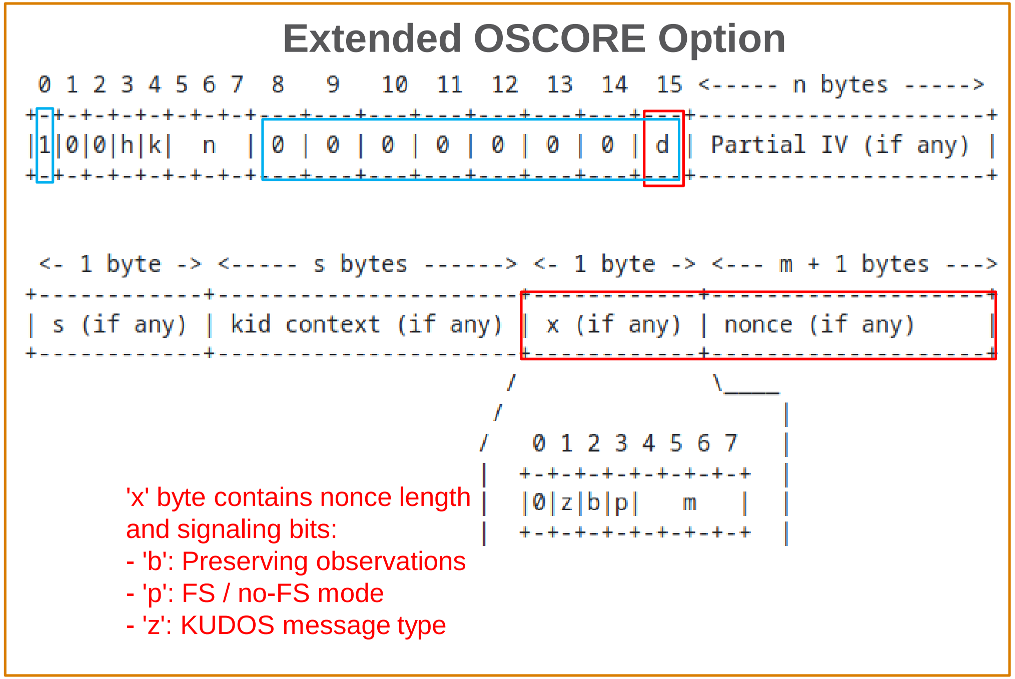
IETF CoRE WG meeting – IETF 123 – July 22nd, 2025

Recap

- 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 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

Rekeying Procedure

- **Key Update for OSCORE (KUDOS)**
  - Message exchange to share two nonces N1 and N2
    - Decoupled from request/response and client/server concepts
  - Nonces are placed in new fields in OSCORE CoAP option
  - *UpdateCtx()* function for deriving new OSCORE Security Context using the two nonces, two ‘x’ bytes and CTX_OLD
  - Two modes
    - FS mode providing forward secrecy
    - No-FS mode for very constrained devices
  - No change of OSCORE identifiers
  - Expected to complete in 1 round trip



Extended OSCORE Option

```
 0 1 2 3 4 5 6 7 8   9   10  11  12  13  14  15 <----- n bytes ----->
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+---------------------+
|1|0|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   |
                                |   +-+-+-+-+-+-+-+-+  |
                                |   |0|z|b|p|   m   |  |
                                |   +-+-+-+-+-+-+-+-+  |
```

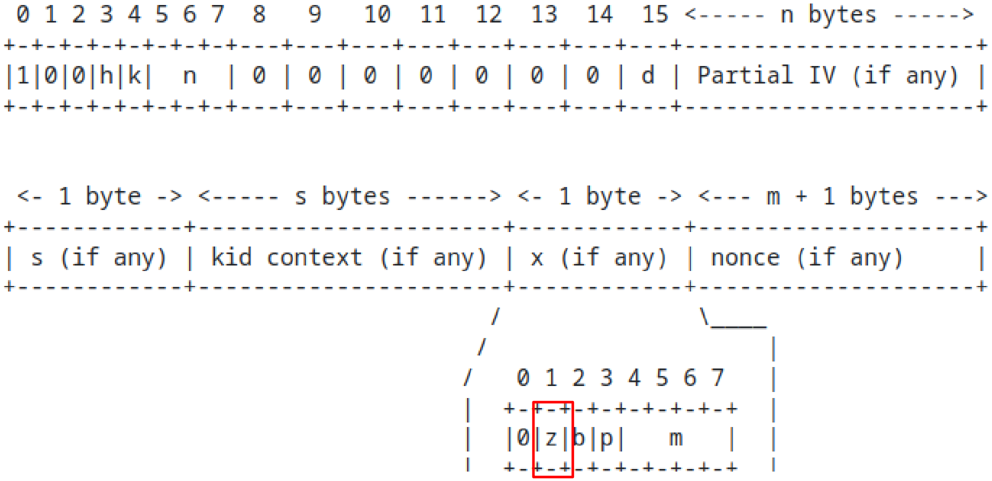
'x' byte contains nonce length and signaling bits:
- 'b': Preserving observations
- 'p': FS / no-FS mode
- 'z': KUDOS message type

KUDOS Message Types

- **Two types of KUDOS messages, distinguished by the 7th least significant bit 'z' in the 'x' byte**
  - Indicates if only one or both nonces have been exchanged

- **z = 0: “divergent message”**
- This message is protected with the temporary Security Context CTX_TEMP (was CTX_1)
- The sender peer is offering its own nonce in the message and waiting to receive the other peer’s nonce.

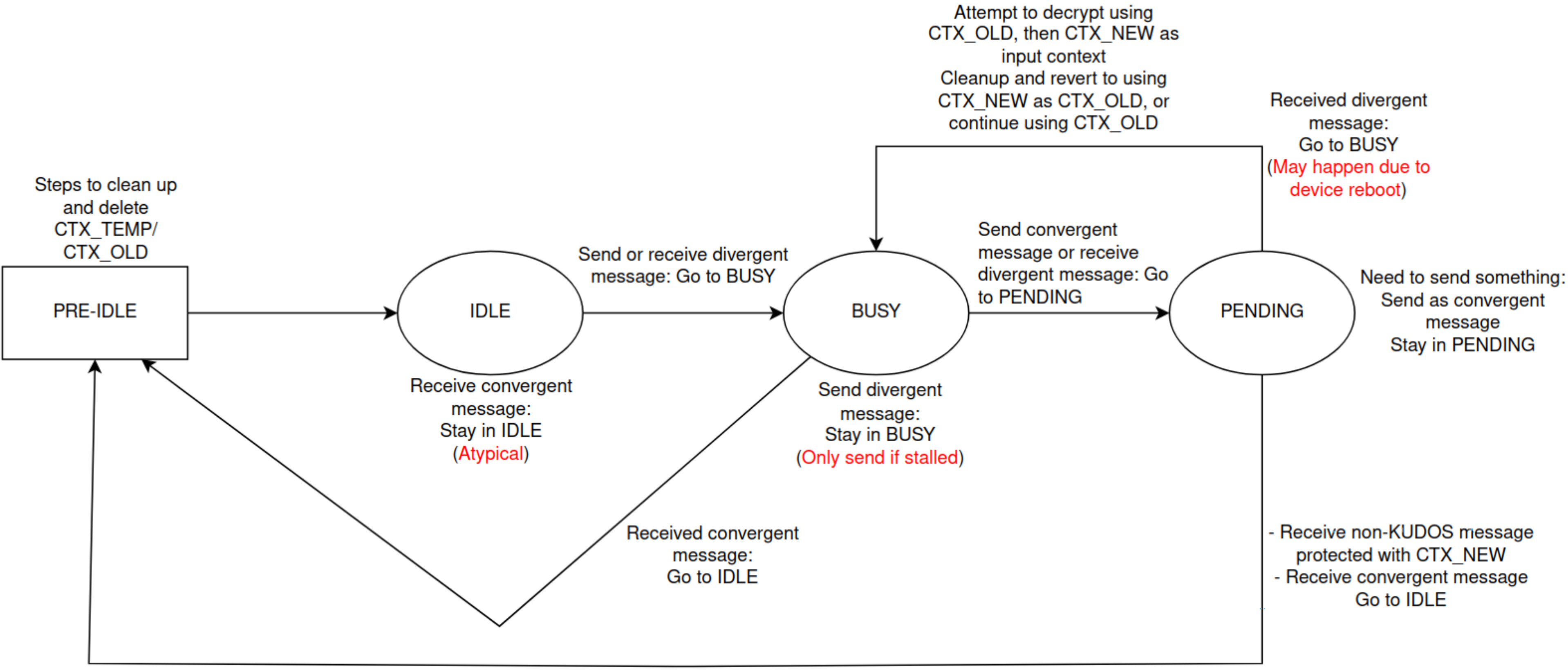
- **z = 1: “convergent message”**
- This message is protected with the final Security Context CTX_NEW.
- The sender peer is offering its own nonce in the message, has received the other peer’s nonce, and is going to wait for key confirmation



```
 0 1 2 3 4 5 6 7  8   9   10  11  12  13  14  15 <----- n bytes ----->
+-+-+-+-+-+-+-+-+---+---+---+---+---+---+---+---+---------------------+
|1|0|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   |
                                 |  +-+-+-+-+-+-+-+-+  |
                                 |  |0|z|b|p|   m   |  |
                                 |  +-+-+-+-+-+-+-+-+  |
```

Note: The z bit used to have another meaning

KUDOS State Machine



Attempt to decrypt using CTX_OLD, then CTX_NEW as input context
Cleanup and revert to using CTX_NEW as CTX_OLD, or continue using CTX_OLD

Received divergent message:
Go to BUSY
(May happen due to device reboot)

Steps to clean up and delete CTX_TEMP/ CTX_OLD

PRE-IDLE

IDLE

Send or receive divergent message: Go to BUSY

BUSY

Send convergent message or receive divergent message: Go to PENDING

PENDING

Need to send something:
Send as convergent message
Stay in PENDING

Receive convergent message:
Stay in IDLE
(Atypical)

Send divergent message:
Stay in BUSY
(Only send if stalled)

Received convergent message:
Go to IDLE

- Receive non-KUDOS message protected with CTX_NEW
- Receive convergent message
Go to IDLE

Main changes for v-11

- **Improved security considerations**
  - Extended discussion on the birthday paradox
  - Average collision for each nonce will happen after the generation of 2^32 (X, nonce) pairs throughout the update of a given OSCORE Security Context
  - **FS-mode**: Practically the expected number of (X, nonce) pairs generated is 1, to update CTX_OLD
  - **No-FS mode**: Since the context to be updated, CTX_BOOTSTRAP, does not change, collisions are more feasible here
  - Thus, we keep recommending a nonce size of 8 bytes

- **Updates to IANA considerations**
  - EDHOC External Authorization Data Registry (for the KUDOS EAD item)
    - *entries* → *entry*
    - Added 'name' field
  - OSCORE Flag Bits registry (for registering the OSCORE CoAP option extension bit)
    - *0 | Extension-1 Flag* → *0 (suggested) | Extension-1 Flag*
    - Set to 1 if the OSCORE Option specifies a second byte

Main changes for v-11

- **Extended section about updated protection of CoAP responses**
  - If the server is using a different Security Context for the response compared to what was used to verify the request (e.g., due to an occurred key update), then the server MUST include its Sender Sequence Number as Partial IV in the response and use it to build the AEAD nonce to protect the response.
  - This prevents the server from using the same AEAD (key, nonce) pair for two responses, protected with different OSCORE Security Contexts.
  - **Update**: Clarify that this should also be done when protecting Observe notifications

- **Combining usage of KUDOS with profiles of ACE**
  - Excluding KUDOS resources from access control
    - A KUDOS request that targets a non-KUDOS resource MUST trigger standard ACE-based access control checks
    - A KUDOS request that targets a KUDOS resource MUST NOT trigger ACE-based access control check
  - In some scenarios, an ACE Access Token may be bound to both CTX_OLD and CTX_NEW

- **Editorial improvements**

Main changes for v-11

- **Optimization upon Receiving a Divergent Message while in PENDING**
  - Currently this is one of the most complex parts of the state machine
  - **Can we avoid taking this path?**
- Solution: Do not transition to **BUSY** when receiving a divergent message that was already processed
  - Avoids repeated cryptographic operations and redundant transitions in the state machine

- **How to determine if a received divergent message was already processed?**
  - **MSG_A**: The just received divergent message
  - **MSG_B**: The previously received divergent message MSG_B that originally caused the latest transition to **PENDING** or **BUSY**
  - If MSG_A and MSG_B contain the same X byte and Nonce, then they are considered identical and no transition to **BUSY** is needed, the peer stays in **PENDING**
  -
- **How to understand what X byte and Nonce was used in MSG_B, if MSG_B is not saved?**
  - The Master Salt of the current OSCORE Security Context can be decomposed into its parts
  - As the Master Salt will be composed of the (X, Nonce) pairs from both peers

Summary and next steps

- **Add figure of state machine as appendix**
  - ASCII version of the figure in a previous slide

- **Add further message flow examples as appendixes**
  - E.g., different combinations of CoAP requests/responses and examples when messages are lost

- **KUDOS implementations**
  - Update the implementation in Java [1] to be aligned with the latest design
  - Update the implementation in C for Contiki-NG to be aligned with the latest design

- **Process a few minor issues captured in the Github repo**

- **Comments and reviews are welcome!**

[1] https://github.com/rikard-sics/californium/blob/oscore_cmd/cf-oscore/

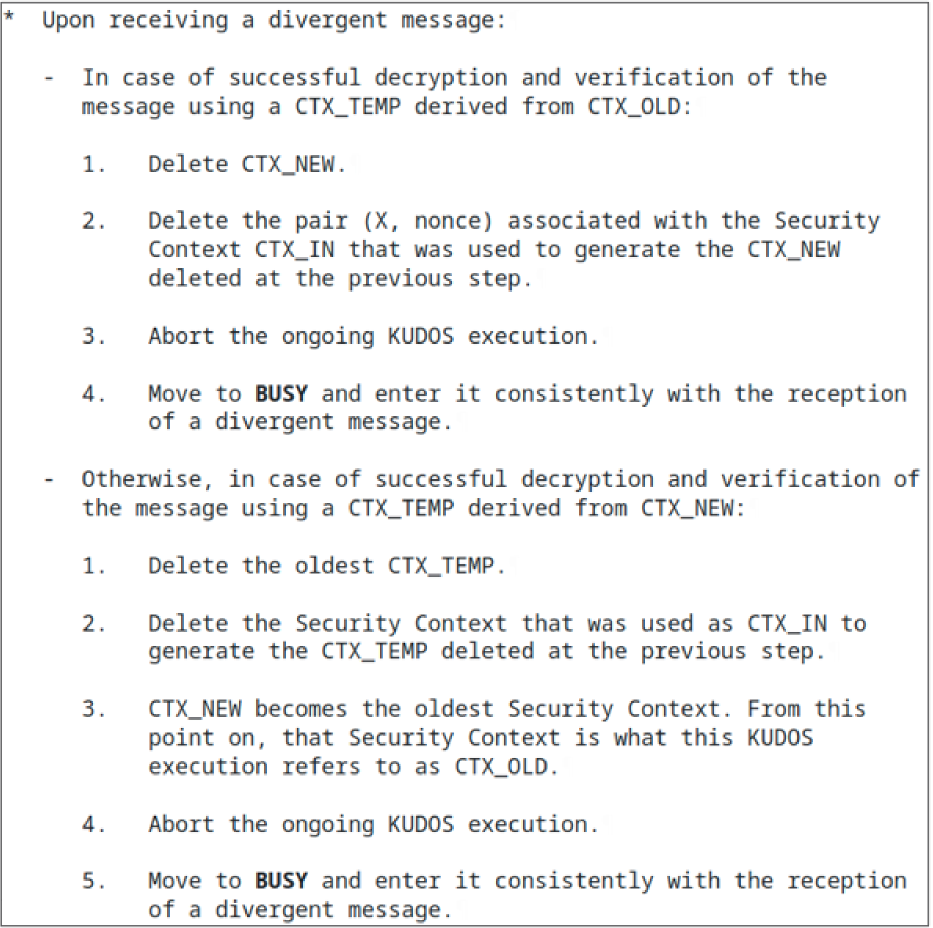
Thank you!

Comments/questions?

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

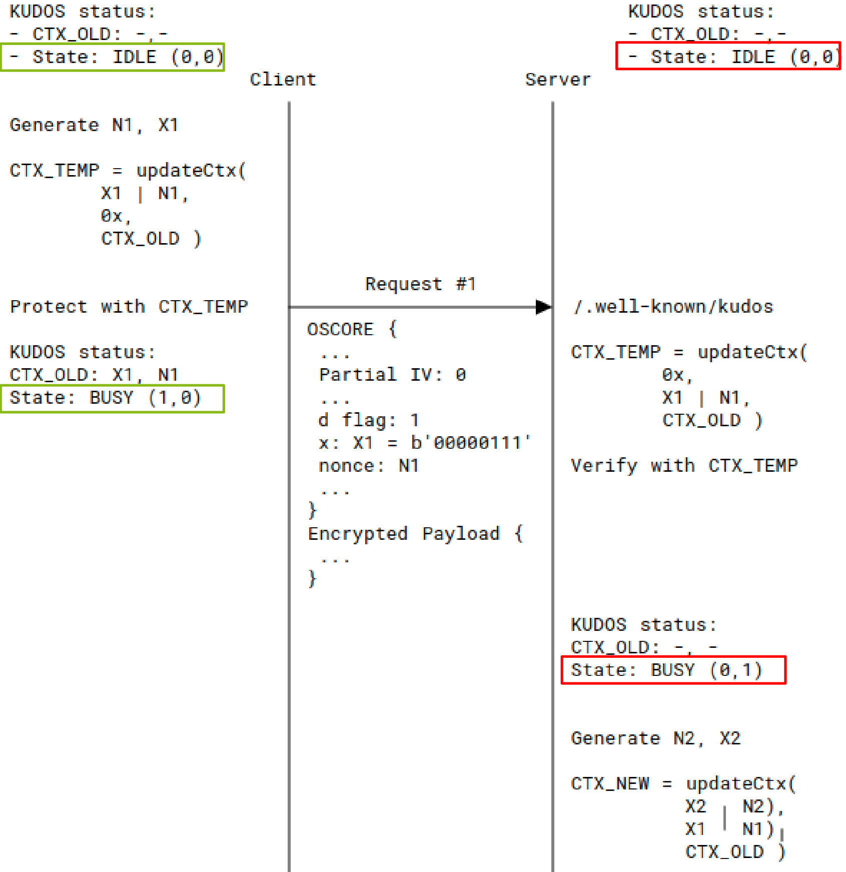
Backup

State Machine when in PENDING



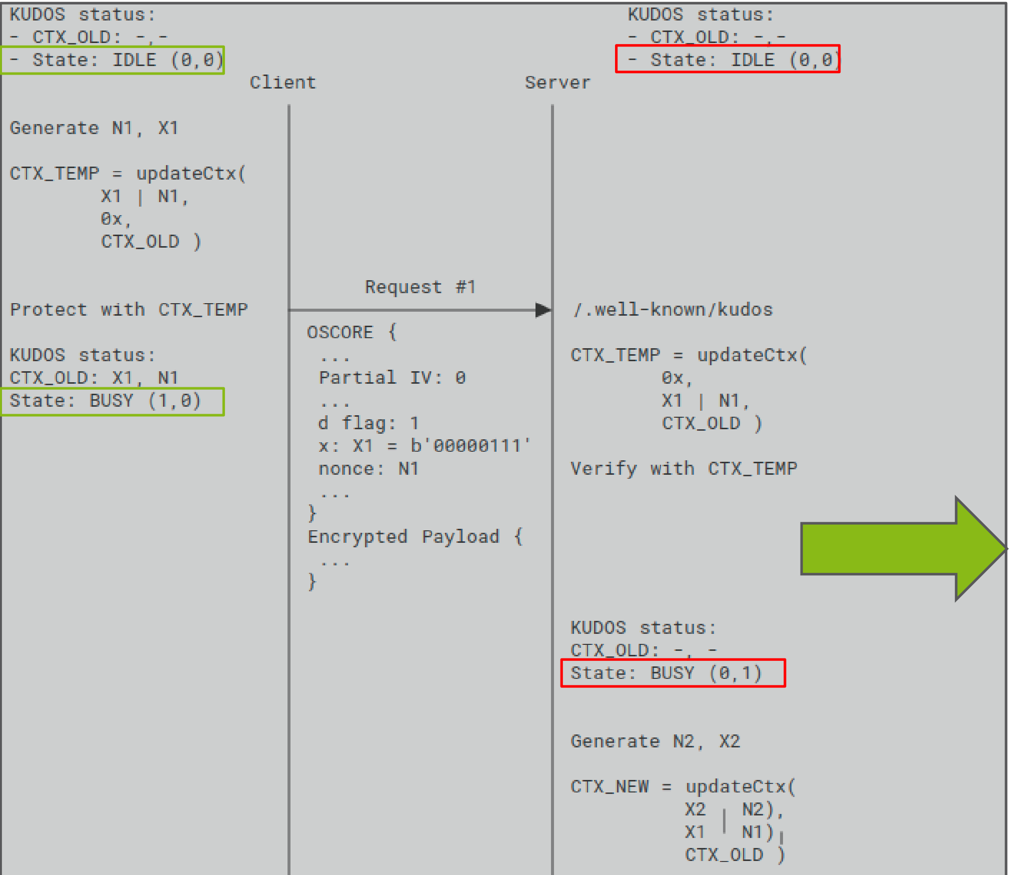
- Upon receiving a divergent message:
  - In case of successful decryption and verification of the message using a CTX_TEMP derived from CTX_OLD:
    1. Delete CTX_NEW.
    2. Delete the pair (X, nonce) associated with the Security Context CTX_IN that was used to generate the CTX_NEW deleted at the previous step.
    3. Abort the ongoing KUDOS execution.
    4. Move to **BUSY** and enter it consistently with the reception of a divergent message.
  - Otherwise, in case of successful decryption and verification of the message using a CTX_TEMP derived from CTX_NEW:
    1. Delete the oldest CTX_TEMP.
    2. Delete the Security Context that was used as CTX_IN to generate the CTX_TEMP deleted at the previous step.
    3. CTX_NEW becomes the oldest Security Context. From this point on, that Security Context is what this KUDOS execution refers to as CTX_OLD.
    4. Abort the ongoing KUDOS execution.
    5. Move to **BUSY** and enter it consistently with the reception of a divergent message.

Example Execution

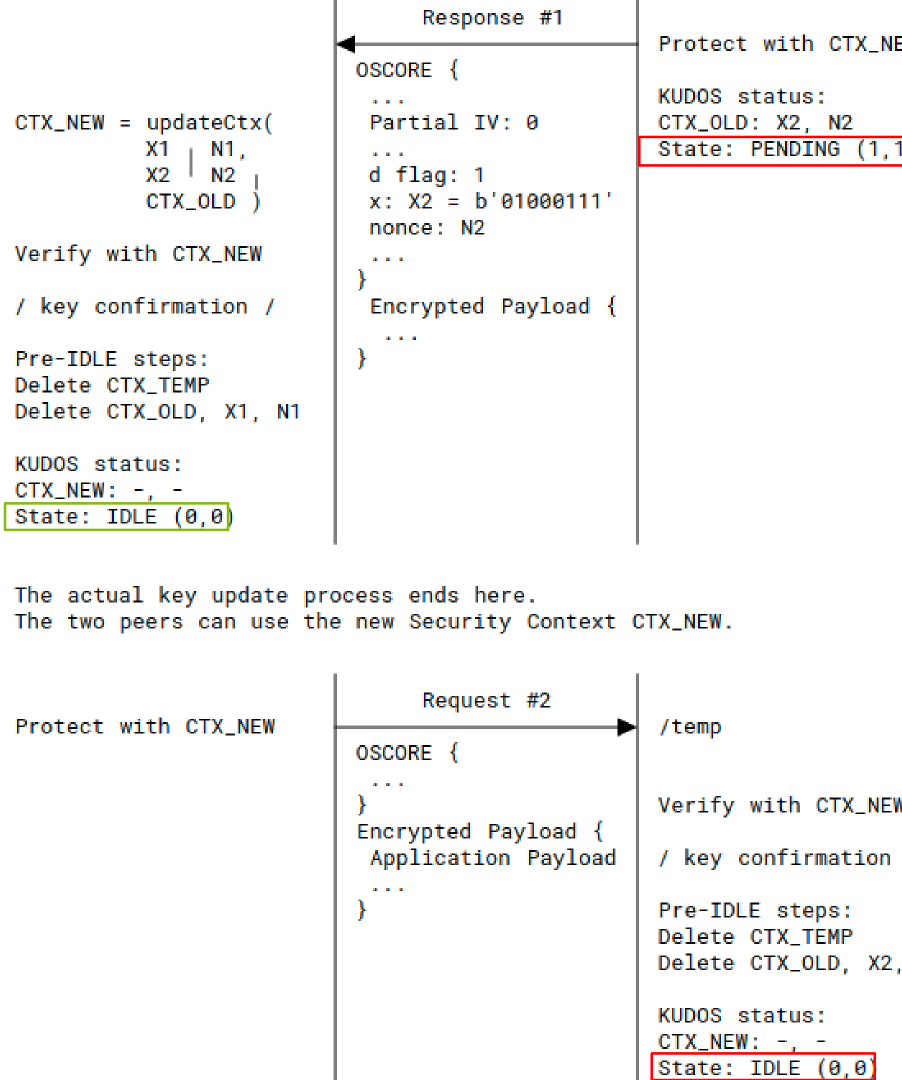


```
KUDOS status:                                          KUDOS status:
- CTX_OLD: -,-                                         - CTX_OLD: -,-
- State: IDLE (0,0)                                    - State: IDLE (0,0)
                      Client                  Server
                         |                       |
Generate N1, X1          |                       |
                         |                       |
CTX_TEMP = updateCtx(    |                       |
        X1 | N1,         |                       |
        0x,              |                       |
        CTX_OLD )        |                       |
                         |                       |
                         |      Request #1       |
Protect with CTX_TEMP    |---------------------->| /.well-known/kudos
                         | OSCORE {              |
KUDOS status:            |  ...                  | CTX_TEMP = updateCtx(
CTX_OLD: X1, N1          |  Partial IV: 0        |         0x,
State: BUSY (1,0)        |  ...                  |         X1 | N1,
                         |  d flag: 1            |         CTX_OLD )
                         |  x: X1 = b'00000111'  |
                         |  nonce: N1            | Verify with CTX_TEMP
                         |  ...                  |
                         | }                     |
                         | Encrypted Payload {   |
                         |  ...                  |
                         | }                     |
                         |                       |
                         |                       | KUDOS status:
                         |                       | CTX_OLD: -, -
                         |                       | State: BUSY (0,1)
                         |                       |
                         |                       |
                         |                       | Generate N2, X2
                         |                       |
                         |                       | CTX_NEW = updateCtx(
                         |                       |           X2 | N2),
                         |                       |           X1 | N1)|
                         |                       |           CTX_OLD )
```

Example Execution



```
KUDOS status:                               KUDOS status:
- CTX_OLD: -,-                              - CTX_OLD: -,-
- State: IDLE (0,0)                         - State: IDLE (0,0
                  Client              Server

Generate N1, X1

CTX_TEMP = updateCtx(
        X1 | N1,
        0x,
        CTX_OLD )

                        Request #1
Protect with CTX_TEMP   ------------>    /.well-known/kudos

KUDOS status:          OSCORE {          CTX_TEMP = updateCtx(
CTX_OLD: X1, N1         ...                      0x,
State: BUSY (1,0)       Partial IV: 0            X1 | N1,
                        ...                      CTX_OLD )
                        d flag: 1
                        x: X1 = b'00000111'
                        nonce: N1        Verify with CTX_TEMP
                        ...
                       }
                       Encrypted Payload {
                        ...
                       }

                                         KUDOS status:
                                         CTX_OLD: -, -
                                         State: BUSY (0,1)

                                         Generate N2, X2

                                         CTX_NEW = updateCtx(
                                                   X2 | N2),
                                                   X1 | N1)|
                                                   CTX_OLD )
```



```
                                 Response #1
                           <-----------------      Protect with CTX_NE
                           OSCORE {
CTX_NEW = updateCtx(        ...                    KUDOS status:
          X1 | N1,          Partial IV: 0          CTX_OLD: X2, N2
          X2 | N2 |         ...                    State: PENDING (1,1
          CTX_OLD )         d flag: 1
                            x: X2 = b'01000111'
Verify with CTX_NEW         nonce: N2
                            ...
/ key confirmation /       }
                            Encrypted Payload {
Pre-IDLE steps:              ...
Delete CTX_TEMP            }
Delete CTX_OLD, X1, N1

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

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

                                 Request #2
Protect with CTX_NEW       ----------------->      /temp
                           OSCORE {
                            ...
                           }                       Verify with CTX_NEW
                           Encrypted Payload {
                            Application Payload    / key confirmation
                            ...
                           }                       Pre-IDLE steps:
                                                   Delete CTX_TEMP
                                                   Delete CTX_OLD, X2,

                                                   KUDOS status:
                                                   CTX_NEW: -, -
                                                   State: IDLE (0,0
```

Key Usage Limits Overview

- **Working group document**
  - Content split out from *Key Update for OSCORE (KUDOS)* (draft-ietf-core-oscore-key-update)
  - Discussed during previous core interim on 2022-09-28 [1]
  - Also discussed and confirmed during IETF 115 [2]

- **Content of the draft: 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

- **Status**
  - Monitoring ongoing activities in CFRG

[1] https://datatracker.ietf.org/meeting/interim-2022-core-13/session/core
[2] https://datatracker.ietf.org/meeting/115/session/core

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