Workgroup: SCHC Working Group
Internet-Draft:
draft-tiloca-schc-8824-update-00
Updates: 8824 (if approved)
Published: 3 April 2023
Intended Status: Standards Track
Expires: 5 October 2023
Authors: M. Tiloca L. Toutain I. Martinez
RISE AB IMT Atlantique IMT Atlantique

Clarifications and Updates on using Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)

Abstract

This document clarifies, updates and extends the method specified in RFC 8824 for compressing Constrained Application Protocol (CoAP) headers using the Static Context Header Compression and fragmentation (SCHC) framework. In particular, it considers recently defined CoAP options and specifies how CoAP headers are compressed in the presence of intermediaries. Therefore, this document updates RFC 8824.

Discussion Venues

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

Discussion of this document takes place on the Static Context Header Compression Working Group mailing list (schc@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/schc/.

Source for this draft and an issue tracker can be found at https://gitlab.com/crimson84/draft-tiloca-schc-8824-update.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 5 October 2023.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] is a web-transfer protocol intended for applications based on the REST

(Representational State Transfer) paradigm, and designed to be affordable also for resource-constrained devices.

In order to enable the use of CoAP in LPWANs (Low-Power Wide-Area Networks) as well as to improve performance, [RFC8824] defines how to use the Static Context Header Compression and fragmentation (SCHC) framework [RFC8724] for compressing CoAP headers.

This document clarifies, updates and extends the SCHC compression of CoAP headers defined in [RFC8824] at the application level, by: providing specific clarifications; updating specific details of the compression processing, based on recent developments related to the security protocol OSCORE [RFC8613] for end-to-end protection of CoAP messages; and extending the compression processing to take into account additional CoAP options and the presence of CoAP proxies.

In particular, this document updates [RFC8824] as follows.

- It clarifies the SCHC compression for the CoAP options Size1, Size2, Proxy-URI and Proxy-Scheme (see Section 2.1).

- It defines the SCHC compression for the CoAP option Hop-Limit (see Section 2.2).

- It defines the SCHC compression for the recently defined CoAP options Echo (see Section 2.3), Request-Tag (see Section 2.4), EDHOC (see Section 2.5), as well as Q-Block1 and Q-Block2 (see Section 3.1).

- It updates the SCHC compression processing for the CoAP option OSCORE (see Section 3.2), in the light of recent developments related to the security protocol OSCORE as defined in [I-D.ietf-core-oscore-key-update] and [I-D.ietf-core-oscore-groupcomm].

- It clarifies how the SCHC compression handles the CoAP payload marker (see Section 4).

- It defines the SCHC compression of CoAP headers in the presence of CoAP proxies (see Section 5).

This document does not alter the core approach, design choices and features of the SCHC compression applied to CoAP headers.

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 the SCHC framework [RFC8724], the web-transfer protocol CoAP [RFC7252], the security protocol OSCORE [RFC8613] and the use of SCHC for CoAP [RFC8824].

2. CoAP Options

This section updates and extends Section 5 of [RFC8824], as to how SCHC compresses some specific CoAP options. In particular, Section 2.1 updates Section 5.4 of [RFC8824].

2.1. CoAP Option Size1, Size2, Proxy-URI, and Proxy-Scheme Fields

The SCHC Rule description MAY define sending some field values by describing an empty TV, with the MO set to "ignore" and the CDA set to "value-sent". A Rule MAY also use a "match-mapping" MO when there are different options for the same FID. Otherwise, the Rule sets the TV to the value, the MO to "equal", and the CDA to "not-sent".

2.2. CoAP Option Hop-Limit Field

The Hop-Limit field is an option defined in [RFC8768] that can be used to detect forwarding loops through a chain of CoAP proxies. The first proxy in the chain that understands the option includes it in a received request with a proper value set, before forwarding the request. Any following proxy that understands the option decrements the option value and forwards the request if the new value is different than zero, or returns a 5.08 (Hop Limit Reached) error response otherwise.

When a packet uses the Hop-Limit option, SCHC compression MUST send its content in the Compression Residue. The SCHC Rule describes an empty TV with the MO set to "ignore" and the CDA set to "value-sent".

2.3. CoAP Option Echo Field

The Echo field is an option defined in [RFC9175] that a server can include in a response as a challenge to the client, and that the client echoes back to the server in one or more requests. This enables the server to verify the freshness of a request and to cryptographically verify the aliveness of the client. Also, it forces the client to demonstrate reachability at its claimed network address.

When a packet uses the Echo option, SCHC compression MUST send its content in the Compression Residue. The SCHC Rule describes an empty TV with the MO set to "ignore" and the CDA set to "value-sent".

2.4. CoAP Option Request-Tag Field

The Request-Tag field is an option defined in [RFC9175] that the client can set in request messages of block-wise operations, with value an ephemeral short-lived identifier of the specific block-wise operation in question. This allows the server to match message fragments belonging to the same request operation and, if the server supports it, to reliably process simultaneous block-wise request operations on a single resource. If requests are integrity protected, this also protects against interchange of fragments between different block-wise request operations.

When a packet uses the Request-Tag option, SCHC compression MUST send its content in the Compression Residue. The SCHC Rule describes an empty TV with the MO set to "ignore" and the CDA set to "value-sent".

2.5. CoAP Option EDHOC Field

The EDHOC field is an option defined in [I-D.ietf-core-oscore-edhoc] that a client can include in a request, in order to perform an optimized, shortened execution of the authenticated key establishment protocol EDHOC [I-D.ietf-lake-edhoc]. Such a request conveys both the final EDHOC message and actual application data, where the latter is protected with OSCORE [RFC8613] using a Security Context derived from the result of the current EDHOC execution.

The option occurs at most once and is always empty. The SCHC Rule MUST describe an empty TV, with the MO set to "equal" and the CDA set to "not-sent".

3. SCHC Compression of CoAP Extensions

This section updates and extends Section 6 of [RFC8824], as to how SCHC compresses some specific CoAP options providing protocol extensions. In particular, Section 3.1 updates Section 6.1 of [RFC8824], while Section 3.2 updates Section 6.4 of [RFC8824].

3.1. Block

When a packet uses a Block1 or Block2 option [RFC7959] or a Q-Block1 or Q-Block2 option [RFC9177], SCHC compression MUST send its content in the Compression Residue. The SCHC Rule describes an empty TV with the MO set to "ignore" and the CDA set to "value-sent". The Block1, Block2, Q-Block1 and Q-Block2 options allow fragmentation at the CoAP level that is compatible with SCHC fragmentation. Both

fragmentation mechanisms are complementary, and the node may use them for the same packet as needed.

3.2. OSCORE

The security protocol OSCORE [RFC8613] provides end-to-end protection for CoAP messages. Group OSCORE [I-D.ietf-core-oscore-groupcomm] builds on OSCORE and defines end-to-end protection of CoAP messages in group communication [I-D.ietf-core-groupcomm-bis]. This section describes how SCHC Rules can be applied to compress messages protected with OSCORE or Group OSCORE.

Figure 1 shows the OSCORE option value encoding, which was originally defined in Section 6.1 of [RFC8613] and has been extended in [I-D.ietf-core-oscore-key-update] [I-D.ietf-core-oscore-groupcomm]. The first byte of the OSCORE option value specifies the content of the OSCORE option using flags, as follows.

*As defined in Section 4.1 of [I-D.ietf-core-oscore-key-update], the eight least significant bit, when set, indicates that the OSCORE option includes a second byte of flags. The seventh least significant bit is currently unassigned.

*As defined in Section 5 of [I-D.ietf-core-oscore-groupcomm], the sixth least significant bit, when set, indicates that the message including the OSCORE option is protected with the group mode of Group OSCORE (see Section 8 of [I-D.ietf-core-oscore-groupcomm]). When not set, the bit indicates that the message is protected either with OSCORE, or with the pairwise mode of Group OSCORE (see Section 9 of [I-D.ietf-core-oscore-groupcomm]), while the specific OSCORE Security Context used to protect the message determines which of the two cases applies.

*As defined in Section 6.1 of [RFC8613], bit h, when set, indicates the presence of the kid context field in the option. Also, bit k, when set, indicates the presence of a kid field. Finally, the three least significant bits form the field n, which indicates the length of the piv (Partial Initialization Vector) field in bytes. When n = 0, no piv is present.

Assuming the presence of a single flag byte, this is followed by the piv field, the kid context field, and the kid field, in that order. Also, if present, the kid context field's length (in bytes) is encoded in the first byte, denoted by "s".

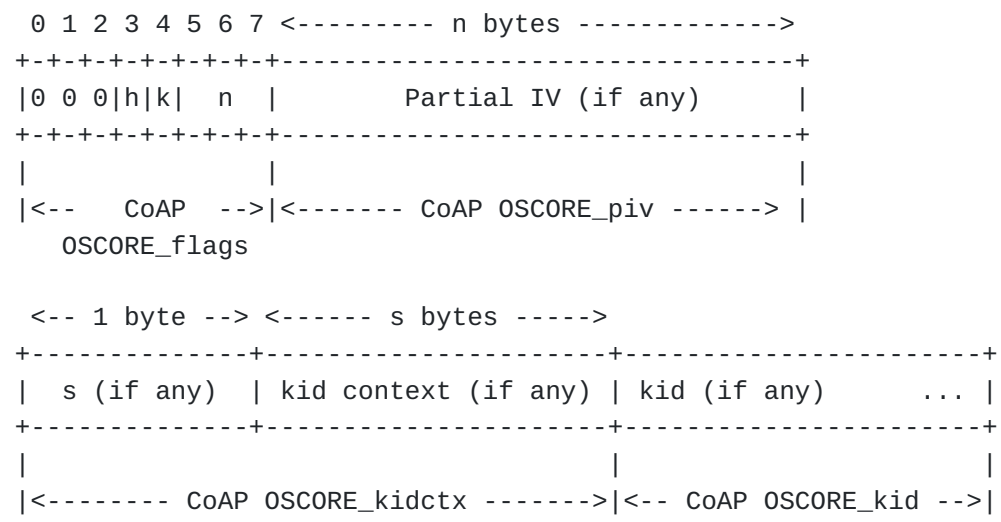

Figure 1: OSCORE Option

Figure 2 shows the OSCORE option value encoding, with the second byte of flags also present. As defined in Section 4.1 of [I-D.ietf-core-oscore-key-update], the least significant bit d of this byte, when set, indicates that two additional fields are included in the option, following the kid context field (if any).

These two fields, namely x and nonce, are used when running the key update protocol KUDOS defined in [I-D.ietf-core-oscore-key-update], with x specifying the length of the nonce field in bytes as well as the specific behavior to adopt during the KUDOS execution. In particular, the figure provides the breakdown of the x field, where its three least significant bits form the sub-field m, which specifies the size of nonce in bytes, minus 1.

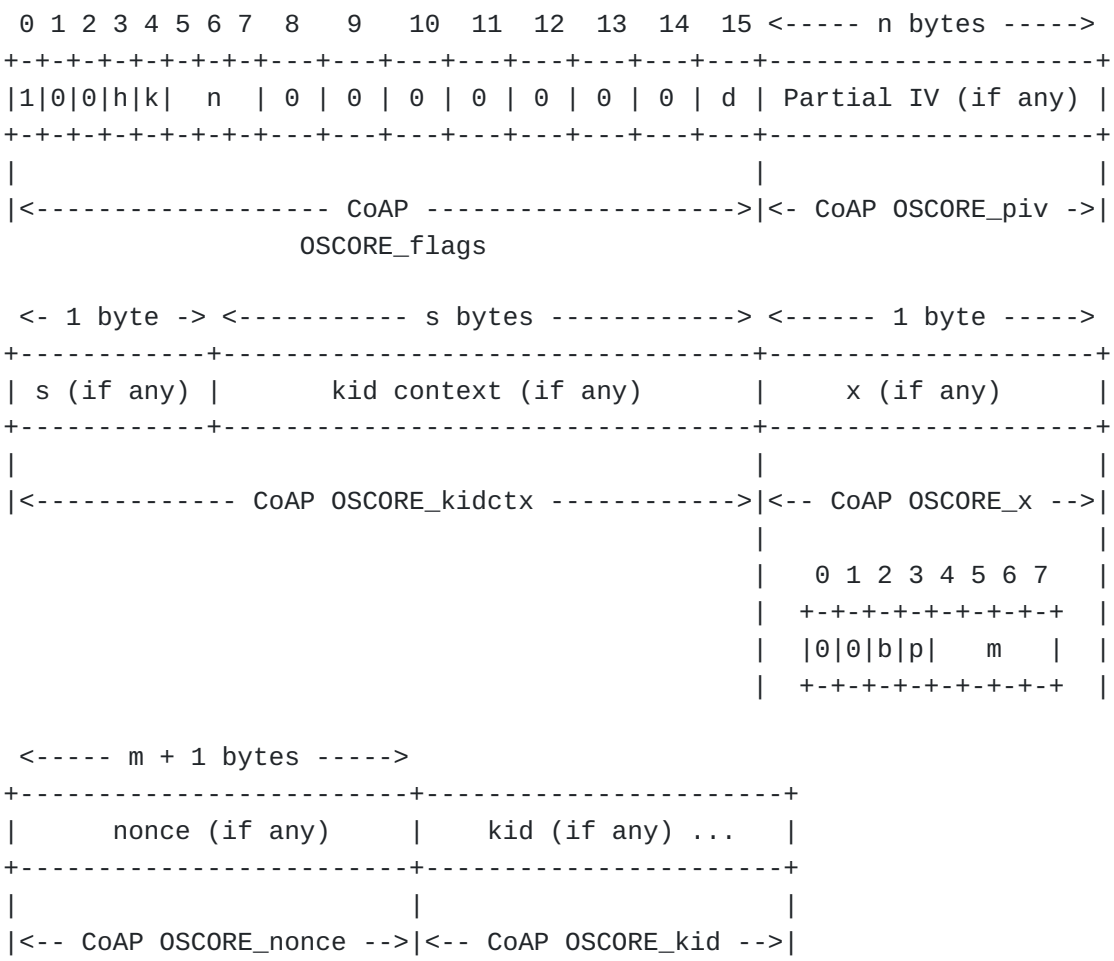

Figure 2: OSCORE Option during a KUDOS execution

To better perform OSCORE SCHC compression, the Rule description needs to identify the OSCORE option and the fields it contains. Conceptually, it discerns up to six distinct pieces of information within the OSCORE option: the flag bits, the piv, the kid context, the x byte, the nonce, and the kid. The SCHC Rule splits the OSCORE option into six Field Descriptors in order to compress them:

- CoAP OSCORE_flags
- CoAP OSCORE_piv
- CoAP OSCORE_kidctx
- CoAP OSCORE_x
- CoAP OSCORE_nonce
- CoAP OSCORE_kid

Figure 1 shows the OSCORE option format with the four fields OSCORE_flags, OSCORE_piv, OSCORE_kidctx and OSCORE_kid superimposed

on it. Also, Figure 2 shows the OSCORE option format with all the six fields superimposed on it, with reference to a message exchanged during an execution of the KUDOS key update protocol.

In both cases, the CoAP OSCORE_kidctx field directly includes the size octet, s. In the latter case, the following applies.

* For the x field, if both endpoints know the value, then the SCHC Rule will describe a TV to this value, with the MO set to "equal" and the CDA set to "not-sent". This models the case where the two endpoints run KUDOS with a pre-agreed size of the nonce field, as well as with a pre-agreed combination of its modes of operations, as per the bits b and p of the m sub-field.

  Otherwise, if the value is changing over time, the SCHC Rule will set the MO to "ignore" and the CDA to "value-sent". The Rule may also use a "match-mapping" MO to compress this field, in case the two endpoints pre-agree on a set of alternative ways to run KUDOS, with respect to the size of the nonce field and the combination of the KUDOS modes of operation to use.

* For the nonce field, the SCHC Rule describes an empty TV with the MO set to "ignore" and the CDA set to "value-sent".

  In addition, for the value of the nonce field, SCHC MUST NOT send it as variable-length data in the Compression Residue, to avoid ambiguity with the length of the nonce field encoded in the x field. Therefore, SCHC MUST use the m sub-field of the x field to define the size of the Compression Residue. SCHC designates a specific function, "osc.x.m", that the Rule MUST use to complete the Field Descriptor. During the decompression, this function returns the length of the nonce field in bytes, as the value of the three least significant bits of the m sub-field of the x field, plus 1.

4. Compression of the CoAP Payload Marker

As originally intended in [RFC8824], the following applies with respect to the 0xFF payload marker. A SCHC compression rule for CoAP includes all the expected CoAP options, therefore the payload marker does not have to be specified.

4.1. Without End-to-End Security

If the CoAP message to compress with SCHC is not going to be protected with OSCORE and includes a payload, then the 0xFF payload marker MUST NOT be included in the compressed message, which is composed of the Compression RuleID, the Compression Residue (if any), and the CoAP payload.

After having decompressed an incoming message, the recipient endpoint MUST prepend the 0xFF payload marker to the CoAP payload, if any was present after the consumed Compression Residue.

4.2. With End-to-End Security

If the CoAP message has to be protected with OSCORE, the same rationale described in Section 4.1 applies to both the Inner SCHC Compression and the Outer SCHC Compression defined in Section 7.2 of [RFC8824]. That is:

- After the Inner SCHC Compression of a CoAP message including a payload, the payload marker MUST NOT be included in the input to the AEAD Encryption, which is composed of the Inner Compression RuleID, the Inner Compression Residue (if any), and the CoAP payload.

- The Outer SCHC Compression takes as input the OSCORE-protected message, which always includes a payload (i.e., the OSCORE Ciphertext) preceded by the payload marker.

- After the Outer SCHC Compression, the payload marker MUST NOT be included in the final compressed message, which is composed of the Outer Compression RuleID, the Outer Compression Residue (if any), and the OSCORE Ciphertext.

After having completed the Outer SCHC Decompression of an incoming message, the recipient endpoint MUST prepend the 0xFF payload marker to the OSCORE Ciphertext.

After having completed the Inner SCHC Decompression of an incoming message, the recipient endpoint MUST prepend the 0xFF payload marker to the CoAP payload, if any was present after the consumed Compression Residue.

5. CoAP Header Compression with Proxies

Building on [RFC8824], this section clarifies how SCHC Compression/Decompression is performed when CoAP proxies are deployed. The following refers to the origin client and origin server as application endpoints.

5.1. Without End-to-End Security

In case OSCORE is not used end-to-end between client and server, the SCHC processing occurs hop-by-hop, by relying on SCHC Rules that are consistently shared between two adjacent hops.

In particular, SCHC is used as defined below.

* The sender application endpoint compresses the CoAP message, by using the SCHC Rules that it shares with the next hop towards the recipient application endpoint. The resulting, compressed message is sent to the next hop towards the recipient application endpoint.

* Each proxy decompresses the incoming compressed message, by using the SCHC Rules that it shares with the (previous hop towards the) sender application endpoint.

  Then, the proxy compresses the CoAP message to be forwarded, by using the SCHC Rules that it shares with the (next hop towards the) recipient application endpoint.

  The resulting, compressed message is sent to the (next hop towards the) recipient application endpoint.

* The recipient application endpoint decompresses the incoming compressed message, by using the SCHC Rules that it shares with the previous hop towards the sender application endpoint.

5.2. With End-to-End Security

In case OSCORE is used end-to-end between client and server (see Section 7.2 of [RFC8824]), the following applies.

The SCHC processing occurs end-to-end as to the Inner SCHC Compression/Decompression, by relying on Inner SCHC Rules that are consistently shared between the two application endpoints acting as OSCORE endpoints and sharing the used OSCORE Security Context.

Instead, the SCHC processing occurs hop-by-hop as to the Outer SCHC Compression/Decompression, by relying on Outer SCHC Rules that are consistently shared between two adjacent hops.

In particular, SCHC is used as defined below.

* The sender application endpoint performs the Inner SCHC Compression on the original CoAP message, by using the Inner SCHC Rules that it shares with the recipient application endpoint.

  Following the AEAD Encryption of the compressed input obtained from the previous step, the sender application endpoint performs the Outer SCHC Compression on the resulting OSCORE-protected message, by using the Outer SCHC Rules that it shares with the next hop towards the recipient application endpoint.

The resulting, compressed message is sent to the next hop towards the recipient application endpoint.

* Each proxy performs the Outer SCHC Decompression on the incoming compressed message, by using the SCHC Rules that it shares with the (previous hop towards the) sender application endpoint.

  Then, the proxy performs the Outer SCHC Compression of the OSCORE-protected message to be forwarded, by using the SCHC Rules that it shares with the (next hop towards the) recipient application endpoint.

  The resulting, compressed message is sent to the (next hop towards the) recipient application endpoint.

* The recipient application endpoint performs the Outer SCHC Decompression on the incoming compressed message, by using the Outer SCHC Rules that it shares with the previous hop towards the sender application endpoint.

  Then, the recipient application endpoint performs the AEAD Decryption of the OSCORE-protected message obtained from the previous step.

  Finally, the recipient application endpoint performs the Inner SCHC Decompression on the compressed input obtained from the previous step, by using the Inner SCHC rules that it shares with the sender application endpoint. The result is the original CoAP message produced by the sender application endpoint.

6. Examples of CoAP Header Compression with Proxies

TBD

7. Security Considerations

The security considerations discussed in [RFC8724] and [RFC8824] continue to apply. When SCHC is used in the presence of CoAP proxies, the security considerations discussed in Section 11.2 of [RFC7252] continue to apply. When SCHC is used with OSCORE, the security considerations discussed in [RFC8613] continue to apply.

The security considerations in [RFC8824] specifically discuss how the use of SCHC for CoAP when OSCORE is also used may result in (more frequently) triggering key-renewal operations for the two endpoints. This can be due to an earlier exhaustion of the OSCORE Sender Sequence Number space, or to the installation of new compression Rules on one of the endpoints.

In either case, the two endpoints can run the key update protocol KUDOS defined in [I-D.ietf-core-oscore-key-update], as the recommended method to update their shared OSCORE Security Context.

8. IANA Considerations

This document has no actions for IANA.

9. References

9.1. Normative References

[I-D.ietf-core-oscore-edhoc] Palombini, F., Tiloca, M., Höglund, R., Hristozov, S., and G. Selander, "Using EDHOC with CoAP and OSCORE", Work in Progress, Internet-Draft, draft-ietf-core-oscore-edhoc-07, 13 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-edhoc-07>.

[I-D.ietf-core-oscore-groupcomm] Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-17, 20 December 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-17>.

[I-D.ietf-core-oscore-key-update] Höglund, R. and M. Tiloca, "Key Update for OSCORE (KUDOS)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-key-update-04, 13 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-key-update-04>.

[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>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959,

DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[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>.

[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/info/rfc8724>.

[RFC8768] Boucadair, M., Reddy.K, T., and J. Shallow, "Constrained Application Protocol (CoAP) Hop-Limit Option", RFC 8768, DOI 10.17487/RFC8768, March 2020, <https://www.rfc-editor.org/info/rfc8768>.

[RFC8824] Minaburo, A., Toutain, L., and R. Andreasen, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", RFC 8824, DOI 10.17487/RFC8824, June 2021, <https://www.rfc-editor.org/info/rfc8824>.

[RFC9175] Amsüss, C., Preuß Mattsson, J., and G. Selander, "Constrained Application Protocol (CoAP): Echo, Request-Tag, and Token Processing", RFC 9175, DOI 10.17487/RFC9175, February 2022, <https://www.rfc-editor.org/info/rfc9175>.

[RFC9177] Boucadair, M. and J. Shallow, "Constrained Application Protocol (CoAP) Block-Wise Transfer Options Supporting Robust Transmission", RFC 9177, DOI 10.17487/RFC9177, March 2022, <https://www.rfc-editor.org/info/rfc9177>.

9.2. Informative References

[I-D.ietf-core-groupcomm-bis] Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-08, 11 January 2023,

<https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-08>.

[I-D.ietf-lake-edhoc] Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-19, 3 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-19>.

Appendix A. YANG data model

TBD

Acknowledgments

The authors sincerely thank Göran Selander for his comments and feedback.

The work on this document has been partly supported by the H2020 projects SIFIS-Home (Grant agreement 952652) and ARCADIAN-IoT (Grant agreement 101020259).

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden

Email: marco.tiloca@ri.se

Laurent Toutain
IMT Atlantique
CS 17607, 2 rue de la Chataigneraie
35576 Cesson-Sevigne Cedex
France

Email: Laurent.Toutain@imt-atlantique.fr

Ivan Martinez
IMT Atlantique
CS 17607, 2 rue de la Chataigneraie
35576 Cesson-Sevigne Cedex
France

Email: ivanmarinomartinez@gmail.com