

Stand-in Key Identifier and Encrypted Partial IV in the Constrained Application Protocol (CoAP) OSCORE Option

draft-tiloca-core-oscore-piv-enc-02

Marco Tiloca, RISE
John Preuß Mattsson, Ericsson
Rikard Höglund, RISE
Göran Selander, Ericsson

IETF 125 meeting – Shenzhen – March 20th, 2026

Recap

› OSCORE (RFC 8613) protects CoAP messages end-to-end at the application layer

- Protected messages convey the CoAP OSCORE Option, which includes:
- A "Partial IV" field, i.e., the sequence number of the sender
- A "kid" field, i.e., the OSCORE Sender ID of the sender

› The OSCORE Option is not encrypted

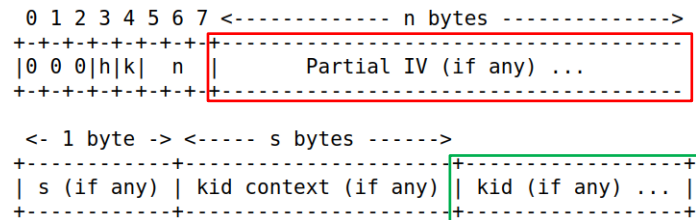
- Visible for enabling the message processing at the recipient

› "Partial IV" and "kid" are exposed in plaintext

- Possible to infer behavioral patterns of endpoints
- Possible to track endpoints across different network paths

› Encrypting "Partial IV" was mentioned at [1]

- Possible future update to consider
- The mechanics has to be simple and efficient
- Not much worth it if nothing is done for the "kid" values (e.g., encryption or update)



OSCORE Option Value

Contribution

- › **Document intended to update RFC 8613**
- › **Method to obfuscate fields of the OSCORE Option**
 - Encrypt the “Partial IV” field
 - Overwrite the “kid” field with a stand-in, ephemeral identifier ← **NEW**: Possible to avoid
 - Use keystreams computed from an Obfuscation Key (established with the Security Context)
- › **Features**
 - Simple and lightweight construct
 - No in-band signaling in OSCORE-protected messages
 - With minor adaptations, it also works for Group OSCORE [2]
- › **Pluggable “add-on”**
 - Its use is indicated as a property of the specific OSCORE Security Context
 - Establishing the Security Context includes setting whether to use this method or not
 - › Without explicit agreement and setting, this method is by default not used
 - If used, the OSCORE Option is accordingly obfuscated in all messages

Main updates since -01

- › **Largely addressing feedback from IETF 124 - Thanks!**
- › **Length of stand-in “kid” set to 3 bytes (was 2 bytes)**
 - This greatly reduces the chances of false positives when trying a Security Context
- › **Improved checks and error handling at the recipient endpoint**
 - Early stops in considering a candidate Security Context → Fewer steps and trials
 - Error response of the right type and not premature
- › **Make it possible to not obfuscate the “kid” field**
 - Specific input from Martine - Thanks!
 - Avoid a less efficient variant of what some use cases achieve differently, e.g., with header compression
- › **Extended security considerations**

New terms

› Ordinary Security Context

- A “vanilla” Security Context, as in OSCORE or Group OSCORE.
- When using this Security Context, this method is not used

Context	Case A	Case B	Case C
Obf. PIV	No	Yes	Yes
Obf. KID	No	No	Yes

› Obfuscating Security Context

- A Security Context including the Obfuscation Key
- When using this Security Context, the “Partial IV” field is always encrypted

› Incognito Security Context

- An Obfuscating Security Context, through which the “kid” field is encrypted too
- The way to locally “label” a Security Context of this type is implementation-specific

› The type of Security Context is agreed at establishment time

- All messages processed with that Security Context are obfuscated in the same way

When/why not obfuscating the KID?

- › **Feature suggested by Martine**
- › **Some use cases practically obfuscate the “kid” already, using other means**
 - E.g.: the stable, plain “kid” field triggers header compression based on static rules [3]
 - The compression residue to transmit does not include the “kid” field at all
 - Forcing an ephemeral stand-in “kid” prevents (full) compression → Unjustified performance penalties
- › **For such use cases**
 - Better to use an Obfuscating Security Context that is not an Incognito Security Context
 - › The “Partial IV” is encrypted
 - › The “kid” is elided by the header compression
- › **Other use cases should use an Incognito Security Context → full obfuscation**
- › **Also discussed in the security considerations in Section 6.6**

Sender side (1/2) – “Partial IV” Obf.

After OSCORE protection:

1. Compose **SAMPLE_1**

- First N bytes of the CoAP payload in the protected message
- $N = \min(\text{LENGTH}, 16)$, with $\text{LENGTH} \geq 9$ being the payload size in bytes

2. Compose **INPUT_1** (padded **SAMPLE_1**)

- If **SAMPLE_1** is less than 16 bytes in size, **INPUT_1** takes **SAMPLE_1** left-padded with zeroes to exactly 16 bytes
- If **SAMPLE_1** is 16 bytes in size, **INPUT_1** takes **SAMPLE_1**

3. Compute **PIV_KEYSTREAM**

- **PIV_KEYSTREAM** = AES-ECB(ENC_KEY, **INPUT_1**)
 - › AES in ECB mode [4]; ENC_KEY is the Obfuscation Key from the Common Context; **INPUT_1** comes from Step 2

4. Compute **ENC_PIV**, by XORing:

- The first Q bytes of **PIV_KEYSTREAM** from Step 3; and
- The plain content of the “Partial IV” field (Q bytes in size)
 - › Note: $Q \geq 3$; the Sender Sequence Number used must be at least 65536

5. Overwrite the “Partial IV” field in the OSCORE Option with **ENC_PIV** from Step 4

Same as in version -01

Sender side (2/2) – “kid” Obf.

6. If using an Incognito Security Context, compose **INPUT_2**

- Take **INPUT_1** from Step 2 (16-byte padded payload sample)
- **INPUT_2** takes **INPUT_1** with the last bit negated

Otherwise, terminate this algorithm and send the message

7. Compute **KID_KEYSTREAM**

- **KID_KEYSTREAM** = AES-ECB(ENC_KEY, **INPUT_2**)
 - › AES in ECB mode [4]; ENC_KEY is the Obfuscation Key from the Common Context; **INPUT_2** from Step 6

8. Compute the **3-byte STAND_IN_KID** value, by XORing:

- The first **3 bytes** of **KID_KEYSTREAM** from Step 7; and
- The first **3 bytes** of **LATEST_PIV**, which is either:
 - › ENC_PIV from Step 4, if this OSCORE Option includes “Partial IV”; or otherwise
 - › The value from “Partial IV” of the OSCORE Option in the corresponding request
 - This message is a response with “kid” but without “Partial IV”
 - Unlikely in OSCORE, but typical in Group OSCORE

9. Overwrite the “kid” field in the OSCORE Option with **STAND_IN_KID** from Step 8

- This might alter the length of the option value; if so “Option Length” must be updated accordingly

Recipient side

Who's the sender? What Security Context should I use? Is this OSCORE Option obfuscated at all?

› If storing Ordinary Security Contexts

- First assume no obfuscation → Retrieve a Security Context like in RFC 8613
- If one Ordinary Security Context is found, use it to **AEAD decrypt and verify** the message (*)
- If none is found or decryption fails, assume obfuscation and continue

› Check the stored Obfuscating Security Contexts that are not checked yet

- If some are **not Incognito Security Contexts**, focus on those and retrieve just like in RFC 8613
- Otherwise, inspect the Incognito Security Contexts sequentially

› When considering a Security Context CTX, revert the obfuscation performed by the sender

- **Only if CTX is an Incognito Security Context, locally recompute STAND_IN_KID**
 - › If different from what is in the “kid” field, try another Security Context
 - › If equal to what is in the “kid” field, CTX is (likely) the right Security Context, and “kid” ← Recipient ID
- Restore the plain “Partial IV” field
- Use CTX to **AEAD decrypt and verify** the message (*)
 - › If decryption fails, try with another Security Context

What else?

› Section 4 – Group OSCORE

- Updated message processing, consistent with the updates for OSCORE
- Same key differences from OSCORE
 - › The Security Context is retrieved as usual, using “kid context”
 - › The group as a whole works in one way, aligned with the type of Security Context
- The selection process is about finding the right Recipient Context within that Security Context
- If none is found, the endpoint can still dynamically create Recipient Contexts, assisted by the Group Manager
- New Section 4.5 on a special case: obfuscation not applicable to Deterministic Requests [5]

› New Section 6.5 – Security considerations on trial decryptions (possible in general, anyway)

- At most 1 early on, if storing both Ordinary Security Contexts and Obfuscating Security Contexts
- With small chances later on, if a matching stand-in “kid” is a false positive
- Regardless, failed decryption → update counters about key usage [6]

› New Section 6.4 – Security considerations on not obfuscating the “kid”

- Better for some use cases that rely on other means
- For other cases, this obfuscation method should be the one used

[5] <https://datatracker.ietf.org/doc/draft-ietf-core-cacheable-oscore/>

[6] <https://datatracker.ietf.org/doc/draft-ietf-core-oscore-key-limits/>

Next steps

- › **More content in Section 4 about Group OSCORE**
 - › Section 4.4 “External Signature Checker”
- › **Describe means for coordinating endpoints on using this method or not**
 - › For OSCORE: build on the placeholders in Section 5.1
 - › For Group OSCORE: build on the placeholders in Section 5.2
 - › Related requests for IANA registrations in Section 7
- › **Gather (more) implementation experience**
- › **Comments are welcome!**

Thank you!

Comments/questions?

<https://gitlab.com/crimson84/draft-tiloca-core-oscore-piv-enc>

Backup

Obfuscation Key

- › **One additional symmetric key, established with the OSCORE Security Context**

- Stored in the Common Context
- Same derivation construct used for the Sender/Recipient Keys

- › **Derivation based on HKDF**

- See Section 3.2.1 of RFC 8613

- › **Obfuscation Key = HKDF(salt, IKM, info, L)**

- ‘salt’ = Master Salt
- ‘IKM’ = Master Secret
- ‘info’
 - › ‘id’ = empty byte string
 - › ‘id_context’ = ID Context from the Common Context
 - › ‘alg_aead’ = encryption algorithm from the Common Context
 - › ‘type’ = “OBFFKey”
 - › ‘L’ = length in bytes of the algorithm in ‘alg_aead’
- ‘L’ = like ‘L’ within ‘info’

```
info = [  
    id : bstr,  
    id_context : bstr / nil,  
    alg_aead : int / tstr,  
    type : tstr,  
    L : uint,  
]
```