CoRE Working Group
Internet-Draft
Updates: 8613 (if approved)
Intended status: Standards Track
Expires: 3 September 2026

M. Tiloca
RISE AB
J. Preu Mattsson
Ericsson AB
R. Hglund
RISE AB
G. Selander
Ericsson AB

2 March 2026

Stand-in Key Identifier and Encrypted Partial IV in the Constrained Application Protocol (CoAP) OSCORE Option
draft-tiloca-core-oscore-piv-enc-02

Abstract

The security protocol Object Security for Constrained RESTful Environments (OSCORE) provides end-to-end protection of messages exchanged with the Constrained Application Protocol (CoAP). Messages protected with OSCORE include a CoAP OSCORE Option, where the "Partial IV" field specifies the sequence number value used by the message sender and the "kid" field specifies the identifier of the message sender. In order to reduce the information exposed on the wire that can be used for fingerprinting traffic and for tracking endpoints, this document defines a lightweight add-on method that obfuscates certain fields of the OSCORE Option, by encrypting the "Partial IV" field and overwriting the "kid" field with a stand-in identifier. Therefore, it updates RFC 8613. With minor adaptations, the defined method is applicable also to the security protocol Group Object Security for Constrained RESTful Environments (Group OSCORE) that protects group communication for CoAP.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Constrained RESTful Environments Working Group mailing list (core@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/core/.

Source for this draft and an issue tracker can be found at https://gitlab.com/crimson84/draft-tiloca-core-oscore-piv-enc.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Copyright Notice

Table of Contents

1. Introduction

The security protocol Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] provides end-to-end protection of messages exchanged with the Constrained Application Protocol (CoAP) [RFC7252]. OSCORE operates at the application layer by using CBOR Object Signing and Encryption (COSE) [RFC9052] and is independent of the specific transport used to exchange CoAP messages.

Messages protected with OSCORE include the CoAP OSCORE Option, which specifies information for the message recipient to correctly perform decryption and verification upon message reception. In particular, some of the fields that can be included in the OSCORE Option comprise:

* The "Partial IV" field, which specifies the sequence number value used by the sender endpoint when protecting an outgoing message. This field is always present in request messages, while it is typically absent in response messages, with a few exceptions mandating its presence.

* The "kid" field, which specifies the identifier of the sender endpoint protecting an outgoing message (i.e., the sender endpoint's OSCORE Sender ID). This field is always present in request messages, while it is typically absent in response messages.

Following a message protection with OSCORE, the OSCORE Option added to the message is not encrypted, since its content provides a recipient endpoint with information for processing the OSCORE-protected incoming message.

However, the information conveyed in plaintext by the "Partial IV" and "kid" fields could be used for fingerprinting traffic from OSCORE endpoints, e.g., by giving hints about the order of messages sent by an endpoint, from which behavioral patterns could be implied. Also, such information could be used to perform trivial tracking of OSCORE endpoints across different network paths, by correlating the values of those fields that are observed in those network paths (e.g., following a network path migration, possibly across different network segments).

In order to reduce the information exposed on the wire that can be used for fingerprinting traffic and for tracking endpoints, this document updates [RFC8613] and defines a lightweight add-on method that obfuscates certain fields of the OSCORE Option, by encrypting the "Partial IV" field and overwriting the "kid" field with a stand-in identifier.

This method does not require in-band signaling and its use does not arbitrarily change on a per-message basis.

Upon establishing an OSCORE Security Context, the communicating OSCORE endpoints already have an agreement about obfuscating the two fields of the OSCORE Option when that Security Context is used, for every OSCORE-protected message that includes the "Partial IV" field or the "kid" field. In the interest of specific use cases, such an agreement can be about always obfuscating both the "Partial IV" and "kid" fields, or instead about always obfuscating only the "Partial IV" field.

Like for the overall protection of messages with OSCORE, this method is agnostic of how exactly the OSCORE Security Context was established and of how the agreement on using this method was reached. Nevertheless, this document also defines means that endpoints can use to reach that agreement. Absent an explicit agreement, the "Partial IV" and "kid" fields in the OSCORE Option are not obfuscated and retain their original values, in order to preserve interoperability.

With minor adaptations to what is defined when OSCORE is used, the method defined in this document is applicable also to the security protocol Group Object Security for Constrained RESTful Environments (Group OSCORE) [I-D.ietf-core-oscore-groupcomm] that protects group communication for CoAP [I-D.ietf-core-groupcomm-bis]. In the

interest of such a case, this document also defines means to align the members of an OSCORE group about obfuscating the "Partial IV" and "kid" fields of protected messages exchanged within the group.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers are expected to be familiar with the terms and concepts related to CoAP [RFC7252], Concise Data Definition Language (CDDL) [RFC8610], Concise Binary Object Representation (CBOR) [RFC8949], COSE [RFC9052], OSCORE [RFC8613], and Group OSCORE [I-D.ietf-core-oscore-groupcomm].

This document refers also to the following terminology.

* Ordinary Security Context: an OSCORE Security Context [RFC8613] or a Group OSCORE Security Context [I-D.ietf-core-oscore-groupcomm] such that, when employed to process a message, the method defined in this document is not used to perform/reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option. An Ordinary Security Context does not include an Obfuscation Key in the Common Context (see Section 2).

* Obfuscating Security Context: an OSCORE Security Context or a Group OSCORE Security Context such that, when employed to process a message, the method defined in this document is used to perform/reverse the obfuscation of the "Partial IV" field in the OSCORE Option. An Obfuscating Security Context includes an Obfuscation Key in the Common Context (see Section 2).

* Incognito Security Context: an Obfuscating Security Context such that, when employed to process a message, the method defined in this document is also used to perform/reverse the obfuscation of the "kid" field in the OSCORE Option.

  Unless a Security Context is an Incognito Security Context, the "kid" field (if present) in the OSCORE Option is left unaltered.

  The particular way to mark an Obfuscating Security Context as an Incognito Security Context is implementation specific. For example, implementations can use an additional parameter in the Security Context.

2. Obfuscation Key

When obfuscation is enabled for the OSCORE Option, the (Group) OSCORE Security Context is extended with one additional parameter in the Common Context. The result is an Obfuscating Security Context.

The new parameter Obfuscation Key specifies the encryption key for deriving two separate keystreams, namely PIV_KEYSTREAM and KID_KEYSTREAM. On the sender side, PIV_KEYSTREAM and KID_KEYSTREAM are used to obfuscate the "Partial IV" and "kid" fields, respectively, when those are included in an outgoing message protected with (Group) OSCORE. On the recipient side, the same keystreams are used to reverse the obfuscation of the two fields.

The Obfuscation Key is derived as defined for the Sender/Recipient Keys in Section 3.2.1 of [RFC8613], with the following differences.

* The 'id' element of the 'info' array is the empty byte string.

* The 'type' element of the 'info' array is "OBFKey". The label is an ASCII string and does not include a trailing NUL byte.

* If the Security Context is used for Group OSCORE and the Group Encryption Algorithm in the Common Context is set (see Section 2.1.7 of [I-D.ietf-core-oscore-groupcomm]), then:

  - The 'alg_aead' element of the 'info' array specifies the Group Encryption Algorithm from the Common Context encoded as a CBOR integer or text string, consistently with the "Value" field in the entry of the "COSE Algorithms" Registry for this algorithm [COSE.Algorithms].

  - The L parameter of the HKDF and the 'L' element of the 'info' array are the length in bytes of the key for the Group Encryption Algorithm specified in the Common Context. While the obtained Obfuscation Key is never used with the Group Encryption Algorithm, its length was chosen to obtain a matching level of security.

* If the Security Context is used for Group OSCORE and the Group Encryption Algorithm in the Common Context is not set (see Section 2.1.7 of [I-D.ietf-core-oscore-groupcomm]), then:

- The ’alg_aead’ element of the ’info’ array specifies the AEAD Algorithm from the Common Context (see Section 2.1.1 of [I-D.ietf-core-oscore-groupcomm]) encoded as a CBOR integer or text string, consistently with the "Value" field in the entry of the "COSE Algorithms" Registry for this algorithm [COSE.Algorithms].

- The L parameter of the HKDF and the ’L’ element of the ’info’ array are the length in bytes of the key for the AEAD Algorithm specified in the Common Context. While the obtained Obfuscation Key is never used with the AEAD Algorithm, its length was chosen to obtain a matching level of security.

3. Processing in OSCORE

This section describes how the method defined in this document is specifically employed for messages protected with OSCORE [RFC8613].

3.1. Sender Side

When a sender endpoint uses a fresh Sender Sequence Number value from its own Sender Context to protect an outgoing message, that Sender Sequence Number value MUST be at least 65536. Consequently, the "Partial IV" field of the OSCORE Option will have a length of at least 3 bytes. As an exception, this requirement does not apply to the special case discussed in Section 3.3.1.

When composing a protected outgoing message MSG, the OSCORE Option MUST NOT include the "s" and "kid context" fields. As an exception, this requirement does not apply to the special case discussed in Section 3.3.3.

Once MSG is composed, if at least one among the "Partial IV" and "kid" fields is included in the OSCORE Option (see Section 6.1 of [RFC8613]), the sender endpoint performs the following steps.

1. Compose SAMPLE_1 as the first N bytes of the CoAP payload of MSG, where N = min(LENGTH, 16) and LENGTH denotes the length in bytes of the CoAP payload of MSG.

   Note that the CoAP payload is the ciphertext of the COSE object and LENGTH is guaranteed to be at least 9.

2. Compose the 16-byte INPUT_1 as follows:

   * If the length of SAMPLE_1 is less than 16 bytes, INPUT_1 is obtained by left-padding SAMPLE_1 with zeroes to exactly 16 bytes.

* If the length of SAMPLE_1 is 16 bytes, then INPUT_1 takes SAMPLE_1.

If the OSCORE Option of MSG includes the "Partial IV" field, move to Step 3. Otherwise, move to Step 6.

3. Compute the 16-byte PIV_KEYSTREAM as below:

PIV_KEYSTREAM = AES-ECB(ENC_KEY, INPUT_1)

where:

* AES-ECB is the AES algorithm in ECB mode [AES].

* ENC_KEY is the Obfuscation Key from the Common Context of the Security Context used to produce MSG (see Section 2). ENC_KEY is used as the encryption key for the AES-ECB encryption.

* INPUT_1 is the result of Step 2. It is used as the plaintext for the AES-ECB encryption.

4. Compute the ENC_PIV value, by XORing with each other:

* The PIV value encoded within the "Partial IV" field of the OSCORE Option of MSG; and

* The Q bytes from PIV_KEYSTREAM’s start, where Q is the length in bytes of the "Partial IV" field and PIV_KEYSTREAM is the result of Step 3.

For example, if the PIV value encoded within the "Partial IV" field of the OSCORE Option of MSG is 0x001122 (Q = 3 bytes) and PIV_KEYSTREAM is 0xffeeddccbbaa99887766554433221100 (16 bytes), then the bytes of PIV_KEYSTREAM to XOR with the PIV value are 0xffeedd.

5. In the "Partial IV" field of the OSCORE Option of MSG, replace its current PIV value with the ENC_PIV value computed at Step 4.

If the OSCORE Option of MSG includes the "kid" field, then move to Step 6. Otherwise, terminate this algorithm.

6. If the Security Context used to produce MSG is an Incognito Security Context, compose the 16-byte INPUT_2, by taking INPUT_1 from Step 2 and negating its last bit. Then, move to Step 7.

Otherwise, terminate this algorithm.

7. Compute the 16-byte KID_KEYSTREAM as below:

   KID_KEYSTREAM = AES-ECB(ENC_KEY, INPUT_2)

   where:

   * AES-ECB is the AES algorithm in ECB mode [AES].

   * ENC_KEY is the Obfuscation Key from the Common Context of the Security Context used to produce MSG (see Section 2). ENC_KEY is used as the encryption key for the AES-ECB encryption.

   * INPUT_2 is the result of Step 6. It is used as the plaintext for the AES-ECB encryption.

8. Compute the 3-byte STAND_IN_KID value, by XORing with each other:

   * The 3 bytes from LATEST_PIV's start, where LATEST_PIV is determined as follows.

     - If the OSCORE Option of MSG includes the "Partial IV" field, then LATEST_PIV is the ENC_PIV value computed at Step 4. Otherwise,

     - MSG is a response and LATEST_PIV is the value encoded within the "Partial IV" field of the OSCORE Option of the corresponding request as it was sent on the wire (i.e., in its obfuscated form).

   * The 3 bytes from KID_KEYSTREAM's start, where KID_KEYSTREAM is the result of Step 7.

9. In the "kid" field of the OSCORE Option of MSG, replace the current KID value with the STAND_IN_KID value computed at Step 8.

   Unless the original KID value had a length of 3 bytes, this step alters the length of the OSCORE Option value. In such a case, the sender endpoint MUST update the "Option Length" field of the OSCORE Option accordingly (see Section 3.1 of [RFC7252]).

Finally, the sender endpoint transmits MSG as expected.

3.2. Recipient Side

Upon receiving a protected incoming message MSG, the recipient endpoint has to determine the OSCORE Security Context to use for decrypting and verifying MSG (see Section 3.2.1).

If a Security Context CTX is found and CTX is an Obfuscating Security Context, then the recipient endpoint uses CTX to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of MSG (see Section 3.2.2). After that, the recipient endpoint uses CTX to decrypt and verify MSG.

3.2.1. Retrieving the Security Context to Use

If the recipient endpoint is a client, hence MSG is a response, the Security Context CTX to use is the same one that was used to protect the request to which MSG replies. That is, CTX is retrieved by leveraging the CoAP Token value that is specified in the "Token" field of MSG and was specified in the "Token" field of the corresponding request.

Then, the following two cases are possible:

* CTX is an Ordinary Security Context. In this case, the client uses CTX to decrypt and verify MSG, as defined in Section 8.4 of [RFC8613].

* CTX is an Obfuscating Security Context. In this case, the client performs the steps defined in Section 3.2.2, in order to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of MSG by using CTX. Building on the result, the client uses CTX to decrypt and verify MSG, as defined in Section 8.4 of [RFC8613].

If the recipient endpoint is a server, hence MSG is a request, the Security Context CTX to use is determined as follows.

1. If the server stores at least one Ordinary Security Context, then move to Step 2. Otherwise, move to Step 3.

2. The server assumes that the "Partial IV" and "kid" fields in the OSCORE Option of MSG were not obfuscated. Then, the server attempts to retrieve a Security Context as defined in Section 8.2 of [RFC8613].

   If this results in retrieving a Security Context CTX and CTX is an Ordinary Security Context, then the server uses CTX to decrypt and verify MSG, as defined in Section 8.2 of [RFC8613]. In case of successful decryption and verification, this algorithm terminates and the server continues processing MSG as expected.

   Instead, if no such CTX is found or if MSG is not processed successfully, the following applies:

* If the server stores at least one Obfuscating Security Context, the server MUST NOT presently reply with any of the optional 4.01 (Unauthorized) or 4.00 (Bad Request) error responses defined in Sections 7.4 and 8.2 of [RFC8613]. Then, this algorithm moves to Step 4.

* Otherwise, the server performs the same error handling as defined in Sections 7.4 and 8.2 of [RFC8613]. Then, this algorithm terminates.

3. If the server stores at least one Obfuscating Security Context, then move to Step 4.

   Otherwise, the server performs the same error handling as defined at Step 2 of Section 8.4 of [RFC8613] for the case where a Security Context is not found. Then, this algorithm terminates.

4. Compose SAMPLE_1 by means of the same method at Step 1 of Section 3.1, with reference to the present incoming message MSG.

5. Compose the 16-byte INPUT_1 by means of the same method at Step 2 of Section 3.1, using as SAMPLE_1 the result of Step 4 of the present algorithm.

6. Compose the 16-byte INPUT_2, by taking INPUT_1 from Step 5 and negating its last bit.

7. Select a Security Context CTX, such that CTX is an Obfuscating Security Context and has not been tested yet during this execution of the present algorithm.

   In case the recipient endpoint does not store any Incognito Security Contexts, the selection process can effectively be the one used in Section 8.2 of [RFC8613].

   If no such CTX is found, then move to Step 12. Otherwise, the selected CTX is marked as tested and the following applies:

   * If CTX is an Incognito Security Context, check the length of the "kid" field of the OSCORE Option of MSG.

     If the length is 3 bytes, move to Step 8. Otherwise, perform Step 7 again.

* If CTX is not an Incognito Security Context, check whether the value encoded in the "kid" field of the OSCORE Option of MSG is equal to the Recipient ID specified in the Recipient Context of CTX.

  In case the two values are equal, CTX is the Security Context to use for decrypting and verifying MSG, and this algorithm moves to Step 11. Otherwise, perform Step 7 again.

8. Compute the 16-byte KID_KEYSTREAM by means of the same method at Step 7 of Section 3.1. In particular:

   * ENC_KEY is the Obfuscation Key from the Common Context of the latest CTX selected at Step 7 of the present algorithm.

   * INPUT_2 is the result of Step 6 of the present algorithm.

9. Compute the 3-byte STAND_IN_KID value, by XORing with each other:

   * The 3 bytes from ENC_PIV’s start, where ENC_PIV is the value encoded within the "Partial IV" field of the OSCORE Option of MSG.

   * The 3 bytes from KID_KEYSTREAM’s start, where KID_KEYSTREAM is the result of Step 8.

10. If the STAND_IN_KID value computed at Step 9 is not equal to the value encoded in the "kid" field of the OSCORE Option of MSG, then move to Step 7.

    Otherwise, the latest CTX selected at Step 7 is the Security Context to use for decrypting and verifying MSG, and this algorithm moves to Step 11.

11. Run the algorithm in Section 3.2.2, in order to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of MSG, by using the latest CTX selected at Step 7.

    Building on the result, the server uses CTX to decrypt and verify MSG, as defined in Section 8.2 of [RFC8613]. In case of successful decryption and verification, this algorithm terminates and the server continues processing MSG as expected.

    Otherwise, if MSG is not processed successfully, the following applies:

* The server MUST NOT presently reply with any of the optional 4.01 (Unauthorized) or 4.00 (Bad Request) error responses defined in Sections 7.4 and 8.2 of [RFC8613].

* The OSCORE Option of MSG is restored to be as it was before running the algorithm in Section 3.2.2.

* This algorithm moves to Step 7.

12. The following applies:

* If, since Step 4 of this execution of the present algorithm, the server has performed and failed at least one decryption and verification of MSG (see Step 6 of Section 8.2 of [RFC8613]), the server performs the same error handling defined at Step 6 of Section 8.2 of [RFC8613] for the case where decryption has failed. Then, this algorithm terminates. Otherwise,

* If, since Step 4 of this execution of the present algorithm, the server has performed and failed at least one replay check on MSG (see Step 3 of Section 8.2 of [RFC8613]), the server performs the same error handling defined in Section 7.4 of [RFC8613] for the case where a replay has been detected. Then, this algorithm terminates. Otherwise,

* The server performs the same error handling defined at Step 2 of Section 8.2 of [RFC8613] for the case where a Security Context is not found.

3.2.2. Reversing the Field Obfuscation

Given an Obfuscating Security Context CTX that was retrieved according to what is specified in Section 3.2.1, the recipient endpoint performs the following steps, in order to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of the protected incoming message MSG.

1. If the OSCORE Option of MSG includes the "kid" field and CTX is an Incognito Security Context, then move to Step 2. Otherwise, move to Step 3.

2. In the "kid" field of the OSCORE Option of MSG, replace the current STAND_IN_KID value with the Recipient ID specified in the Recipient Context of CTX.

Unless the Recipient ID has a length of 3 bytes, this step alters the length of the OSCORE Option value. In such a case, the recipient endpoint MUST update the "Option Length" field of the OSCORE Option accordingly (see Section 3.1 of [RFC7252]).

3. If the OSCORE Option of MSG includes the "Partial IV" field, move to Step 4. Otherwise, terminate this algorithm.

4. Compute the 16-byte PIV_KEYSTREAM by means of the same method defined at Step 3 of Section 3.1. In particular:

   * ENC_KEY is the Obfuscation Key from the Common Context of the Security Context CTX that is used during this execution of the present algorithm.

   * INPUT_1 is composed by means of the same method defined at Step 5 of Section 3.2.1. Note that, if the recipient endpoint is a server, then INPUT_1 was already computed when actually performing Step 5 of Section 3.2.1.

5. Compute the PIV value, by XORing with each other:

   * The ENC_PIV value encoded within the "Partial IV" field of the OSCORE Option of MSG; and

   * The Q bytes from PIV_KEYSTREAM’s start, where Q is the length in bytes of the "Partial IV" field and PIV_KEYSTREAM is the result of Step 4.

6. In the "Partial IV" field of the OSCORE Option of MSG, replace its current ENC_PIV value with the PIV value computed at Step 5.

3.3. Special Cases

This section discusses some special cases where the use of the method defined in this document deviates from what is specified in Section 3.1 and Section 3.2.

3.3.1. EDHOC + OSCORE Request

Two endpoints can run the authenticated key agreement Ephemeral Diffie-Hellman over COSE (EDHOC) [RFC9528] in order to establish an OSCORE Security Context (see Appendix A.1 of [RFC9528]). In particular, EDHOC messages can be transported over CoAP (see Appendix A.2 of [RFC9528]).

When doing so, if the endpoint that sends the first EDHOC message acts as a CoAP client, it is possible to use the optimized workflow defined in Section 3 of [RFC9668]. In such a case, the first OSCORE-protected CoAP request sent by the client additionally embeds the final EDHOC message, and it is protected with the OSCORE Security Context established from the present and still ongoing EDHOC session.

Upon receiving such an EDHOC + OSCORE request, the server first extracts and processes the EDHOC message embedded therein, completes the EDHOC session, establishes the OSCORE Security Context, and finally uses the latter to decrypt and verify the OSCORE-protected CoAP request.

When using this optimized workflow, the method defined in this document cannot be used to obfuscate the "Partial IV" and "kid" fields in the OSCORE Option of the EDHOC + OSCORE request.

Therefore, the following applies for the OSCORE Option of that specific request:

* The "Partial IV" field conveys the Sender Sequence Number of the client in plaintext. As an exception to the requirement defined in Section 3.1, the "Partial IV" field can have a length smaller than 3 bytes. In fact, it is expected to have a length of 1 byte and to encode the Sender Sequence Number 0.

* The "kid" field conveys the actual OSCORE Sender ID of the client, which the server offered earlier in the EDHOC session as its own EDHOC connection identifier C_R. Upon receiving the EDHOC + OSCORE request, the server needs to retrieve such an identifier as-is from the request, in order to correctly retrieve the EDHOC session and complete it, before establishing the OSCORE Security Context shared with the client.

  That is, even if the OSCORE Security Context under establishment is an Incognito Security Context, the "kid" field in the OSCORE Option of the EDHOC + OSCORE request is not obfuscated.

  Note that, if EDHOC is instead run as per the original workflow (see Appendix A.2.1 of [RFC9528], the OSCORE Sender ID of the client is anyway exposed at least once, since C_R is prepended to EDHOC message_3 within the CoAP payload of a request sent by the client.

3.3.2. KUDOS

Two endpoints can use Key Update for OSCORE (KUDOS) [I-D.ietf-core-oscore-key-update], a lightweight procedure for updating their OSCORE keying material by establishing a new OSCORE Security Context.

Given the Security Context CTX_OLD to be replaced, there are two possible types of KUDOS messages that are exchanged during a KUDOS execution:

* A divergent KUDOS message is protected with a temporary OSCORE Security Context CTX_TEMP, which is derived from CTX_OLD.

* A convergent KUDOS message is protected with the OSCORE Security Context CTX_NEW, which is derived from CTX_OLD and is intended to replace CTX_OLD.

When using the method defined in this document to obfuscate the "Partial IV" and "kid" fields in the OSCORE Option of a KUDOS message, the following applies.

* The Obfuscation Key to use MUST be the one specified in the Common Context of the Security Context CTX_OLD, from which the Security Context CTX_TEMP (CTX_NEW) is derived for protecting a divergent (convergent) KUDOS message.

  That is, with reference to a divergent (convergent) KUDOS message, the Obfuscation Key to use is not the one specified in the Common Context of the Security Context CTX_TEMP (CTX_NEW) that is used to protect the message.

3.3.3. 6TiSCH

The Constrained Join Protocol (CoJP) defined in [RFC9031] specifies a "secure join" solution for a new device, called a "pledge", to securely join a 6TiSCH network where communications are protected with OSCORE. The Join process is assisted by a central entity called Join Registrar/Coordinator (JRC).

In particular, as defined in Section 7.3 of [RFC9031], the following holds for a given pledge and the JRC using OSCORE:

* The OSCORE ID Context in the shared OSCORE Security Context is set to the pledge identifier.

* The OSCORE Sender ID of the pledge is set to the empty byte string.

* The OSCORE Sender ID of the JRC is set to the byte string 0x4a5243 ("JRC" in ASCII).

In the Join Request that the pledge sends to the JRC, the OSCORE Option includes the "s" and "kid context" fields, with the latter encoding the OSCORE ID Context.

When using the method defined in this document to obfuscate the "Partial IV" and "kid" fields in the OSCORE Option of CoJP messages, the following applies:

* If the "s" and "kid context" fields are present in the OSCORE Option of an outgoing CoJP message, the sender endpoint MUST remove the "kid context" field and MUST update the "s" field to encode the value 0.

  This step alters the length of the OSCORE Option value. Therefore, the sender endpoint MUST update the "Option Length" field of the OSCORE Option accordingly (see Section 3.1 of [RFC7252]).

* If the OSCORE Option of an incoming CoJP message MSG does not include the "kid context" field and includes the "s" field encoding the value 0, the recipient endpoint performs the following steps in addition to those compiled in Section 3.2.2:

  - Add the "kid context" field to the OSCORE Option of MSG.

  - In the "kid context" field of the OSCORE Option, set as value the ID Context that is specified in the OSCORE Security Context CTX to be used for decrypting and verifying MSG.

  - In the "s" field of the OSCORE Option, set as value the length in bytes of the "kid context" field.

  Unless the ID Context has a length of 0 bytes, this step alters the length of the OSCORE Option value. In such a case, the recipient endpoint MUST update the "Option Length" field of the OSCORE Option accordingly (see Section 3.1 of [RFC7252]).

4. Processing in Group OSCORE

This section describes how the method defined in this document is specifically employed for messages protected with Group OSCORE [I-D.ietf-core-oscore-groupcomm].

In particular, the following presents the differences that apply with respect to the case where OSCORE is used (see Section 3).

4.1. Keying Material

When the Group OSCORE Security Context is an Obfuscating Security Context, it is further extended with one additional parameter Obfuscation Sender Key in the Sender Context (see Section 4.1.1) and with one additional parameter Obfuscation Recipient Key in each Recipient Context (see Section 4.1.2).

4.1.1. Obfuscation Sender Key

Within the Sender Context, the new parameter Obfuscation Sender Key specifies the encryption key for deriving the keystreams PIV_KEYSTREAM and KID_KEYSTREAM, when those are used to obfuscate the "Partial IV" and "kid" fields in the OSCORE Option of an outgoing message.

The Obfuscation Sender Key is derived as the output OKM of an HKDF-Expand step [RFC5869], i.e., OKM = HKDF-Expand(PRK, info, L), where:

* The HKDF used is the HKDF Algorithm specified in the Common Context.

* PRK is the Obfuscation Key from the Common Context.

* info is the Sender ID specified in the Sender Context.

* L is the length in bytes of the Obfuscation Key from the Common Context.

4.1.2. Obfuscation Recipient Key

Within a given Recipient Context, the new parameter Obfuscation Recipient Key specifies the encryption key for deriving the keystreams PIV_KEYSTREAM and KID_KEYSTREAM, when those are used to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of an incoming message.

The Obfuscation Recipient Key is derived as the output OKM of an HKDF-Expand step [RFC5869], i.e., OKM = HKDF-Expand(PRK, info, L), where:

* The HKDF used is the HKDF Algorithm specified in the Common Context.

* PRK is the Obfuscation Key from the Common Context.

* info is the Recipient ID specified in the Recipient Context.

* L is the length in bytes of the Obfuscation Key from the Common Context.

4.2. Sender Side

When composing a protected outgoing message MSG, the OSCORE Option includes the "s" and "kid context" fields according to what is specified for Group OSCORE in [I-D.ietf-core-oscore-groupcomm]. That is, the OSCORE Option always includes those fields in a request and may include those fields in a response.

Once MSG is composed, if at least one among the "Partial IV" and "kid" fields is included in the OSCORE Option (see Section 6.1 of [RFC8613]), the sender endpoint performs the same steps of Section 3.1, with the following differences:

* At Step 1, LENGTH is guaranteed to be at least 9. In particular:

  - If MSG is protected with the group mode of Group OSCORE (see Section 7 of [I-D.ietf-core-oscore-groupcomm]), then the CoAP payload is the ciphertext of the COSE object concatenated with the encrypted countersignature.

  - If MSG is protected with the pairwise mode of Group OSCORE (see Section 8 of [I-D.ietf-core-oscore-groupcomm]), then the CoAP payload is the ciphertext of the COSE object.

* At Step 3, ENC_KEY is the Obfuscation Sender Key from the Sender Context.

* At Step 7, ENC_KEY is the Obfuscation Sender Key from the Sender Context.

Section 4.5 defines an exceptional case where a value smaller than 65536 is used as Sender Sequence Number and the "kid" field of the OSCORE Option is not overwritten by a STAND_IN_KID value, even if the Group OSCORE Security Context is an Incognito Security Context.

4.3. Recipient Side

Upon receiving a protected incoming message MSG, the recipient endpoint determines the Group OSCORE Security Context CTX to use according to what is specified in [I-D.ietf-core-oscore-groupcomm], i.e.:

* If MSG is a request, CTX is retrieved by leveraging the Group Identifier value (Gid) of the group, which is encoded within the "kid context" field in the OSCORE Option of MSG.

* If MSG is a response, CTX is the same Group OSCORE Security Context that was used to protect the request to which MSG replies. That is, CTX is retrieved by leveraging the CoAP Token value that is specified in the "Token" field of MSG and was specified in the "Token" field of the corresponding request. The possible presence of the "kid context" field in the OSCORE Option can further aid the client, e.g., in case the group has been rekeyed and its Gid has changed.

If the retrieved Group OSCORE Security Context CTX is an Obfuscating Security Context, then the method defined in this document is used to obfuscate the "Partial IV" field in the OSCORE Option of every protected message exchanged within the group. Furthermore, if CTX is specifically an Incognito Security Context, then the method defined in this document is used to obfuscate also the "kid" field in the OSCORE Option of every protected message exchanged within the group.

Consequently, within CTX, the recipient endpoint has to determine the specific Recipient Context REC_CTX to use for decrypting and verifying MSG (see Section 4.3.1).

Then, the recipient endpoint uses REC_CTX to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of MSG (see Section 4.3.2). After that, the recipient endpoint uses REC_CTX to decrypt and verify MSG.

4.3.1. Retrieving the Recipient Context to Use

Given the retrieved Group OSCORE Security Context CTX, the following describes how the recipient endpoint retrieves from CTX the specific Recipient Context REC_CTX to use.

If the recipient endpoint is a client, hence MSG is a response, the client is able to simply retrieve the Recipient Context REC_CTX to use, in case both the following conditions apply:

* The request corresponding to MSG was protected with the pairwise mode of Group OSCORE; and

* The "kid" field is not included in the OSCORE Option of MSG.

In such a case, REC_CTX is the Recipient Context associated with the other endpoint for which the request corresponding to MSG was protected. That is, this client protected such request by using its Pairwise Sender Key associated with that other endpoint. Consequently, the client performs the steps defined in Section 4.3.2, in order to reverse the obfuscation of the "Partial IV" field (if present) in the OSCORE Option of MSG by using REC_CTX. Building on

the result, the client uses REC_CTX to decrypt and verify MSG, as defined in [I-D.ietf-core-oscore-groupcomm]. The specific operations to perform depend on whether MSG is protected with the group mode or with the pairwise mode of Group OSCORE.

In any other case, the recipient endpoint determines the Recipient Context REC_CTX to use as follows.

1. If CTX is an Incognito Security Context and the length of the "kid" field of the OSCORE Option of MSG is different from 3 bytes, move to Step 17.

   Otherwise, compose SAMPLE_1 as the first N bytes of the CoAP payload of MSG, where N = min(LENGTH, 16) and LENGTH denotes the length in bytes of the CoAP payload of MSG.

   The same considerations about LENGTH from Section 4.2 apply.

2. Compose the 16-byte INPUT_1 by means of the same method at Step 2 of Section 3.1, using as SAMPLE_1 the result of Step 1 of the present algorithm.

3. Compose the 16-byte INPUT_2, by taking INPUT_1 from Step 2 and negating its last bit.

4. Within CTX, select a Recipient Context REC_CTX that has not been tested yet during this execution of the present algorithm.

   In case CTX is not an Incognito Security Context, the selection process can effectively be the one used in Section 6 of [I-D.ietf-core-oscore-groupcomm].

   If no such REC_CTX is found, then move to Step 9. Otherwise, the selected REC_CTX is marked as tested and the following applies:

   * If CTX is an Incognito Security Context, move to Step 5.

   * If CTX is not an Incognito Security Context, check whether the value encoded in the "kid" field of the OSCORE Option of MSG is equal to the Recipient ID specified in REC_CTX.

     In case the two values are equal, REC_CTX is the Recipient Context to use for decrypting and verifying MSG, and this algorithm moves to Step 8. Otherwise, perform Step 4 again.

5. Compute the 16-byte KID_KEYSTREAM by means of the same method at Step 7 of Section 3.1. In particular:

* ENC_KEY is the Obfuscation Recipient Key from the latest REC_CTX selected at Step 4 of the present algorithm.

* INPUT_2 is the result of Step 3 of the present algorithm.

6. Compute the 3-byte STAND_IN_KID value, by XORing with each other:

   * The 3 bytes from LATEST_PIV’s start, where LATEST_PIV is determined as follows.

     - If the OSCORE Option of MSG includes the "Partial IV" field, then LATEST_PIV is the value encoded within that field. Otherwise,

     - MSG is a response and LATEST_PIV is the value encoded within the "Partial IV" field of the OSCORE Option of the corresponding request as it was sent on the wire (i.e., in its obfuscated form).

   * The 3 bytes from KID_KEYSTREAM’s start, where KID_KEYSTREAM is the result of Step 5.

7. If the STAND_IN_KID value computed at Step 6 is not equal to the value encoded in the "kid" field of the OSCORE Option of MSG, then move to Step 4.

   Otherwise, the latest REC_CTX selected at Step 4 is the Recipient Context to use for decrypting and verifying MSG, and this algorithm moves to Step 8.

8. Run the algorithm in Section 4.3.2, in order to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of MSG, by using the latest REC_CTX selected at Step 4.

   Building on the result, the recipient endpoint uses REC_CTX to decrypt and verify MSG, as defined in [I-D.ietf-core-oscore-groupcomm]. The specific operations to perform depend on whether MSG is protected with the group mode or with the pairwise mode of Group OSCORE.

   In case of successful decryption and verification, this algorithm terminates and the recipient endpoint continues processing MSG as expected.

   Otherwise, if MSG is not processed successfully, the following applies:

* If MSG is a request, the recipient endpoint MUST NOT presently reply with any of the optional 4.01 (Unauthorized), 5.03 (Service Unavailable), or 4.00 (Bad Request) error responses that pertain to a failed processing of incoming requests. Note that, although such processing is defined in Sections 5.3, 7.2, and 8.4 of [I-D.ietf-core-oscore-groupcomm], some of the corresponding error handling is inherited from Sections 7.4 and 8.2 of [RFC8613].

* The OSCORE Option of MSG is restored to be as it was before running the algorithm in Section 4.3.2.

* This algorithm moves to Step 4.

9. If the application admits the dynamic derivation of new Recipient Contexts and the recipient endpoint intends to take advantage of that, move to Step 10. Otherwise, move to Step 17.

10. The recipient endpoint contacts the Group Manager responsible for the OSCORE group (see Section 12 of [I-D.ietf-core-oscore-groupcomm]) and retrieves a set of pairs P = (ID, CRED), where ID and CRED in each pair P are the Sender ID and the public authentication credential of a current group member.

    Depending on the particular realization of Group Manager, it can also be possible to retrieve a selected subset of those pairs, e.g., such that the ID specified therein is not part of a list provided in the request to the Group Manager. The realization of Group Manager specified in [I-D.ietf-ace-key-groupcomm-oscore] makes it possible to do so.

11. From the set obtained at Step 10, select a pair P such that:

    * P has not been selected yet during this execution of the present algorithm; and

    * The ID specified within P is not the Recipient ID stored in any of the Recipient Contexts within CTX.

    If no such P is found, then move to Step 17. Otherwise, move to Step 12.

    In case CTX is not an Incognito Security Context, the first pair P to select (if present) should be the one such that the ID specified therein is equal to the value encoded in the "kid" field of the OSCORE Option of MSG.

12. Check if either of the following conditions applies:

    * CTX is an Incognito Security Context; or

    * CTX is not an Incognito Security Context and the value encoded in the "kid" field of the OSCORE Option of MSG is equal to the ID specified within the latest pair P selected at Step 11.

    If any of the two conditions above apply, then establish within CTX a new Recipient Context REC_CTX associated with the same other group member with which the latest pair P selected at Step 11 is associated. That is, within REC_CTX, ID and CRED from P are stored as the Recipient ID and authentication credential associated with the other group member.

    If the first of the two conditions above applies, this algorithm moves to Step 13.

    If the second of the two conditions above applies, REC_CTX is the Recipient Context to use for decrypting and verifying MSG, and this algorithm moves to Step 16.

    If none of the two conditions above applies, this algorithm moves to Step 11.

13. Compute the 16-byte KID_KEYSTREAM by means of the same method at Step 7 of Section 3.1. In particular:

    * ENC_KEY is the Obfuscation Recipient Key from the latest REC_CTX established at Step 12 of the present algorithm.

    * INPUT_2 is the result of Step 3 of the present algorithm.

14. Compute the 3-byte STAND_IN_KID value, by XORing with each other:

    * The 3 bytes from LATEST_PIV’s start, where LATEST_PIV is the same one determined at Step 6.

    * The 3 bytes from KID_KEYSTREAM’s start, where KID_KEYSTREAM is the result of Step 13.

15. If the STAND_IN_KID value computed at Step 14 is not equal to the value encoded in the "kid" field of the OSCORE Option of MSG, then the following applies:

* Depending on what is specified by the application, the recipient endpoint MAY delete the latest REC_CTX established at Step 12.

  If REC_CTX is deleted in this particular circumstance, then this deletion does not require the recipient endpoint to initialize as invalid the Replay Window of any new Recipient Context created later within CTX (see Section 2.6.1.2 of [I-D.ietf-core-oscore-groupcomm]).

* This algorithm moves to Step 11.

Otherwise, the latest REC_CTX established at Step 12 is the Recipient Context to use for decrypting and verifying MSG, and this algorithm moves to Step 16.

16. Run the algorithm in Section 4.3.2, in order to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of MSG, by using the latest REC_CTX established at Step 12.

    Building on the result, the recipient endpoint uses REC_CTX to decrypt and verify MSG, as defined in [I-D.ietf-core-oscore-groupcomm]. The specific operations to perform depend on whether MSG is protected with the group mode or with the pairwise mode of Group OSCORE.

    In case of successful decryption and verification, this algorithm terminates and the recipient endpoint continues processing MSG as expected.

    Otherwise, if MSG is not processed successfully, the following applies:

    * If MSG is a request, the recipient endpoint MUST NOT presently reply with any of the optional 4.01 (Unauthorized), 5.03 (Service Unavailable), or 4.00 (Bad Request) error responses that pertain to a failed processing of incoming requests. Note that, although such processing is defined in Sections 5.3, 7.2, and 8.4 of [I-D.ietf-core-oscore-groupcomm], some of the corresponding error handling is inherited from Sections 7.4 and 8.2 of [RFC8613].

    * Depending on what is specified by the application, the recipient endpoint MAY delete the latest REC_CTX established at Step 12.

If REC_CTX is deleted in this particular circumstance, then this deletion does not require the recipient endpoint to initialize as invalid the Replay Window of any new Recipient Context created later within CTX (see Section 2.6.1.2 of [I-D.ietf-core-oscore-groupcomm]).

* The OSCORE Option of MSG is restored to be as it was before running the algorithm in Section 4.3.2.

* This algorithm moves to Step 11.

17. The following applies:

* If, during this execution of the present algorithm, the server has performed and failed at least one decryption and verification of MSG, the server performs the same error handling defined in Sections 7.2 and 8.4 of [I-D.ietf-core-oscore-groupcomm] for the case where decryption or signature verification has failed. Then, this algorithm terminates. Otherwise,

* If, during this execution of the present algorithm, the server has performed and failed at least one replay check on MSG (see Section 5.3 of [I-D.ietf-core-oscore-groupcomm]), the server performs the same error handling defined in Section 5.3 of [I-D.ietf-core-oscore-groupcomm] for the case where a replay has been detected. Then, this algorithm terminates. Otherwise,

* The server performs the same error handling defined in Sections 7.2 and 8.4 of [I-D.ietf-core-oscore-groupcomm] for the case where a Security Context is not found.

4.3.2. Reversing the Field Obfuscation

Given a Recipient Context RX_CTX that was retrieved according to what is specified in Section 4.3.1, the recipient endpoint performs the following steps, in order to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of the protected incoming message MSG.

1. If the OSCORE Option of MSG includes the "kid" field and CTX is an Incognito Security Context, then move to Step 2. Otherwise, move to Step 3.

2. In the "kid" field of the OSCORE Option of MSG, replace the current STAND_IN_KID value with the Recipient ID specified in RX_CTX.

Unless the Recipient ID has a length of 3 bytes, this step alters the length of the OSCORE Option value. In such a case, the recipient endpoint MUST update the "Option Length" field of the OSCORE Option accordingly (see Section 3.1 of [RFC7252]).

3. If the OSCORE Option of MSG includes the "Partial IV" field, move to Step 4. Otherwise, terminate this algorithm.

4. Compute the 16-byte PIV_KEYSTREAM by means of the same method defined at Step 3 of Section 3.1. In particular:

   * ENC_KEY is the Obfuscation Recipient Key from the RX_CTX that is used during this execution of the present algorithm.

   * INPUT_1 is composed by means of the same method defined at Step 2 of Section 4.3.1. Note that, except for the particular case discussed at the beginning of Section 4.3.1 where the recipient endpoint is a client, INPUT_1 was already computed when actually performing Step 2 of Section 4.3.1.

5. Compute the PIV value, by XORing with each other:

   * The ENC_PIV value encoded within the "Partial IV" field of the OSCORE Option of MSG; and

   * The Q bytes from PIV_KEYSTREAM’s start, where Q is the length in bytes of the "Partial IV" field and PIV_KEYSTREAM is the result of Step 4.

6. In the "Partial IV" field of the OSCORE Option of MSG, replace its current ENC_PIV value with the PIV value computed at Step 5.

4.4. External Signature Checker

TBD

Editor’s note: describe how to ensure that an external signature checker (see Section 7.5 of [I-D.ietf-core-oscore-groupcomm]) can still perform its intended operations, when the "Partial IV" and "kid" fields of the OSCORE Option are obfuscated.

4.5. Deterministic Requests

This section defines an exceptional case where a value smaller than 65536 is used as the Sender Sequence Number and the "kid" field of the OSCORE Option is not overwritten by a STAND_IN_KID value, even if the Group OSCORE Security Context is an Incognito Security Context.

The specification [I-D.ietf-core-cacheable-oscore] defines an approach for restoring cacheability of CoAP responses that are protected end-to-end with Group OSCORE. The approach relies on the concept of Deterministic Request, i.e., a request protected with the pairwise mode of Group OSCORE that any client in the OSCORE group is able to prepare.

When preparing a Deterministic Request, a client effectively impersonates a Deterministic Client, i.e., a fictitious member of the OSCORE group associated with a dedicated OSCORE Sender ID. Also, each Deterministic Request is computed by using the Sender Sequence Number 0.

Therefore, even when using the method defined in the present document, the following applies to the OSCORE Option of a Deterministic Request:

* The "Partial IV" field has a length of 1 byte and encodes the Sender Sequence Number 0.

* The "kid" field conveys the actual OSCORE Sender ID of the Deterministic Client.

  That is, even if the Group OSCORE Security Context used is an Incognito Security Context, the "kid" field in the OSCORE Option of a Deterministic Request is not obfuscated.

5. Agreement on Obfuscating Fields in the OSCORE Option

If an endpoint does not have an explicit agreement with its peer(s) about employing the method specified in this document when using a (Group) OSCORE Security Context CTX, the following applies in order to preserve interoperability:

* The endpoint MUST NOT obfuscate the "Partial IV" and "kid" fields in the OSCORE Option of its outgoing messages protected with CTX.

* The endpoint MUST NOT attempt to reverse the obfuscation of the "Partial IV" and "kid" fields in the OSCORE Option of incoming messages protected with CTX.

The rest of this section defines means that endpoints can use to reach an agreement about obfuscating the "Partial IV" and "kid" fields as per the method specified in this document.

5.1. Agreement for OSCORE

TBD

Editor’s note: expected means to cover include:

* Pre-provisioning
* In EDHOC
* In the OSCORE profile of the ACE framework
* In OMA Lightweight Machine-to-Machine (LwM2M)

5.2. Agreement for Group OSCORE

TBD

Editor’s note: expected means to cover include:

* The OSCORE Group Manager based on the ACE framework
  - Messages to (candidate) group members
  - Messages to external signature verifiers
  - Message to/from an Administrator
* A CoAP server supporting observe multicast notifications and self-managing the OSCORE group for its group observations.

6. Security Considerations

The same security considerations from [RFC8613] and [I-D.ietf-core-oscore-groupcomm] hold for this document when messages are protected with OSCORE or Group OSCORE, respectively. Furthermore, the following considerations also apply.

6.1. Minimum Length of Sender Sequence Numbers

As per Section 3.1, a Sender Sequence Number value has to be at least 65536 when using the method defined in this document.

This ensures that the "Partial IV" field of the OSCORE Option has a length of at least 3 bytes. In turn, this defeats possible attempts to track an endpoint or to fingerprint its traffic that leverage a transition of the length of the "Partial IV" field from 1 to 2 bytes, or from 2 to 3 bytes.

An exception applies to the special case discussed in Section 3.3.1, where the requirement above does not apply for the one-off EDHOC + OSCORE request [RFC9668]. However, the requirement does apply for all the messages that the two endpoints exchange after the EDHOC + OSCORE request and that are protected with the same OSCORE Security Context.

6.2. Limitations

The method defined in this document provides confidentiality protection of the Partial IV against passive adversaries.

An active adversary could guess the plain Partial IV and have a recipient OSCORE endpoint confirm the guesses, e.g., taking advantage of timing side channels. For instance, this can be the case when the recipient endpoint discards an incoming message that is detected as a replay, i.e., without attempting to decrypt and verify the message and hence revealing information through timing side channels.

Similarly, depending on whether the processing of an incoming request message fails due to a replay detection or instead to a failed decryption and verification, the recipient endpoint would follow-up by sending different, unprotected error response messages, which the adversary can leverage to confirm the guesses.

6.3. Encryption Robustness

When performing the steps at Section 3.1 and Section 3.2, using the same Obfuscation Key and SAMPLE_1 more than once risks compromising the encryption of the PIV value in the "Partial IV" field. That is, encrypting the PIV_A and PIV_B values of two different "Partial IV" fields by leveraging the same Obfuscation Key and SAMPLE_1 reveals the exclusive OR of PIV_A and PIV_B.

Assuming that SAMPLE_1 is consistent with the outcome of a pseudorandom function (PRF), if L bits are sampled, then the probability that two SAMPLE_1 byte strings of length L are identical approach P = 2^(-L/2), that is, the birthday bound. For messages protected with (Group) OSCORE, SAMPLE_1 has a minimum length L_MIN of 72 bits and a maximum length L_MAX of 128 bits. Therefore, P is at least 2^36 (when the CoAP payload has a length of L_MIN bits) and at most 2^64 (when the CoAP payload has a length of L_MAX bits or more).

6.4. Impact on Endpoint Trackability

The tracking of an OSCORE endpoint that migrates to a new network path can be largely counteracted by using the method defined in this document, if combined with the use of new source addressing information (e.g., IP address and link-layer address). If addressing information does not change upon network migration, an on-path adversary might still be able to track an endpoint.

Even if combined with the change of addressing information upon network migration, the method defined in this document does not prevent other properties of network packets, e.g., their timing or length, from being used to correlate activities of the same endpoint across different network paths.

6.5. Trial Decryptions

Due to the possible obfuscation of the "kid" field of the OSCORE Option, a recipient endpoint could end up performing trial decryptions on an incoming message, when attempting to retrieve the right Recipient Context to use.

Such trial decryptions will fail, when the recipient endpoint retrieves a Recipient Context that appears to be the right one to use, while in fact it is not and its selection was the result of a false positive matching of (stand-in) key identifiers.

This could be the case in the following circumstances.

* OSCORE is used and the recipient endpoint is a server (see Section 3.2.1).

  - If the recipient endpoint stores at least one Ordinary Security Context, the endpoint first assumes that the "Partial IV" and "kid" fields in the OSCORE Option of the incoming message were not obfuscated.

    Consequently, the endpoint first attempts to retrieve an Ordinary Security Context to decrypt and verify the message (see Step 2 in Section 3.2.1). If a Security Context is found and the OSCORE processing of the incoming message fails, the endpoint proceeds with looking for Obfuscating Security Contexts to use.

- If the recipient endpoint stores Incognito Security Contexts, multiple of those might result in computing a STAND_IN_KID value that is equal to the value encoded in the "kid" field of the OSCORE Option of the incoming message (see Step 10 in Section 3.2.1).

  In case of a positive match, the selected Incognito Security Context is used for the OSCORE processing of the incoming message (see Step 11 in Section 3.2.1), which fails unless the selected Security Context is actually the right one to use. If the OSCORE processing fails, the endpoint continues inspecting the set of Obfuscating Security Contexts.

* Group OSCORE is used and the Group OSCORE Security Context is an Incognito Security Context (see Section 4.3.1).

  - Within the Group OSCORE Security Context, multiple Recipient Contexts might result in computing a STAND_IN_KID value that is equal to the value encoded in the "kid" field of the OSCORE Option of the incoming message (see Step 7 in Section 4.3.1).

    In case of a positive match, the selected Recipient Context is used for the Group OSCORE processing of the incoming message (see Step 8 in Section 4.3.1), which fails unless the selected Recipient Context is actually the right one to use. If the Group OSCORE processing fails, the endpoint continues inspecting the set of Recipient Contexts.

  - If the recipient endpoint does not find an eligible Recipient Context among the stored ones, the endpoint might proceed with the dynamic derivation of new Recipient Contexts, if allowed by the application (see Step 8 in Section 4.3.1).

    After establishing a new Recipient Context, this might result in computing a STAND_IN_KID value that is equal to the value encoded in the "kid" field of the OSCORE Option of the incoming message (see Step 15 in Section 4.3.1).

    In case of a positive match, the newly established Recipient Context is used for the Group OSCORE processing of the incoming message (see Step 16 in Section 4.3.1), which fails unless the Recipient Context is actually the right one to use. If the Group OSCORE processing fails, the endpoint can continue establishing and trying further available Recipient Contexts, as long as information to do so is available.

Although the specific circumstances above are new and introduced by the method defined in this document, trial decryption is in fact already a possibility:

* When using OSCORE, a server receiving a request might end up processing it with multiple OSCORE Security Contexts in sequence. This can happen, e.g., in case of collisions with the values of ID Context and Recipient ID across stored Security Contexts.

* When using OSCORE, a client receiving a response might end up processing it with multiple OSCORE Security Contexts in sequence. This can happen, e.g., in case the same Token value is used for multiple messages exchanges that are simultaneously active.

* When using Group OSCORE, a server receiving a request might end up processing it with multiple Group OSCORE Security Contexts in sequence. This can happen, e.g., in case of collisions with the values of ID Context and Recipient ID across stored Security Contexts. For example, both the sender endpoint and the recipient endpoint can be in two different OSCORE groups under different Group Managers, with both OSCORE groups identified by the same Group ID and with the sender endpoint using the same Sender ID in both groups.

In either case, the recipient endpoint can attempt using multiple available Security Contexts in sequence, until the right one is possibly found and message decryption and verification succeed.

Regardless of the specific circumstance by which trial decryptions occur, recipient endpoints still have to keep updated the status of their keying material, including after decryption failure. In particular, symmetric keys ought not to be used beyond certain key usage limits, i.e., after having reached a maximum number of failed decryptions [I-D.ietf-core-oscore-key-limits][I-D.irtf-cfrg-aead-limits].

Clearly, broadening the opportunities for trial decryption increases the pace by which key usage limits are approached, thereby increasing the frequency by which keying material needs to be updated, e.g., by using the key update procedure KUDOS [I-D.ietf-core-oscore-key-update].

6.6. Not Obfuscating the "kid" Field

When using an Obfuscating Security Context that is not an Incognito Security Context, the method specified in this document results only in obfuscating the "Partial IV" field of the OSCORE Option, but not the "kid" field.

This is useful in some use cases where such information is effectively still obfuscated by other means. In such use cases, obfuscating the "kid" field by using the method defined in this document would simply result in unjustified (performance) penalties.

For example, the specification [I-D.ietf-schc-8824-update] defines how to use the Static Context Header Compression and fragmentation (SCHC) framework [RFC8724] for compressing headers of CoAP messages, also when messages are protected with OSCORE or Group OSCORE.

Two endpoints using (Group) OSCORE and SCHC header compression can enforce SCHC compression Rules that include a Field Descriptor for the "kid" field of the OSCORE Option. The intent of such a SCHC Rule is typically to match with a protected message where the "kid" field conveys the same value specified in the corresponding Field Descriptor of the Rule. As a result, the SCHC compression elides the "kid" field before transmission, and the field will be reconstructed by the recipient endpoint when performing SCHC Decompression per the same SCHC Rule.

In such a scenario, obfuscating the "kid" field by means of the method defined in this document will prevent the intended SCHC compression Rule to match with the protected message to compress, since the stand-in identifier that is overwritten in the "kid" field of the OSCORE Option will differ from the static, expected value specified in the Field Descriptor of the SCHC Rule. In fact, unless other installed SCHC compression Rules produce a match, the message as a whole will simply not be compressed.

To ensure that at least other fields of the message are compressed, the SCHC Rule would instead have to include the Field Descriptor for the "kid" field that always results in preserving the "kid" field as-is, without compression. Clearly, it is instead better to enforce the originally intended SCHC Rule, while not obfuscating the "kid" field through the method specified in this document.

With the exception of such use cases that effectively achieve obfuscation of the "kid" field by other means, endpoints that support the method defined in this document and explicitly agree to use it ought to rely on Incognito Security Contexts and thus obfuscate the "kid" field of the OSCORE Option accordingly.

7. IANA Considerations

TBD

Editor’s note: expected actions are registrations of new parameters that effectively enable the means defined in Section 5.

8. References

8.1. Normative References

[AES] NIST, "Advanced Encryption Standard (AES)", May 2023, <https://doi.org/10.6028/NIST.FIPS.197-upd1>.

[COSE.Algorithms] IANA, "COSE Algorithms", <https://www.iana.org/assignments/cose/cose.xhtml#algorithms>.

[I-D.ietf-core-cacheable-oscore] Amsss, C. and M. Tiloca, "Cacheable OSCORE", Work in Progress, Internet-Draft, draft-ietf-core-cacheable-oscore-00, 22 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-cacheable-oscore-00>.

[I-D.ietf-core-oscore-groupcomm] Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and R. Hglund, "Group Object Security for Constrained RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-28, 23 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-28>.

[I-D.ietf-core-oscore-key-update] Hglund, R. and M. Tiloca, "Key Update for OSCORE (KUDOS)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-key-update-12, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-key-update-12>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/rfc/rfc5869>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/rfc/rfc8613>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9031] Vuini, M., Ed., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", RFC 9031, DOI 10.17487/RFC9031, May 2021, <https://www.rfc-editor.org/rfc/rfc9031>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9528] Selander, G., Preu Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, DOI 10.17487/RFC9528, March 2024, <https://www.rfc-editor.org/rfc/rfc9528>.

[RFC9668] Palombini, F., Tiloca, M., Hglund, R., Hristozov, S., and G. Selander, "Using Ephemeral Diffie-Hellman Over COSE (EDHOC) with the Constrained Application Protocol (CoAP) and Object Security for Constrained RESTful Environments (OSCORE)", RFC 9668, DOI 10.17487/RFC9668, November 2024, <https://www.rfc-editor.org/rfc/rfc9668>.

8.2. Informative References

[I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M. and F. Palombini, "Key Management for Group Object Security for Constrained RESTful Environments (Group OSCORE) Using Authentication and Authorization for

Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-oscore-20, 25 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-oscore-20>.

[I-D.ietf-core-groupcomm-bis]
Dijk, E. and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-18, 10 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-18>.

[I-D.ietf-core-oscore-key-limits]
Hglund, R. and M. Tiloca, "Key Usage Limits for OSCORE", Work in Progress, Internet-Draft, draft-ietf-core-oscore-key-limits-06, 7 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-key-limits-06>.

[I-D.ietf-schc-8824-update]
Tiloca, M., Toutain, L., Martnez, I., and A. Minaburo, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-schc-8824-update-07, 1 December 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-schc-8824-update-07>.

[I-D.irtf-cfrg-aead-limits]
Gnther, F., Thomson, M., and C. A. Wood, "Usage Limits on AEAD Algorithms", Work in Progress, Internet-Draft, draft-irtf-cfrg-aead-limits-11, 4 December 2025, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-aead-limits-11>.

[RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <https://www.rfc-editor.org/rfc/rfc8724>.

Acknowledgments

The authors sincerely thank Christian Amsss, Carsten Bormann, and Martine Lenders for their comments and feedback.

This work was supported by the Sweden’s Innovation Agency VINNOVA within the EUREKA CELTIC-NEXT project CYPRESS.

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-164 40 Kista
Sweden
Email: marco.tiloca@ri.se

John Preu Mattsson
Ericsson AB
SE-164 40 Kista
Sweden
Email: john.mattsson@ericsson.com

Rikard Hglund
RISE AB
Isafjordsgatan 22
SE-164 40 Kista
Sweden
Email: rikard.hoglund@ri.se

Gran Selander
Ericsson AB
SE-164 40 Kista
Sweden
Email: goran.selander@ericsson.com