CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2022

F. Palombini
Ericsson
M. Tiloca
R. Hoeglund
RISE AB
S. Hristozov
Fraunhofer AISEC
G. Selander
Ericsson

July 12, 2021

Combining EDHOC and OSCORE
draft-ietf-core-oscore-edhoc-01

Abstract

This document defines an optimization approach for combining the lightweight authenticated key exchange protocol EDHOC run over CoAP with the first subsequent OSCORE transaction. This combination reduces the number of round trips required to set up an OSCORE Security Context and to complete an OSCORE transaction using that Security Context.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Ephemeral Diffie-Hellman Over COSE (EDHOC) [I-D.ietf-lake-edhoc] is a lightweight authenticated key exchange protocol, especially intended for use in constrained scenarios. In particular, EDHOC messages can be transported over the Constrained Application Protocol (CoAP) [RFC7252] and used for establishing a Security Context for Object Security for Constrained RESTful Environments (OSCORE) [RFC8613].

This document defines an optimization approach that combines EDHOC run over CoAP with the first subsequent OSCORE transaction. This allows for a minimum number of round trips necessary to setup the OSCORE Security Context and complete an OSCORE transaction, for

example when an IoT device gets configured in a network for the first time.

This optimization is desirable, since the number of protocol round trips impacts on the minimum number of flights, which in turn can have a substantial impact on the latency of conveying the first OSCORE request, when using certain radio technologies.

Without this optimization, it is not possible, not even in theory, to achieve the minimum number of flights. This optimization makes it possible also in practice, since the last message of the EDHOC protocol can be made relatively small (see Section 1 of [I-D.ietf-lake-edhoc]), thus allowing additional OSCORE protected CoAP data within target MTU sizes.

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.

The reader is expected to be familiar with terms and concepts defined in CoAP [RFC7252], CBOR [RFC8949], CBOR sequences [RFC8742], OSCORE [RFC8613] and EDHOC [I-D.ietf-lake-edhoc].

2. EDHOC Overview

The EDHOC protocol allows two peers to agree on a cryptographic secret, in a mutually-authenticated way and by using Diffie-Hellman ephemeral keys to achieve perfect forward secrecy. The two peers are denoted as Initiator and Responder, as the one sending or receiving the initial EDHOC message_1, respectively.

After successful processing of EDHOC message_3, both peers agree on a cryptographic secret that can be used to derive further security material, and especially to establish an OSCORE Security Context [RFC8613]. The Responder can also send an optional EDHOC message_4 to achieve key confirmation, e.g., in deployments where no protected application message is sent from the Responder to the Initiator.

Appendix A.3 of [I-D.ietf-lake-edhoc] specifies how to transport EDHOC over CoAP. That is, the EDHOC data (referred to as "EDHOC messages") are transported in the payload of CoAP requests and responses. The default message flow consists in the CoAP Client acting as Initiator and the CoAP Server acting as Responder.

Alternatively, the two roles can be reversed. In the rest of this document, EDHOC messages are considered to be transported over CoAP.

Figure 1 shows a Client and Server running EDHOC as Initiator and Responder, respectively. That is, the Client sends a POST request with payload EDHOC message_1 to a reserved resource at the CoAP Server, by default at Uri-Path "/.well-known/edhoc". This triggers the EDHOC exchange at the Server, which replies with a 2.04 (Changed) Response with payload EDHOC message_2. Finally, the Client sends a CoAP POST request to the same resource used for EDHOC message_1, with payload EDHOC message_3. The Content-Format of these CoAP messages may be set to "application/edhoc".

After this exchange takes place, and after successful verifications as specified in the EDHOC protocol, the Client and Server can derive an OSCORE Security Context, as defined in Appendix A.2 of [I-D.ietf-lake-edhoc]. After that, they can use OSCORE to protect their communications.

```
         CoAP Client                                  CoAP Server
      (EDHOC Initiator)                            (EDHOC Responder)
              | ------------- EDHOC message_1 ------------> |
              |          Header: POST (Code=0.02)           |
              |       Uri-Path: "/.well-known/edhoc"        |
              |      Content-Format: application/edhoc      |
              |                                             |
              | <------------ EDHOC message_2 ------------- |
              |            Header: 2.04 Changed             |
              |      Content-Format: application/edhoc      |
              |                                             |
      EDHOC verification                                    |
              |                                             |
              | ------------- EDHOC message_3 ------------> |
              |          Header: POST (Code=0.02)           |
              |        Uri-Path: "/.well-known/edhoc"       |
              |      Content-Format: application/edhoc      |
              |                                             |
              |                                    EDHOC verification
              |                                             +
      OSCORE Sec Ctx                                OSCORE Sec Ctx
        Derivation                                     Derivation
              |                                             |
              | ------------- OSCORE Request -------------> |
              |          Header: POST (Code=0.02)           |
              |                                             |
              | <------------ OSCORE Response ------------- |
              |            Header: 2.04 Changed             |
              |                                             |
```

Figure 1: EDHOC and OSCORE run sequentially

As shown in Figure 1, this purely-sequential way of first running EDHOC and then using OSCORE takes three round trips to complete.

Section 3 defines an optimization for combining EDHOC with the first subsequent OSCORE transaction. This reduces the number of round trips required to set up an OSCORE Security Context and to complete an OSCORE transaction using that Security Context.

3. EDHOC Combined with OSCORE

This section defines an optimization for combining the EDHOC exchange with the first subsequent OSCORE transaction, thus minimizing the number of round trips between the two peers.

This approach can be used only if the default EDHOC message flow is used, i.e., when the Client acts as Initiator and the Server acts as Responder, while it cannot be used in the case with reversed roles.

When running the purely-sequential flow of Section 2, the Client has all the information to derive the OSCORE Security Context already after receiving EDHOC message_2 and before sending EDHOC message_3.

Hence, the Client can potentially send both EDHOC message_3 and the subsequent OSCORE Request at the same time. On a semantic level, this requires sending two REST requests at once, as in Figure 2.

```
      CoAP Client                                      CoAP Server
   (EDHOC Initiator)                                 (EDHOC Responder)
           | ------------- EDHOC message_1 ------------> |
           |          Header: POST (Code=0.02)           |
           |       Uri-Path: "/.well-known/edhoc"        |
           |      Content-Format: application/edhoc      |
           |                                             |
           | <------------ EDHOC message_2 ------------- |
           |            Header: 2.04 Changed             |
           |      Content-Format: application/edhoc      |
           |                                             |
   EDHOC verification                                    |
           +                                             |
     OSCORE Sec Ctx                                      |
       Derivation                                        |
           |                                             |
           | ---- EDHOC message_3 + OSCORE Request ----> |
           |          Header: POST (Code=0.02)           |
           |                                             |
           |                                    EDHOC verification
           |                                             +
           |                                     OSCORE Sec Ctx
           |                                        Derivation
           |                                             |
           | <------------ OSCORE Response ------------- |
           |            Header: 2.04 Changed             |
           |                                             |
```

Figure 2: EDHOC and OSCORE combined

To this end, the specific approach defined in this section consists of sending EDHOC message_3 inside an OSCORE protected CoAP message.

The resulting EDHOC + OSCORE request is in practice the OSCORE Request from Figure 1, as still sent to a protected resource and with

the correct CoAP method and options, but with the addition that it also transports EDHOC message_3.

As EDHOC message_3 may be too large to be included in a CoAP Option, e.g., if containing a large public key certificate chain, it has to be transported in the CoAP payload of the EDHOC + OSCORE request.

The rest of this section specifies how to transport the data in the EDHOC + OSCORE request and their processing order. In particular, the use of this approach is explicitly signalled by including an EDHOC Option (see Section 3.1) in the EDHOC + OSCORE request. The processing of the EDHOC + OSCORE request is specified in Section 3.2 for the Client side and in Section 3.3 for the Server side.

3.1. EDHOC Option

This section defines the EDHOC Option. The option is used in a CoAP request, to signal that the request payload conveys both an EDHOC message_3 and OSCORE protected data, combined together.

The EDHOC Option has the properties summarized in Figure 3, which extends Table 4 of [RFC7252]. The option is Critical, Safe-to-Forward, and part of the Cache-Key. The option MUST occur at most once and is always empty. If any value is sent, the value is simply ignored. The option is intended only for CoAP requests and is of Class U for OSCORE [RFC8613].

No.	C	U	N	R	Name	Format	Length	Default
TBD21	x				EDHOC	Empty	0	(none)

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

Figure 3: The EDHOC Option.

The presence of this option means that the message payload contains also EDHOC data, that must be extracted and processed as defined in Section 3.3, before the rest of the message can be processed.

Figure 4 shows the format of a CoAP message containing both the EDHOC data and the OSCORE ciphertext, using the newly defined EDHOC option for signalling.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| T |  TKL  |      Code     |          Message ID           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Token (if any, TKL bytes) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  OSCORE option  |   EDHOC option  | Other options (if any) ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 1 1 1 1 1 1|    Payload
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: CoAP message for EDHOC and OSCORE combined - signalled with the EDHOC Option

3.2. Client Processing

The Client prepares an EDHOC + OSCORE request as follows.

1. Compose EDHOC message_3 as per Section 5.4.2 of [I-D.ietf-lake-edhoc].

   Since the Client is the EDHOC Initiator, the EDHOC message_3 always includes the connection identifier C_R and CIPHERTEXT_3. Note that C_R is the OSCORE Sender ID of the Client, encoded as per Appendix A.1.

2. Encrypt the original CoAP request as per Section 8.1 of [RFC8613], using the new OSCORE Security Context established after receiving EDHOC message_2.

   Note that the OSCORE ciphertext is not computed over EDHOC message_3, which is not protected by OSCORE. That is, the result of this step is the OSCORE Request as in Figure 1.

3. Build a CBOR sequence [RFC8742] composed of two CBOR byte strings in the following order.

   * The first CBOR byte string is the CIPHERTEXT_3 of the EDHOC message_3 resulting from step 3.

   * The second CBOR byte string has as value the OSCORE ciphertext of the OSCORE protected CoAP request resulting from step 2.

4. Compose the EDHOC + OSCORE request, as the OSCORE protected CoAP request resulting from step 2, where the payload is replaced with the CBOR sequence built at step 3.

5. Signal the usage of this approach within the EDHOC + OSCORE request, by including the new EDHOC Option defined in Section 3.1.

3.3. Server Processing

When receiving a request containing the EDHOC option, i.e., an EDHOC + OSCORE request, the Server MUST perform the following steps.

1. Check that the payload of the EDHOC + OSCORE request is a CBOR sequence composed of two CBOR byte strings. If this is not the case, the Server MUST stop processing the request and MUST respond with a 4.00 (Bad Request) error message.

2. Extract CIPHERTEXT_3 from the payload of the EDHOC + OSCORE request, as the first CBOR byte string in the CBOR sequence.

3. Rebuild EDHOC message_3, as a CBOR sequence composed of two CBOR byte strings in the following order.

   * The first CBOR byte string is the 'kid' of the Client indicated in the OSCORE option of the EDHOC + OSCORE request (i.e., the OSCORE Sender ID of the Client), encoded as per Appendix A.1.

   * The second CBOR byte string is the CIPHERTEXT_3 retrieved at step 2.

4. Perform the EDHOC processing on the EDHOC message_3 rebuilt at step 3, including verifications as per Section 5.4.3 of [I-D.ietf-lake-edhoc] and the OSCORE Security Context derivation as per Appendix A.2 of [I-D.ietf-lake-edhoc].

   If the applicability statement used in the EDHOC session specifies that EDHOC message_4 shall be sent, the Server MUST stop the EDHOC processing and consider it failed, as due to a client error.

5. Extract the OSCORE ciphertext from the payload of the EDHOC + OSCORE request, as the value of the second CBOR byte string in the CBOR sequence.

6. Rebuild the OSCORE protected CoAP request as the EDHOC + OSCORE request, where the payload is replaced with the OSCORE ciphertext resulting from step 5.

7. Decrypt and verify the OSCORE protected CoAP request resulting from step 6, as per Section 8.2 of [RFC8613], by using the new OSCORE Security Context established at step 4.

8. Process the CoAP request resulting from step 7.

If steps 4 (EDHOC processing) and 7 (OSCORE processing) are both successfully completed, the Server MUST reply with an OSCORE protected response, in order for the Client to achieve key confirmation (see Section 5.4.2 of [I-D.ietf-lake-edhoc]). The usage of EDHOC message_4 as defined in Section 5.5 of [I-D.ietf-lake-edhoc] is not applicable to the approach defined in this document.

If step 4 (EDHOC processing) fails, the server discontinues the protocol as per Section 5.4.3 of [I-D.ietf-lake-edhoc] and responds with an EDHOC error message, formatted as defined in Section 6.2 of [I-D.ietf-lake-edhoc]. In particular, the CoAP response conveying the EDHOC error message MUST have Content-Format set to application/edhoc defined in Section 8.9 of [I-D.ietf-lake-edhoc].

If step 4 (EDHOC processing) is successfully completed but step 7 (OSCORE processing) fails, the same OSCORE error handling applies as defined in Section 8.2 of [RFC8613].

3.4. Example of EDHOC + OSCORE Request

Figure 5 shows an example of EDHOC + OSCORE Request, based on the OSCORE test vector from Appendix C.4 of [RFC8613] and the EDHOC test vector from Appendix D.2 of [I-D.ietf-lake-edhoc]. In particular, the example assumes that:

- o The used OSCORE Partial IV is 0, consistently with the first request protected with the new OSCORE Security Context.

- o The OSCORE Sender ID of the Client is 0x00. This corresponds to the numeric EDHOC connection identifier C_R with value 0, which in EDHOC message_3 is encoded as the CBOR integer 0, hence as 0x00.

- o The EDHOC option is registered with CoAP option number 21.

- OSCORE option value: 0x090020 (3 bytes)

- EDHOC option value: - (0 bytes)

- C_R: 0x00 (1 byte)

- CIPHERTEXT_3: 0x52d5535f3147e85f1cfacd9e78abf9e0a81bbf
  (19 bytes)

- EDHOC message_3: 0x00 52d5535f3147e85f1cfacd9e78abf9e0a81bbf
  (20 bytes)

- OSCORE ciphertext: 0x612f1092f1776f1c1668b3825e (13 bytes)

From there:

- Protected CoAP request (OSCORE message):

```
0x44025d1f               ; CoAP 4-byte header
  00003974               ; Token
  39 6c6f63616c686f7374  ; Uri-Host Option: "localhost"
  63 090020              ; OSCORE Option
  C0                     ; EDHOC Option
  ff 52d5535f3147e85f1cfacd9e78abf9e0a81bbf
     4d612f1092f1776f1c1668b3825e
(57 bytes)
```

Figure 5: Example of CoAP message with EDHOC and OSCORE combined

4. Security Considerations

The same security considerations from OSCORE [RFC8613] and EDHOC [I-D.ietf-lake-edhoc] hold for this document.

TODO (more considerations)

5. IANA Considerations

RFC Editor: Please replace "[[this document]]" with the RFC number of this document and delete this paragraph.

This document has the following actions for IANA.

5.1. CoAP Option Numbers Registry

IANA is asked to enter the following option numbers to the "CoAP Option Numbers" registry defined in [RFC7252] within the "CoRE Parameters" registry.

[

The CoAP option numbers 13 and 21 are both consistent with the properties of the EDHOC Option defined in Section 3.1, and they both allow the EDHOC Option to always result in an overall size of 1 byte. This is because:

- The EDHOC option is always empty, i.e., with zero-length value; and
- Since the OSCORE option with option number 9 is always present in the CoAP request, the EDHOC option would be encoded with a maximum delta of 4 or 12, depending on its option number being 13 or 21.

At the time of writing, the CoAP option numbers 13 and 21 are both unassigned in the "CoAP Option Numbers" registry, as first available and consistent option numbers for the EDHOC option.

This document suggests 21 (TBD21) as option number to be assigned to the new EDHOC option, since both 13 and 21 are consistent for the use case in question, but different use cases or protocols may make better use of the option number 13.

]

Number	Name	Reference
TBD21	EDHOC	[[this document]]

6. Normative References

[I-D.ietf-lake-edhoc]
Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", draft-ietf-lake-edhoc-06 (work in progress), April 2021.

[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/info/rfc2119>.

[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/info/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/info/rfc8174>.

[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/info/rfc8613>.

[RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <https://www.rfc-editor.org/info/rfc8742>.

[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/info/rfc8949>.

Appendix A. Additional OSCORE/EDHOC-related Processing

Appendix A.1 in [I-D.ietf-lake-edhoc] defines a rule for converting from EDHOC connection identifier to OSCORE Sender/Recipient ID.

This appendix defines the rule for converting from OSCORE Sender/ Recipient ID to EDHOC connection identifier, and related processing.

A.1. From OSCORE to EDHOC Identifier

The process defined in this section ensures that every OSCORE Sender/ Recipient ID is converted to only one of the two corresponding, equivalent EDHOC connection identifiers, see Appendix A.1 in [I-D.ietf-lake-edhoc].

An OSCORE Sender/Recipient ID, OSCORE_ID, is converted to an EDHOC connection identifier, EDHOC_ID, as follows.

- If OSCORE_ID is 0 bytes in size, it is converted to the empty byte string EDHOC_ID (0x40 in CBOR encoding).

- If OSCORE_ID is longer than 5 bytes in size, it is converted to a byte-valued EDHOC_ID, i.e., a CBOR byte string with value OSCORE_ID.

  For example, the OSCORE_ID 0x001122334455 is converted to the byte-valued EDHOC_ID 0x001122334455 (0x46001122334455 in CBOR encoding).

- If OSCORE_ID is 1-5 bytes in size, the following applies.

* If OSCORE_ID is a valid CBOR encoding for an integer value (i.e., it can be correctly decoded as a CBOR integer), then it is converted to a numeric EDHOC_ID having OSCORE_ID as its CBOR encoded form.

  For example, the OSCORE_ID 0x01 is converted to the numeric EDHOC_ID 1 (0x01 in CBOR encoding), while the OSCORE_ID 0x2B is converted to the numeric EDHOC_ID -12 (0x2B in CBOR encoding).

* If OSCORE_ID is _not_ a valid CBOR encoding for an integer value (i.e., it _cannot_ be correctly decoded as a CBOR integer), then it is converted to a byte-valued EDHOC_ID having OSCORE_ID as its value.

  For example, the OSCORE_ID 0xFF is converted to the byte-valued EDHOC_ID 0xFF (0x41FF in CBOR encoding).

Implementations can easily determine which case holds for a given OSCORE_ID with no need to try to actually CBOR-decode it, e.g., by using the approach in Appendix A.3.

A.2. EDHOC Message Processing

This section specifies additional EDHOC message processing in addition to what is specified in Section 5 of [I-D.ietf-lake-edhoc].

A.2.1. Initiator Processing of Message 1

The Initiator selects C_I as follows.

1. The Initiator initializes a set ID_SET as the empty set.

2. The Initiator selects an available OSCORE Recipient ID, ID*, which is not included in ID_SET.

3. The Initiator converts ID* to the EDHOC connection identifier C_I, as per Appendix A.1.

4. If the resulting C_I is already used, the Initiator adds ID* to ID_SET and moves back to step 2. Otherwise, it uses C_I as its EDHOC connection identifier.

A.2.2. Responder Processing of Message 1

The Responder MUST discontinue the protocol and reply with an EDHOC error message, if C_I is a CBOR byte string and it has as value a valid CBOR encoding of an integer value (e.g., C_I is CBOR encoded as 0x4100).

In fact, this would mean that the Initiator has not followed the conversion rule in Appendix A.1 when converting its (to be) OSCORE Recipient ID to C_I.

A.2.3. Responder Processing of Message 2

The Responder selects C_R as follows.

1. The Responder initializes a set ID_SET as the empty set.

2. The Responder selects an available OSCORE Recipient ID, ID*, which is not included in ID_SET.

3. The Responder converts ID* to the EDHOC connection identifier C_R, as per Appendix A.1.

4. If the resulting C_R is already used or it is equal byte-by-byte to the C_I specified in EDHOC message_1, the Initiator adds ID* to ID_SET and moves back to step 2. Otherwise, it uses C_R as its EDHOC connection identifier.

A.2.4. Initiator Processing of Message 2

The Initiator MUST discontinue the protocol and reply with an EDHOC error message, if any of the following conditions holds.

- C_R is equal byte-by-byte to the C_I that was specified in EDHOC message_1.

- C_R is a CBOR byte string and it has as value a valid CBOR encoding of an integer value (e.g., C_R is CBOR encoded as 0x4100).

  In fact, this would mean that the Responder has not followed the conversion rule in Appendix A.1 when converting its (to be) OSCORE Recipient ID to C_R.

A.3. Checking CBOR Encoding of Numeric Values

Given a binary string of N bytes in size, it is a valid CBOR encoding of an integer value if and only if, for that size N, its first byte is equal to one of the byte values specified in the "First byte" column of the table below.

N	First byte
1	0x00-0x17 , 0x20-0x37
2	0x18 , 0x38
3	0x19 , 0x39
4	0x1A , 0x3A
5	0x1B , 0x3B

Appendix B. Document Updates

RFC Editor: Please remove this section.

B.1. Version -00 to -01

- o Improved background overview of EDHOC.
- o Added explicit rules for converting OSCORE Sender/Recipient IDs to EDHOC connection identifiers following the removal of bstr_identifier from EDHOC.
- o Revised section organization.
- o Recommended number for EDHOC option changed to 21.
- o Editorial improvements.

Acknowledgments

The authors sincerely thank Christian Amsuess, Klaus Hartke, Jim Schaad and Malisa Vucinic for their feedback and comments in the discussion leading up to this draft.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; and by the H2020 project SIFIS-Home (Grant agreement 952652).

Authors' Addresses

Francesca Palombini
Ericsson

Email: francesca.palombini@ericsson.com

Marco Tiloca
RISE AB
Isafjordsgatan 22
Kista SE-16440 Stockholm
Sweden

Email: marco.tiloca@ri.se

Rikard Hoeglund
RISE AB
Isafjordsgatan 22
Kista SE-16440 Stockholm
Sweden

Email: rikard.hoglund@ri.se

Stefan Hristozov
Fraunhofer AISEC

Email: stefan.hristozov@aisec.fraunhofer.de

Goeran Selander
Ericsson

Email: goran.selander@ericsson.com