SCHC Working Group
Internet-Draft
Obsoletes: 8824 (if approved)
Intended status: Standards Track
Expires: 8 January 2026

M. Tiloca
RISE AB
L. Toutain
IMT Atlantique
I. Martínez
IRISA
A. Minaburo
Consultant

7 July 2025

Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)
draft-ietf-schc-8824-update-05

Abstract

This document defines how to compress Constrained Application Protocol (CoAP) headers using the Static Context Header Compression and fragmentation (SCHC) framework. SCHC defines a header compression mechanism adapted for constrained devices. SCHC uses a static description of the header to reduce the header's redundancy and size. While RFC 8724 describes the SCHC compression and fragmentation framework and its application for IPv6 and UDP headers, this document applies SCHC to CoAP headers. The CoAP header structure differs from that of IPv6 and UDP headers, since CoAP uses a flexible header with a variable number of options that are in turn of variable length. The CoAP message format is asymmetric, i.e., request messages have a header format different from that of response messages. This specification gives guidance on applying SCHC to flexible headers and on leveraging the message format asymmetry for defining more efficient compression Rules. This document replaces and obsoletes 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://github.com/ietf-wg-schc/draft-ietf-schc-8824-update.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] is a request/response protocol designed for microcontrollers with small RAM and ROM, and optimized for services based on REST (Representational State Transfer). Although the constrained devices are a leading factor in the design of CoAP, a CoAP header's size is still too large for LPWANs (Low-Power Wide-Area Networks). Static Context Header Compression and fragmentation (SCHC) over CoAP headers is required to increase performance or to use CoAP over LPWAN technologies.

[RFC8724] defines the SCHC framework, which includes a header compression mechanism for LPWANs that is based on a static context. Section 5 of [RFC8724] explains where compression and decompression occur in the architecture. The SCHC compression scheme assumes as a prerequisite that both endpoints know the static context before transmission. The way the context is configured, provisioned, or exchanged is out of the scope of this document.

Since CoAP is an application-layer protocol, compressing CoAP headers requires installing common Rules between the two SCHC instances. SCHC compression may apply at two different levels: at the IP and UDP level in the LPWAN, as well as at the application level for CoAP. These two compression techniques may be independent. Both follow the same principle as that described in [RFC8724]. As different entities manage the CoAP compression process at different levels, the SCHC Rules driving the compression/decompression are also different. [RFC8724] describes how to use SCHC for IP and UDP headers. This document specifies how to apply SCHC compression to CoAP headers.

SCHC compresses and decompresses headers based on common contexts between Devices. The SCHC context includes multiple Rules. Each Rule can match the header fields to specific values or ranges of values. If a Rule matches, the matched header fields are replaced by the RuleID and the Compression Residue that contains the residual bits of the compression. Thus, different Rules may correspond to different protocol headers in the packet that a Device expects to send or receive.

A Rule describes the packets' entire header with an ordered list of Field Descriptors (see Section 7 of [RFC8724]). In turn, each Field Descriptor contains the Field ID (FID), Field Length (FL), and Field Position (FP), as well as a Direction Indicator (DI) (upstream,

downstream, or bidirectional) and some associated Target Values (TVs). The DI allows the compression to be based on the best TV for the Field Descriptor, when the TV to consider is different for the different transmission directions. Therefore, a field may be described several times in the same Rule.

Furthermore, a Matching Operator (MO) is associated with each header Field Descriptor. The Rule is selected if all the MOs fit the TVs for all the fields of the header. A Rule cannot be selected if the message contains a field that is unknown to the SCHC compressor.

In that case, a Compression/Decompression Action (CDA) associated with each field specifies the method to compress and decompress that field. Compression mainly results in one of four actions:

* send the field value (value-sent),

* send nothing (not-sent),

* send some Least Significant Bits (LSBs) of the field, or

* send an index (mapping-sent).

After applying the compression, there may be some bits to be sent. These values are called "Compression Residue".

SCHC is a general mechanism applied to different protocols, with the exact Rules to be used depending on the protocol and the application. Section 10 of [RFC8724] describes the compression scheme for IPv6 and UDP headers. This document targets CoAP header compression using SCHC.

The use of SCHC compression applied to CoAP headers was originally defined in [RFC8824]. While this document does not alter the core approach, design choices, and features specified therein, this document clarifies, updates, and extends the SCHC compression of CoAP headers defined in [RFC8824].

In particular, this documents replaces and obsoletes [RFC8824] as follows:

* It provides clarifications and amendments to the original specification text, based on collected feedback and reported errata.

* It clarifies how the SCHC compression handles CoAP options in general (see Section 5.1).

* It clarifies the SCHC compression for the CoAP options: Size1, Size2, Proxy-Uri, and Proxy-Scheme (see Section 5.5); ETag and If-Match (see Section 5.8); and If-None-Match (see Section 5.9).

* It defines the SCHC compression for the recently defined CoAP options Proxy-Cri and Proxy-Scheme-Number (see Section 5.6).

* It defines the SCHC compression for the CoAP option Hop-Limit (see Section 5.10).

* It defines the SCHC compression for the recently defined CoAP options Echo (see Section 5.11), Request-Tag (see Section 5.12), EDHOC (see Section 5.13), as well as Q-Block1 and Q-Block2 (see Section 6.1).

* It updates the SCHC compression processing for the CoAP option OSCORE (see Section 6.4), also in the light of recent developments related to the security protocol Object Security for Constrained RESTful Environments (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 7).

* It defines the SCHC compression of CoAP headers in the presence of CoAP proxies (see Section 9), for which examples are provided (see Section 10).

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], CoAP [RFC7252], and the security protocols OSCORE [RFC8613] and Group Object Security for Constrained RESTful Environments (Group OSCORE) [I-D.ietf-core-oscore-groupcomm].

2. SCHC Applicability to CoAP

SCHC compression for CoAP headers MAY be done in conjunction with the lower layers (IPv6/UDP) or independently. The SCHC adaptation layers described in Section 5 of [RFC8724] may be used as shown in Figure 1, Figure 2, and Figure 3 below.

In the first example depicted in Figure 1, a Rule compresses the complete header stack from IPv6 to CoAP. In this case, the Device and the Network Gateway (NGW) perform SCHC Compression/Decompression (SCHC C/D), see [RFC8724]). The application communicating with the Device does not implement SCHC C/D.

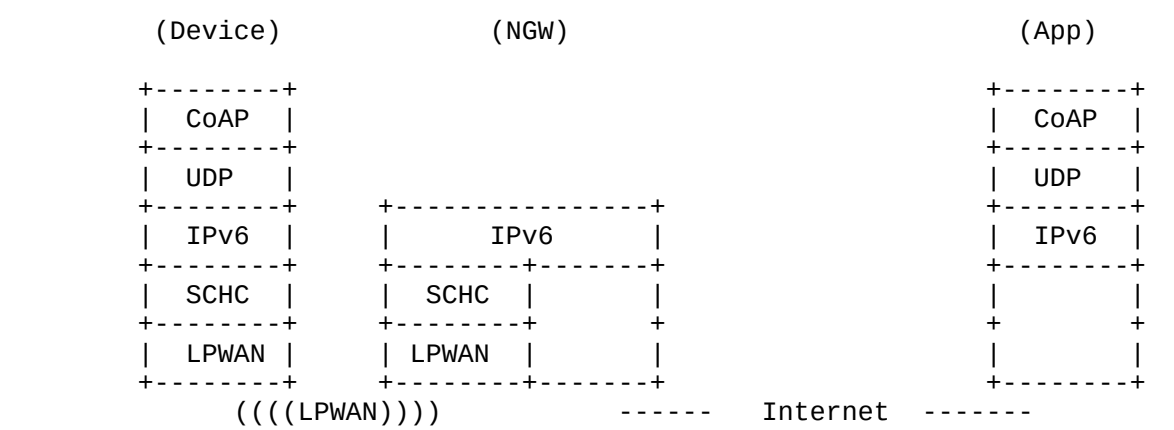


Figure 1: Compression/Decompression at the LPWAN Boundary.

Figure 1 shows the use of SCHC header compression above Layer 2 in the Device and the NGW. The SCHC layer receives non-encrypted packets and can apply compression Rules to all the headers in the stack. On the other end, the NGW receives the SCHC packet and reconstructs the headers using the Rule and the Compression Residue. After the decompression, the NGW forwards the IPv6 packet toward the destination. The same process applies in the other direction when a non-encrypted packet arrives at the NGW. Thanks to the IP forwarding based on the IPv6 prefix, the NGW identifies the Device and compresses headers using the Device’s Rules.

In the second example depicted in Figure 2, SCHC compression is applied in the CoAP layer, compressing the CoAP header independently of the other layers. The RuleID, Compression Residue, and CoAP payload are encrypted using a mechanism such as DTLS [RFC9147]. Only the other end (App) can decipher the information. If needed, layers below use SCHC to compress the header as defined in [RFC8724] (represented by dotted lines in the figure).

This use case needs an end-to-end context initialization between the Device and the application. The context initialization is out of scope for this document.

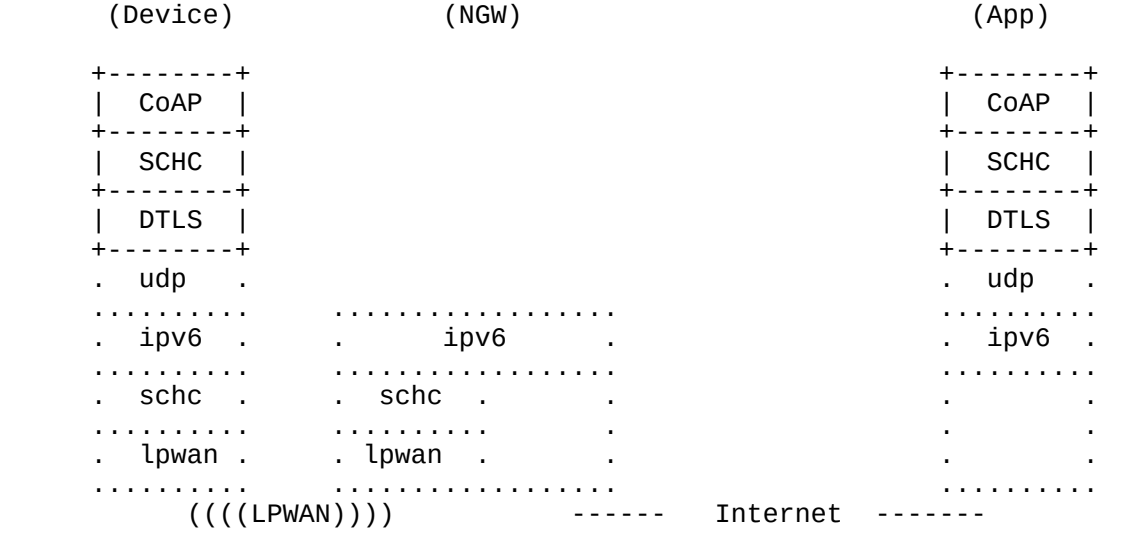


Figure 2: Standalone CoAP End-to-End Compression/Decompression.

The third example depicted in Figure 3 shows the use of the security protocol OSCORE [RFC8613]. In this case, SCHC needs two Rules to compress the CoAP header. A first Rule focuses on the Inner header. The result of this first compression is encrypted using OSCORE. Then, a second Rule compresses the Outer header including the CoAP option OSCORE.

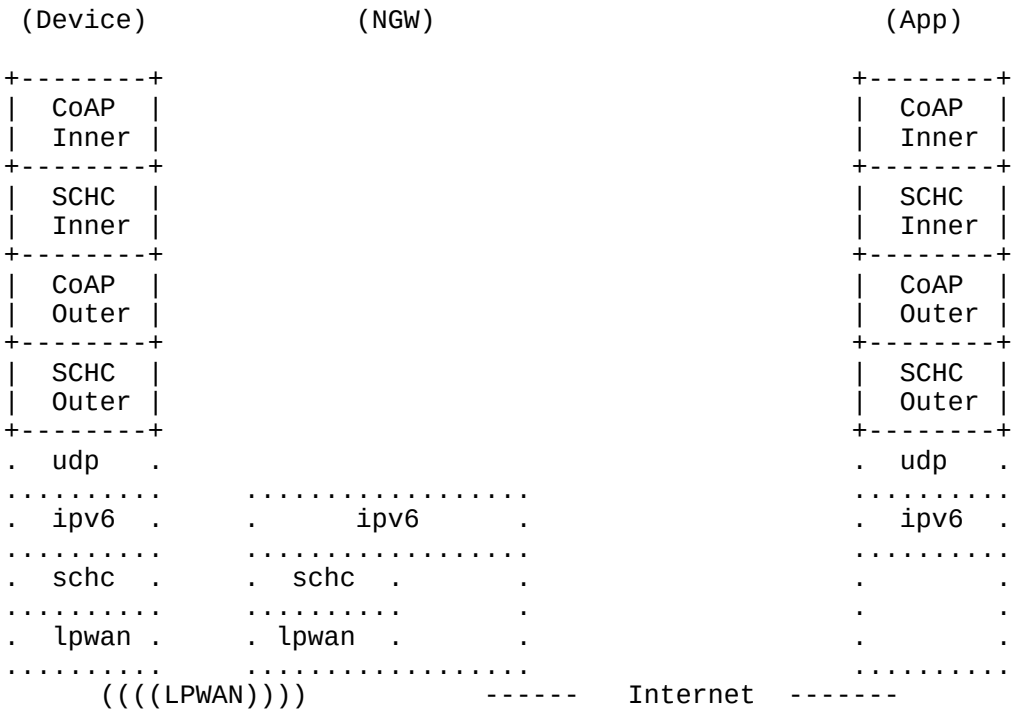


Figure 3: Compression/Decompression when Using OSCORE.

In the case of several SCHC instances as shown in Figure 2 and Figure 3, the Rules may come from different provisioning domains.

This document focuses on CoAP compression, as represented by the dashed boxes in the previous figures.

3. CoAP Headers Compressed with SCHC

The use of SCHC over the CoAP header relies on the same principles and compression/decompression techniques used for IP and UDP headers, as explained in [RFC8724]. For CoAP, the SCHC Rules description uses the direction information to optimize the compression by reducing the number of Rules needed to compress headers. The Field Descriptor MAY define both request/response headers and TVs in the same Rule, using the DI to indicate the header type.

Like for other header compression protocols, when the compressor does not find a correct Rule to compress the header, the packet MUST be sent uncompressed using the RuleID dedicated to this purpose. In such a case, the Compression Residue is the complete header of the packet (see Section 6 of [RFC8724]).

3.1. Differences between CoAP and UDP/IP Compression

CoAP compression differs from IPv6 and UDP compression in the following aspects:

* The CoAP message format is asymmetric, i.e., the headers are different for a request and a response.

  For example, the Uri-Path Option can be used in a request, while it is not used in a response. A request might contain an Accept Option, while both a request and a response might include a Content-Format Option. In comparison, the IPv6 and UDP returning path swaps the value of some fields in the header. However, all the directions have the same fields (e.g., source and destination address fields).

  [RFC8724] defines the use of a DI in the Field Descriptor, which allows a single Rule to process a message header differently, depending on the direction.

* Even when a field is "symmetric" (i.e., found in both directions), the values carried in each direction are different. The compression may use a "match-mapping" MO to limit the range of expected values in a particular direction and reduce the Compression Residue’s size. Through the DI, a Field Descriptor in the Rules splits the possible field value into two parts, one for each direction.

  For instance, if a client sends only Confirmable (CON) requests [RFC7252], the Type can be elided through the compression process, and the reply from the server may use one single bit to carry either the Acknowledgement (ACK) or Reset (RST) type. The field Code has the same behavior: the 0.0X code format value in a request and the Y.ZZ code format in a response.

* In SCHC, the Rule defines the different header fields’ length, so SCHC does not need to send it. In IPv6 and UDP headers, the fields have a fixed size, known by definition.

  On the other hand, some CoAP header fields have variable lengths, and the Rule description specifies it. For example, the size of the Token field may vary from 0 to 8 bytes, and the CoAP options rely on the Type-Length-Value encoding format to specify the size of the actual option value in bytes.

  When doing SCHC compression of a variable-length field, Section 7.4.2 of [RFC8724] makes it possible to define a function for the Field Length in the Field Descriptor, in order to

determine the length before compression. If the Field Length is unknown, the Rule will set it as a variable, and SCHC will send the compressed field's length in the Compression Residue.

* A field can appear several times in a CoAP header. This is typically the case for elements of a URI (i.e., path segments or query parameters). The SCHC specification [RFC8724] allows a FID to appear several times in the Rule and uses the Field Position (FP) to identify the correct instance, thus preventing MO's possible ambiguities.

* Field Lengths defined in CoAP can be too large when it comes to LPWAN traffic constraints. For instance, this is particularly true for the Message ID field and the Token field. SCHC uses different MOs to perform the compression (see Section 7.4 of [RFC8724]). In this case, SCHC can apply the Most Significant Bits (MSBs) MO to reduce the information carried on LPWANs.

4. Compression of CoAP Header Fields

This section discusses the SCHC compression of the CoAP header fields (see Section 3 of [RFC7252]), building on what is specified in Section 7.1 of [RFC8724].

In a SCHC Rule, the first Field Descriptors MUST be those related to the CoAP header fields discussed in this section. In particular, such Field Descriptors MUST be listed in the same order according to which the related CoAP header fields are specified in a CoAP message, i.e.: Version; Type; Token Length; Code; Message ID; and Token (if any). In the rest of this section, those CoAP header fields are discussed according to such an order.

4.1. CoAP Version Field

The Version field is described as bidirectional in a SCHC Rule, and it MUST be elided during SCHC compression, since it always contains the same value. If a new version of CoAP is defined in the future, new Rules will be needed to avoid ambiguities between versions.

4.2. CoAP Type Field

The Type field specifies one of the four types of CoAP messages, encoded as specified in Section 3 of [RFC7252]: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), and Reset (RST).

The SCHC compression scheme SHOULD elide this field if, for instance, a client is sending only NON messages or only CON messages. For RST messages, SCHC may use a dedicated Rule. For other usages, SCHC can use a "match-mapping" MO.

4.3. CoAP Token Length (TKL) Field

The Token Length (TKL) field specifies the size in bytes of the later Token field (see Section 4.6), and is described as bidirectional in a SCHC Rule.

If the field value does not change over time, the SCHC Rule describes the TV set to that value, the MO set to "equal", and the CDA set to "not-sent", thereby eliding the field.

Otherwise, if the field value changes over time, the SCHC Rule does not set the TV, while setting the MO to "ignore" and the CDA to "value-sent". The Rule may also use a "match-mapping" MO to compress the value.

4.4. CoAP Code Field

The Code field takes value from the "Code" column of the "CoAP Codes" IANA registry, encoded as specified in Section 3 of [RFC7252]. This field indicates the Method Code of a CoAP request or the Response Code of a CoAP Response, while the value 0.00 indicates an Empty message. The compression of the CoAP Code field follows the same principle as that of the CoAP Type field.

If the Device plays a specific role, SCHC may split the code values into two Field Descriptors: (1) the Method Codes with the 0 class and (2) the Response Codes. SCHC will then use the DI to identify the correct value in the packet. If the Device only implements a CoAP client, SCHC compression may focus only on the Method Codes that the client uses in its outgoing requests.

For known values, SCHC can use a "match-mapping" MO. If SCHC cannot compress the Code field, it will send the values in the Compression Residue.

4.5. CoAP Message ID Field

SCHC can compress the Message ID field with the MSB MO and the LSB CDA (see Section 7.4 of [RFC8724]).

4.6. CoAP Token Field

A CoAP message fully specifies the Token by using two CoAP fields: the Token Length (TKL) field in the mandatory header (see Section 4.3) and the variable-length Token field that directly follows the mandatory CoAP header and specifies the Token value.

For the Token field, SCHC MUST NOT send it as variable-size data in the Compression Residue. As a result, SCHC does not send the size of the residue resulting from the compression of the Token field, which is otherwise requested for variable-size fields when the CDA specified in the Field Descriptor is "value-sent" or LSB (see Section 7.4.2 of [RFC8724]).

Instead, SCHC MUST use the value of the Token Length field to define the size of the Token field in the Compression Residue. To this end, SCHC designates a specific function, "tkl", that the Rule MUST use to complete the Field Descriptor. During the decompression, this function returns the value contained in the Token Length field, hence the length of the Token field.

This construct avoids ambiguity with the Token Length field and results in a more efficient compression of the Token field.

5. Compression of CoAP Options

CoAP defines the use of options, which are placed after the mandatory header and the Token field and are ordered by option number (see Section 3 of [RFC7252]). As per Section 3.1 of [RFC7252], each option instance in a message relies on a format consisting of an Option Delta (D), an Option Length (L), and an Option Value (V).

The Option Delta is used to express the option number of a CoAP option within a CoAP message, as the difference between the Option Number of that option and the Option Number of the previous option in that message (or zero for the first option). In the byte-representation of CoAP options used on the wire, Option Delta is encoded either by a 4-bit "Option Delta" field or by that field together with an additional 1- or 2-byte "Option Delta (Extended)" field.

The Option Length specifies the length of the Option Value in bytes. In the byte-representation of CoAP options used on the wire, Option Length is encoded either by a 4-bit "Option Length" field or by that field together with an additional 1- or 2-byte "Option Length (Extended)" field.

5.1. Field Descriptors for CoAP Options

In a SCHC Rule, the Field Descriptors related to CoAP options MUST be specified after the Field Descriptors related to the CoAP header fields discussed in Section 4.

In particular, the Field Descriptors related to CoAP options MUST be listed in the same order according to which the corresponding CoAP options appear in the CoAP message (i.e., ordered by option number).

If a SCHC Rule is intended to compress a CoAP message where a repeatable CoAP option is specified multiple times, then the SCHC Rule MUST include different Field Descriptors that separately correspond to the different instances of that CoAP option. Those Field Descriptors MUST be listed in the same order of the corresponding CoAP option instances in the CoAP message.

As further discussed in [I-D.ietf-schc-universal-option], the composition and use of Field Descriptors for compressing/ decompressing CoAP options can take a "syntactic" approach or a "semantic" approach.

The syntactic approach operates faithfully to the byte-representation of CoAP options used on the wire. Consequently, it requires multiple Field Descriptors for each given instance of CoAP option to be compressed/decompressed. That is, each of such Field Descriptors pertains to the compression/decompression of the Option Delta, the Option Length, or the Option Value of the CoAP option in question.

On the contrary, the typically used semantic approach abstracts away from the byte-representation of CoAP options (or, more generally, of protocol header fields) and map those into generic representations identified by the FIDs of the related Field Descriptors. The semantic approach effectively streamlines Field Descriptors related to CoAP options as required to specify only information about the compression/decompression of the Option Value.

The rest of this document refers to the semantic approach, especially when defining the SCHC compression/decompression of CoAP options as well as when providing examples of CoAP header compression.

5.1.1. Option Value

For most CoAP options, the Option Value is a single indivisible field. Consequently, the compression/decompression of the Option Value for such a CoAP option is specified by a Field Descriptor for which the following applies:

* The FID is set to an identifier that unambiguously refers to the CoAP option in question. To this end, the FID provides the following information:

  - An unambiguous identifier of CoAP, as the protocol for which a message is meant to be compressed/decompressed per the present Field Descriptor.

  - The option number of the CoAP option to be compressed/ decompressed per the present Field Descriptor. For registered CoAP options, the value is taken from the "Number" column of the corresponding entry in the "CoAP Option Numbers" IANA registry [CoAP.Option.Numbers].

  For example, the FID can be set to "CoAP.option(3)" in a Field Descriptor related to the CoAP Uri-Host Option (see Section 5.3).

* The FL represents the Option Length L of the CoAP option encoded as per Section 7.1 of [RFC8724].

* The TV is either set to an appropriate value (e.g., the Option Value V of the CoAP option) or not set, consistently with the intent of the SCHC Rule and with the MO and CDA used in the Field Descriptor.

For some CoAP options, it might be possible and more convenient for SCHC to consider the Option Value as composed of distinct subfields. An example is the CoAP OSCORE Option defined in [RFC8613] and for which the SCHC compression/decompression is defined in Section 6.4 of this document. Instead of pertaining to the SCHC compression/ decompression of the Option Value as a whole, a Field Descriptor related to such a CoAP option can instead specifically pertain to the SCHC compression/decompression of one subfield of the Option Value. In this case, the Field Descriptors related to different subfields of the Option Value of a given CoAP option MUST be listed in the same order according to which the corresponding subfields appear in the Option Value. Furthermore, the following applies to each of such Field Descriptors:

* The FID is set to an identifier that unambiguously refers to the CoAP option in question and to the subfield to be compressed/ decompressed. To this end, in addition to the two pieces of information mentioned above for the previous case, the FID further provides the following information:

  - An unambiguous identifier of the subfield of the Option Value of the CoAP option to be compressed/decompressed per the present Field Descriptor.

For example, the FID can be set to "CoAP.option(9).flags" in a Field Descriptor pertaining to the "flags" subfield of the Option Value of the CoAP OSCORE Option (see Section 6.4).

* The FL either represents the length of the subfield encoded as per Section 7.1 of [RFC8724] or denotes a designated function to compute that length.

* The TV is either set to an appropriate value (e.g., the value of the subfield) or not set, consistently with the intent of the SCHC Rule and with the MO and CDA used in the Field Descriptor.

Note that the MO and the CDA specified in the Field Descriptor operates only on the (subfield of the) Option Value V. That is, SCHC compression produces a residue from the (subfield of the) Option Value V, while ignoring the option number, the Option Delta, and the Option Length or the length of the Option Value's subfield. Therefore, the residue of a SCHC packet conveying a compressed CoAP header does not include the option number, the Option Delta, and the Option Length of the compressed CoAP options. The recipient will be able to reconstruct those when performing SCHC decompression, leveraging the FID and FL of the Field Descriptors within the SCHC Rule used.

When the Option Length or the length of the Option Value's subfield has a well-known value, the Rule may specify that information in the FL of the Field Descriptor (see above). In such a case, SCHC compression treats the (subfield of the) Option Value as a fixed-length field (see Section 7.4.1 of [RFC8724]).

Otherwise, the Rule specifies the FL of the Field Descriptor as indicating a variable length and SCHC compression treats the (subfield of the) Option Value as a variable-length field (see Section 7.4.2 of [RFC8724]). In such a case, when the CDA specified in the Field Descriptor is "value-sent" or LSB, then SCHC compression additionally carries the length of the Compression Residue, as prepended to the Compression Residue value. Note that the length coding differs between CoAP options and the Compression Residue of SCHC variable-length fields.

CoAP requests and responses do not include the same options. Compression Rules may reflect this asymmetry by using the DI.

The following sections present how SCHC compresses some specific CoAP options. Unless otherwise indicated, the referred CoAP options are specified in [RFC7252].

If the use of an additional CoAP option is later introduced, the SCHC Rules MAY be updated, in which case a new FID description MUST be assigned to perform the compression of the CoAP option. Otherwise, if no Rule describes that CoAP option, SCHC compression is not achieved and SCHC sends the CoAP header without compression.

5.2. CoAP Option Content-Format and Accept Fields

If the client expects a single specific value, SCHC can elide these fields, by specifying the value in the TV of a Rule description with an "equal" MO and a "not-sent" CDA.

Otherwise, if the client expects several possible values, a "match-mapping" MO SHOULD be used to limit the Compression Residue's size. If not, SCHC has to send the Option Value in the Compression Residue (with fixed or variable length).

5.3. CoAP Option Max-Age, Uri-Host, and Uri-Port Fields

SCHC compresses these three fields in the same way. When the values of these options are known, SCHC can elide these fields. If the option uses well-known values, SCHC can use a "match-mapping" MO.

Otherwise, these options' values will be sent in the Compression Residue, i.e., the SCHC Rule description does not set the TV, while setting the MO to "ignore" and the CDA to "value-sent".

5.4. CoAP Option Uri-Path and Uri-Query Fields

The Uri-Path and Uri-Query fields are repeatable options, i.e., the CoAP header may include them several times and with different values. The SCHC Rule description uses the FP to distinguish the different instances of such options.

To compress these repeatable field values, SCHC can use a "match-mapping" MO to reduce the size of variable paths or queries. When doing so, several elements can be regrouped into a single entry in order to optimize the compression. The numbering of elements does not change, and the first matching element sets the MO comparison.

For example, as per the Rule descriptions shown in Table 1, SCHC can use a single bit in the Compression Residue to code the path segments "/a/b" or the path segments "/c/d". If regrouping were not allowed, then 2 bits in the Compression Residue would be needed. At the same time, SCHC sends the third path element following "/a/b" or "/c/d" as a variable-size field in the Compression Residue.

FID	FL	FP	DI	TV	MO	CDA
CoAP. option(11)		1	Up	["/a/ b", "/c/ d"]	match-mapping	mapping-sent
CoAP. option(11)	var (B)	3	Up		ignore	value-sent

Table 1: Complex Path Example. CoAP Option Numbers: 11 (Uri-Path).

The length of the Uri-Path and Uri-Query Options may be known when the Rule is defined. In any other case, SCHC MUST set the Field Length (FL) to a variable value. The unit of the variable length is bytes, hence the Compression Residue size is expressed in bytes, encoded as defined in Section 7.4.2 of [RFC8724].

SCHC compression can use the MSB MO for a Uri-Path or Uri-Query element. In such a case, care must be taken when specifying the MSB parameter value in bits, which MUST be a multiple of 8. The length sent at the beginning of the variable-size field Compression Residue indicates the LSB’s size in bytes, consistent with the unit of the variable length in the Rule description.

For instance, for a CORECONF path /c/X6?k=eth0, the Rule description can be as shown in Table 2. That is, SCHC compresses the first part of the URI path with a "not-sent" CDA. Also, SCHC will send the second element of the URI path preceded by the length (i.e., 0b0010 "X6"), which is followed by the query parameter’s value preceded by the length (i.e., 0b0100 "eth0").

FID	FL	FP	DI	TV	MO	CDA
CoAP. option(11)		1	Up	"c"	equal	not-sent
CoAP. option(11)	var (B)	2	Up		ignore	value-sent
CoAP. option(15)	var (B)	1	Up	"k="	MSB(16)	LSB

Table 2: CORECONF URI compression. CoAP Option Numbers: 11 (Uri-Path), 15 (Uri-Query).

5.4.1. Variable Number of Path or Query Elements

SCHC fixes the number of Uri-Path or Uri-Query elements in a Rule at Rule creation time. If the number of such elements varies, SCHC SHOULD either:

* create several Rules to cover all possibilities; or

* create a Rule that defines several entries for Uri-Path to cover the longest path and send a Compression Residue with a length of 0 to indicate that a Uri-Path entry is empty.

  However, this adds 4 bits to the variable Compression Residue size (see Section 7.4.2 of [RFC8724]).

5.5. CoAP Option Size1, Size2, Proxy-Uri, and Proxy-Scheme Fields

The Size2 field is an option defined in [RFC7959].

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

5.6. CoAP Option Proxy-Cri and Proxy-Scheme-Number Fields

The Proxy-Cri field is an option defined in [I-D.ietf-core-href]. The option carries an encoded CBOR data item [RFC8949] that represents an absolute CRI reference (see Section 5 of [I-D.ietf-core-href]). The option is used analogously to the Proxy-Uri option as defined in Section 5.10.2 of [RFC7252].

The Proxy-Scheme-Number field is an option defined in [I-D.ietf-core-href]. The option carries a CRI Scheme Number represented as a CoAP unsigned integer (see Sections 5.1.1 and 8.1 of [I-D.ietf-core-href]). The option is used analogously to the Proxy-Scheme option as defined in Section 5.10.2 of [RFC7252].

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

5.7. CoAP Location-Path and Location-Query Fields

A Rule entry cannot store these fields’ values. Therefore, SCHC compression MUST always send these values in the Compression Residue. That is, in the SCHC Rule, the TV is not set, while the MO is set to "ignore" and the CDA is set to "value-sent".

5.8. CoAP Option ETag and If-Match Fields

When a CoAP message uses the ETag Option or the If-Match Option, SCHC compression MAY send its content in the Compression Residue. That is, in the SCHC Rule, the TV is not set, while the MO is set to "ignore" and the CDA is set to "value-sent". Alternatively, if a pre-defined set of values determined by the server is known and is used by the client as ETag values or If-Match values, then a Rule MAY use a "match-mapping" MO when there are different alternatives for the same FID.

5.9. CoAP Option If-None-Match

The If-None-Match 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".

5.10. 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 CoAP message uses the Hop-Limit Option, SCHC compression SHOULD send its content in the Compression Residue. That is, in the SCHC Rule, the TV is not set, while the MO is set to "ignore" and the CDA is set to "value-sent". As an exception, and consistently with the default value 16 defined for the Hop-Limit Option in Section 3 of [RFC8768], a Rule MAY describe a TV with value 16, with the MO set to "equal" and the CDA set to "not-sent".

5.11. CoAP Option Echo Field

The Echo field is an option defined in [RFC9175] that a server can include in a CoAP response as a challenge to the client, and that the client echoes back to the server in one or more CoAP 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 CoAP message uses the Echo Option, SCHC compression SHOULD send its content in the Compression Residue. That is, in the SCHC Rule, the TV is not set, while the MO is set to "ignore" and the CDA is set to "value-sent". An exception applies in case the server generates the values to use for the Echo Option by means of a persistent counter (see Appendix A of [RFC9175]). In such a case, a Rule MAY use the MSB MO and the LSB CDA. This would be effectively applicable until the persistent counter at the server becomes greater than the maximum threshold value that produces an MSB-matching.

5.12. CoAP Option Request-Tag Field

The Request-Tag field is an option defined in [RFC9175] that the client can set in CoAP requests throughout 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 CoAP message uses the Request-Tag Option, SCHC compression MAY send its content in the Compression Residue. That is, in the SCHC Rule, the TV is not set, while the MO is set to "ignore" and the CDA is set to "value-sent". Alternatively, if a pre-defined set of Request-Tag values used by the client is known, a Rule MAY use a "match-mapping" MO when there are different alternatives for the same FID.

5.13. CoAP Option EDHOC Field

The EDHOC field is an option defined in [RFC9668] that a client can include in a CoAP request, in order to perform an optimized, shortened execution of the authenticated key exchange protocol EDHOC [RFC9528]. 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 EDHOC 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".

6. Compression of CoAP Extensions

The following sections present how SCHC compresses some specific CoAP options that, when used, play a major role in the processing and exchange of CoAP messages.

6.1. Block-Wise Transfers

When a CoAP message 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. In the SCHC Rule, the TV is not set, while the MO is set to "ignore" and the CDA is 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.

6.2. Observe

[RFC7641] defines the Observe Option. The SCHC Rule description does not set the TV, while setting the MO to "ignore" and the CDA to "value-sent". SCHC does not limit the maximum size for this option (3 bytes). To reduce the transmission size, either the Device implementation MAY limit the delta between two consecutive values or a proxy can modify the increment.

Since the client MAY use a RST message to inform a server that the Observe response is not required, a specific SCHC Rule SHOULD exist to allow the compression of a RST message.

6.3. No-Response

[RFC7967] defines a No-Response Option limiting the CoAP responses made by a server to a CoAP request. Different behaviors exist while using this option to limit the responses made by a server to a request. If both ends know the specific value, then the SCHC Rule describes the TV set to that value, the MO set to "equal", and the CDA set to "not-sent".

Otherwise, if the value changes over time, the SCHC Rule does not set the TV, while setting the MO to "ignore" and the CDA to "value-sent". The Rule may also use a "match-mapping" MO to compress the value.

6.4. 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.

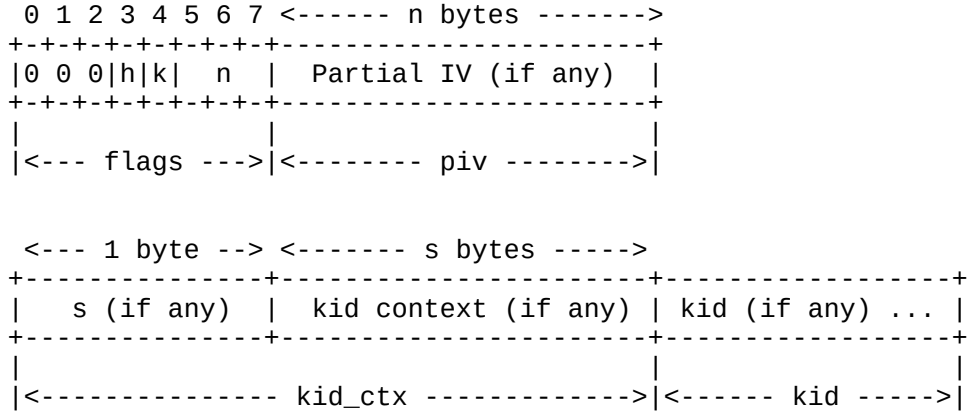
Figure 4 shows the OSCORE Option value encoding, as it was originally defined in Section 6.1 of [RFC8613]. As explained later in this section, this has been extended in [I-D.ietf-core-oscore-key-update] and [I-D.ietf-core-oscore-groupcomm]. The first byte of the OSCORE Option value specifies information to parse the rest of the value by using flags, as described below.

* 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 value 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 OSCORE Option value. Also, bit k, when set, indicates the presence of the kid field. Finally, the three least significant bits form the n field, which indicates the length of the Partial IV (Partial Initialization Vector) field in bytes. When n = 0, no Partial IV is present.

Assuming the presence of a single flag byte, this is followed by the Partial IV field. After that, if the h bit is set, the kid context field is present, preceded by one byte "s" indicating its length in bytes. After that, if the k bit is set, the kid field is present, and it ends where the OSCORE Option value ends.



0	0	0	h	k	n	Partial IV (if any)

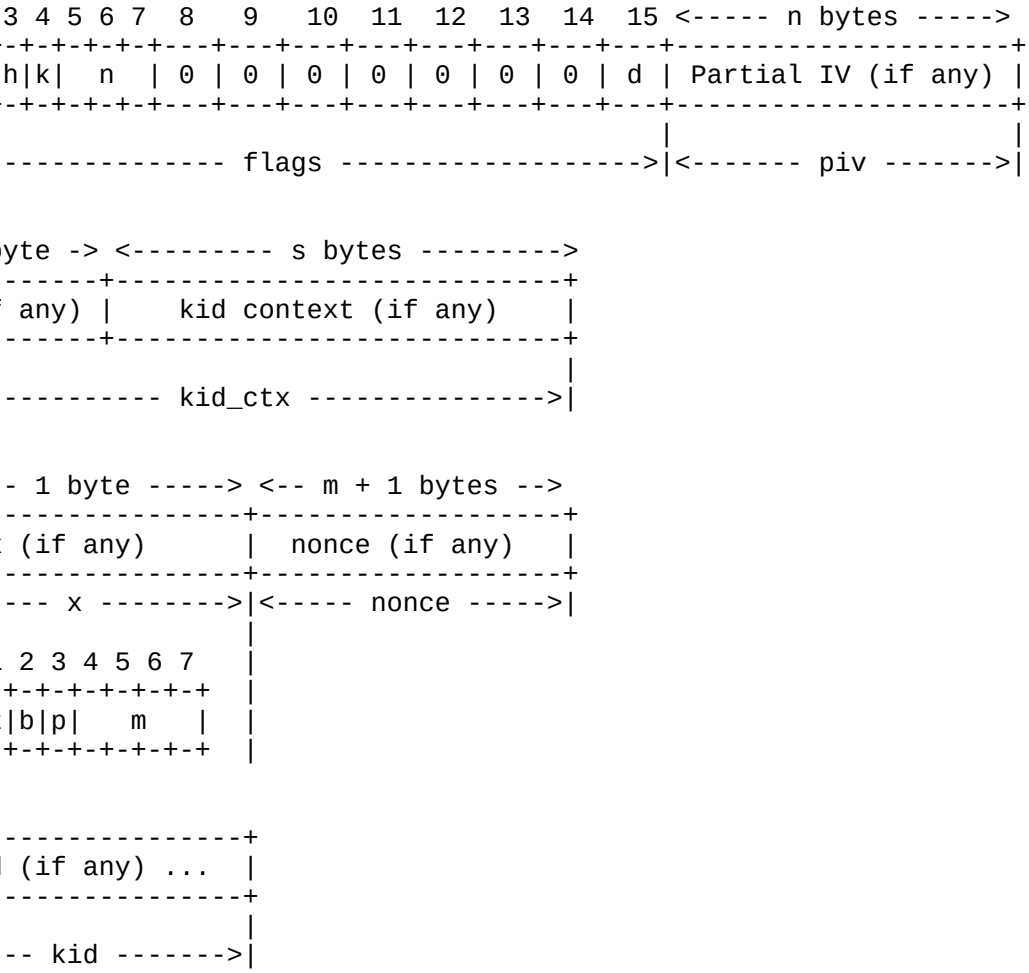
s (if any)	kid context (if any)	kid (if any) ...

Figure 4: OSCORE Option Value.

Figure 5 shows the extended 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 OSCORE Option value, 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 further information concerning the KUDOS execution in question.

Figure 5 provides the breakdown of the x field, where its four least significant bits encode the value m, which specifies the size of nonce in bytes, minus 1.



To better perform OSCORE SCHC compression, the Rule description needs to identify the OSCORE Option value and its inner fields mentioned above.

Conceptually, SCHC discerns six distinct pieces of information within the OSCORE Option value: the flag bits, the Partial IV, the kid context prepended by its size s, the x byte, the nonce, and the kid. The SCHC Rule splits the OSCORE Option value into six corresponding Field Descriptors, in order to separately compress those pieces of information as distinct subfields:

* flags
* piv
* kid_ctx
* x
* nonce
* kid

If a SCHC Rule is intended to compress a CoAP message that specifies the OSCORE Option, then the related Field Descriptors defined above MUST be listed in the same order according to which the corresponding pieces of information appear in the OSCORE Option value.

Figure 4 shows the original format of the OSCORE Option value with the four subfields flags, piv, kid_ctx, and kid superimposed on it. Also, Figure 5 shows the extended format of the OSCORE Option value with all the six subfields superimposed on it.

If a subfield is not present, then the corresponding Field Descriptor in the SCHC Rule describes the TV set to b'', with the MO set to "equal" and the CDA set to "not-sent". Note that, if the subfield kid_context is present, it directly includes the size octet, i.e., s.

In addition, the following applies.

* If the piv subfield is present, SCHC MUST NOT send it as variable-size data in the Compression Residue. As a result, SCHC does not send the size of the residue resulting from the compression of the piv subfield, which is otherwise requested for variable-size fields when the CDA specified in the Field Descriptor is "value-sent" or LSB (see Section 7.4.2 of [RFC8724]).

Instead, SCHC MUST use the value n from the first byte of the OSCORE Option value to define the size of the piv subfield in the Compression Residue. To this end, SCHC designates a specific function, "osc.piv", that the Rule MUST use to complete the Field Descriptor. During the decompression, this function returns the value n, hence the length of the piv subfield.

This construct avoids ambiguity with the value n from the first byte of the OSCORE Option value and results in a more efficient compression of the piv subfield.

* For the x subfield, if both endpoints know the value, then the corresponding Field Descriptor in the SCHC Rule describes the TV set to that value, with the MO set to "equal" and the CDA set to "not-sent". This models the following cases:

  - The x subfield is not present, and thus TV is set to b''.

  - Given a fixed z bit of the x subfield as denoting either a "divergent" or "convergent" KUDOS message, the two endpoints run KUDOS with a pre-agreed size of the nonce subfield as per the value encoded by m within the x subfield, as well as with a pre-agreed combination of its modes of operation, as per the bits b and p of the x subfield.

    Under the assumed pre-agreements above, this requires two distinct SCHC Rules, whose respective TV is set to a value that reflects the z bit as set or not set, respectively.

  As an alternative that is more flexible to changes in the value of the x subfield, the corresponding Field Descriptor in the SCHC Rule does not set the TV, while it sets the MO to "ignore" and the CDA to "value-sent". In the same spirit, the Rule may also use a "match-mapping" MO to compress this subfield, in case the two endpoints pre-agree on a set of alternative ways to run KUDOS, with respect to the size of the nonce subfield and the combination of the KUDOS modes of operation to use.

* If the nonce subfield is present, then the corresponding Field Descriptor in the SCHC Rule has the TV not set, while the MO is set to "ignore" and the CDA is set to "value-sent".

For the value of the nonce subfield, SCHC MUST NOT send it as variable-length data in the Compression Residue. As a result, SCHC does not send the size of the residue resulting from the compression of the nonce subfield, which is otherwise requested for variable-size fields when the CDA specified in the Field Descriptor is "value-sent" or LSB (see Section 7.4.2 of [RFC8724]).

Instead, SCHC MUST use the value encoded by m within the x subfield 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 subfield in bytes, as the value encoded by m within the x subfield, plus 1.

This construct avoids ambiguity with the value m within the x subfield and results in a more efficient compression of the nonce subfield.

7. Compression of the CoAP Payload Marker

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 in a SCHC Rule description.

If the CoAP message to compress with SCHC is not going to be protected with OSCORE [RFC8613] 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.

If the CoAP message to compress with SCHC is going to be protected with OSCORE, the 0xFF payload marker is compressed as specified later in Section 8.2.

8. Examples of CoAP Header Compression

8.1. Mandatory Header with CON Message

In this first scenario, the SCHC compressor on the NGW side receives a POST message from an Internet client, which is immediately acknowledged by the Device. Table 3 describes the SCHC Rule descriptions for this scenario.

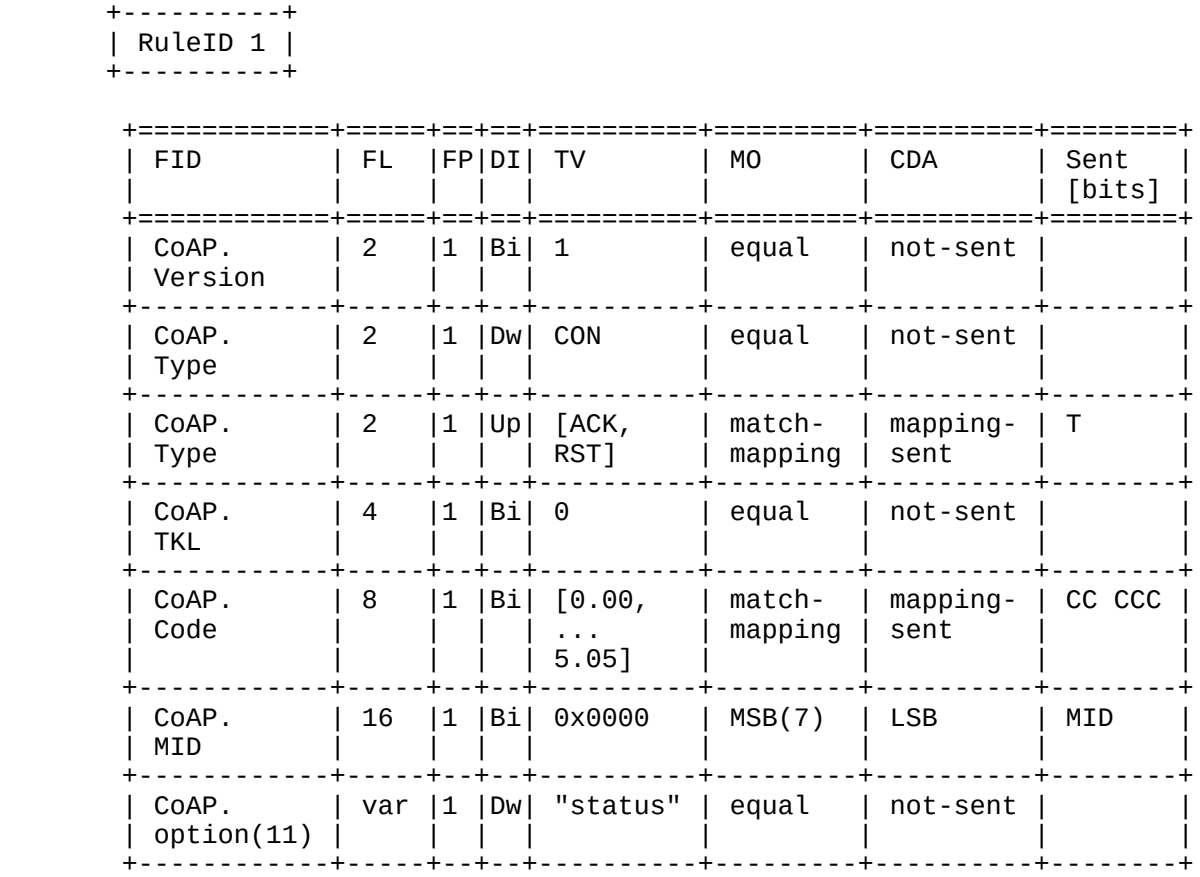


Table 3: CoAP Context to compress header without Token. CoAP Option Numbers: 11 (Uri-Path).

In this example, SCHC compression elides the version and Token Length fields. The 25 Method and Response Codes defined in [RFC7252] have been shrunk to 5 bits using a "match-mapping" MO. The Uri-Path contains a single element with the TV set to "status" and the CDA set to "not-sent", thereby eliding the single occurrence of the Uri-Path Option with value "status".

SCHC compression reduces the header, sending only a mapped Type (and only for uplink messages), a mapped code, and the least significant bits of the Message ID (9 bits in the example above).

Note that, if a client is located in an Application Server and sends a request to a server located in the Device, then the request may not be compressed through this Rule, since the MID might not start with 7 bits equal to 0. A CoAP proxy placed before SCHC C/D can rewrite the Message ID to fit the value and match the Rule.

8.2. OSCORE Compression

The OSCORE security protocol specified in [RFC8613] provides end-to-end protection for CoAP messages. When doing so, it hides as much as possible of a CoAP message, while still enabling proxy operations.

Conceptually, this is achieved by splitting the CoAP message into an Inner Plaintext and an Outer OSCORE message. The Inner Plaintext contains (sensitive) information that is not necessary for performing proxy operations. Therefore, that information can be encrypted end-to-end until it reaches the other origin endpoint as its final destination. The Outer Message acts as a shell matching the regular CoAP message format, and includes all the CoAP options and information needed for performing proxy operations and caching. This is summarized in Figure 6.

In particular, the CoAP options are arranged into three classes, each of which is granted a specific type of protection by the OSCORE protocol:

* Class E: Encrypted and integrity-protected options, which are moved to the Inner Plaintext.

* Class I: Integrity-protected options, which are included in the Additional Authenticated Data (AAD) when protecting the Plaintext, but are otherwise left untouched in the Outer Message.

* Class U: Unprotected options, which are left untouched in the Outer Message.

As per these classes, the Outer options comprise the OSCORE Option, which indicates that the message is protected with OSCORE and carries the information necessary for the recipient endpoint to retrieve the Security Context for decrypting the message.

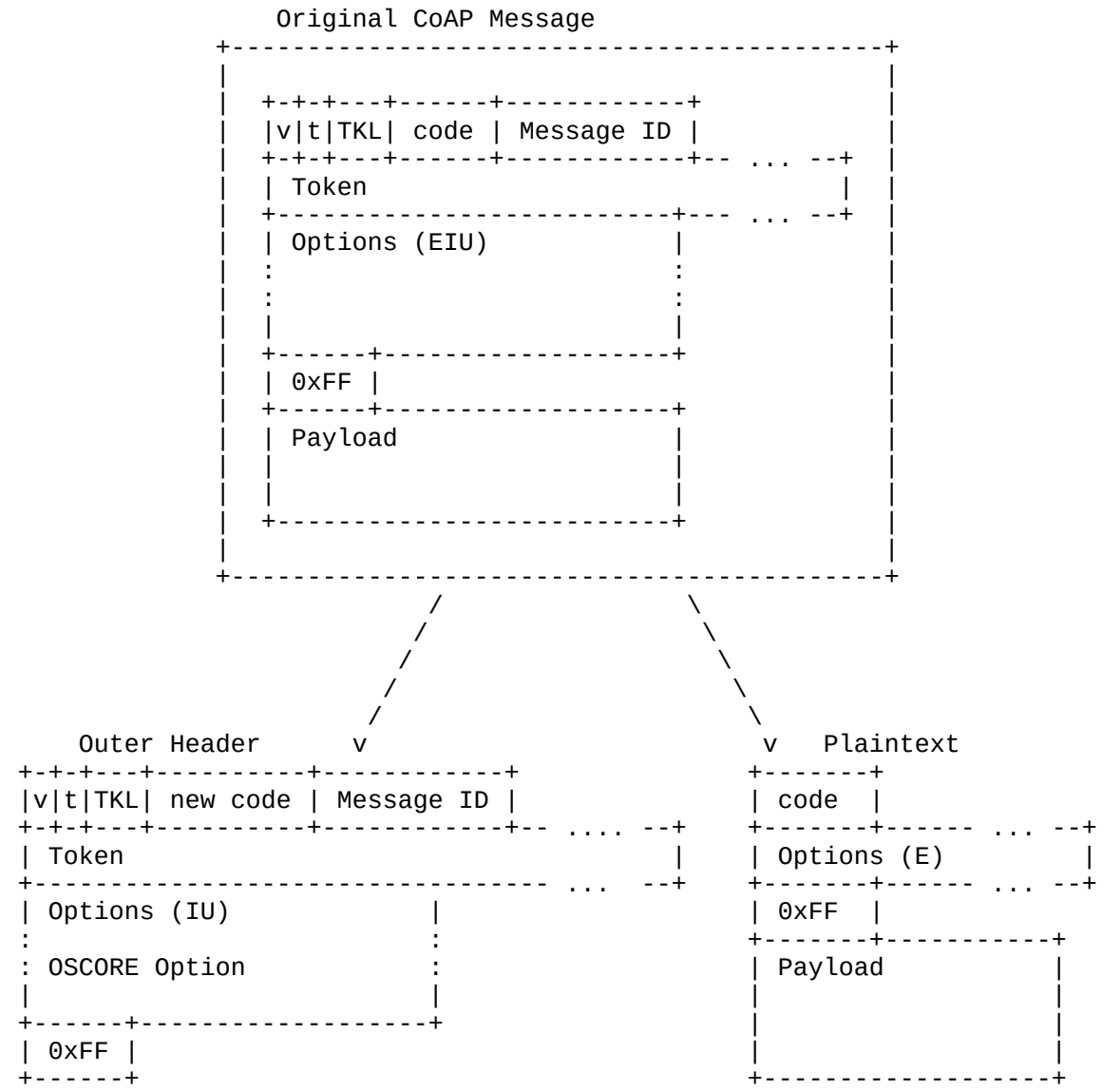


Figure 6: CoAP Message Split into OSCORE Outer Header and Plaintext.

Figure 6 shows the packet format for the OSCORE Outer header and Plaintext.

In the Outer header, the original header code is hidden and replaced by a well-known value. As specified in Sections 4.1.3.5 and 4.2 of [RFC8613], the original header code is replaced with POST for

requests and Changed for responses, when the message does not include the Observe Option. Otherwise, the original header code is replaced with FETCH for requests and Content for responses.

The first byte of the Plaintext contains the original header code, the Class E options, and, if present, the original message payload preceded by the payload marker.

After that, an Authenticated Encryption with Associated Data (AEAD) algorithm encrypts the Plaintext. This also integrity-protects the Security Context parameters and, if present, any Class I options from the Outer header. The resulting ciphertext becomes the new payload of the OSCORE message, as illustrated in Figure 7.

As defined in [RFC5116], this ciphertext is the encrypted Plaintext's concatenation with the Authentication Tag. Note that Inner Compression only affects the Plaintext before encryption. The Authentication Tag, fixed in length and uncompressed, is considered part of the cost of protection.

When the CoAP message is specifically protected with the group mode of Group OSCORE (see Section 8 of [I-D.ietf-core-oscore-groupcomm]), the ciphertext is followed by a signature, which is computed over the ciphertext and additional authenticated data. That is, in the message protected with Group OSCORE, the CoAP payload includes the ciphertext concatenated with the signature. This has no impact on the SCHC compression/decompression. That is, like in any other case, the CoAP payload of the protected message is sent as-is within the SCHC packet, following the Compression Residue (if any).

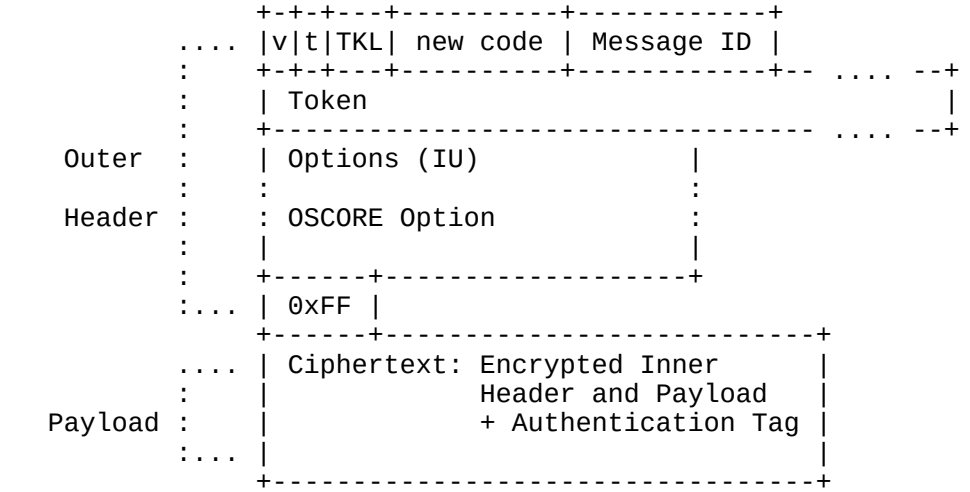


Figure 7: OSCORE Message.

The SCHC compression scheme consists of compressing both the Plaintext before encryption and the resulting OSCORE message after encryption, as shown in Figure 8. Note that, since the recipient endpoint can only decrypt the Inner part of the message, that endpoint will also have to implement Inner SCHC Compression/ Decompression.

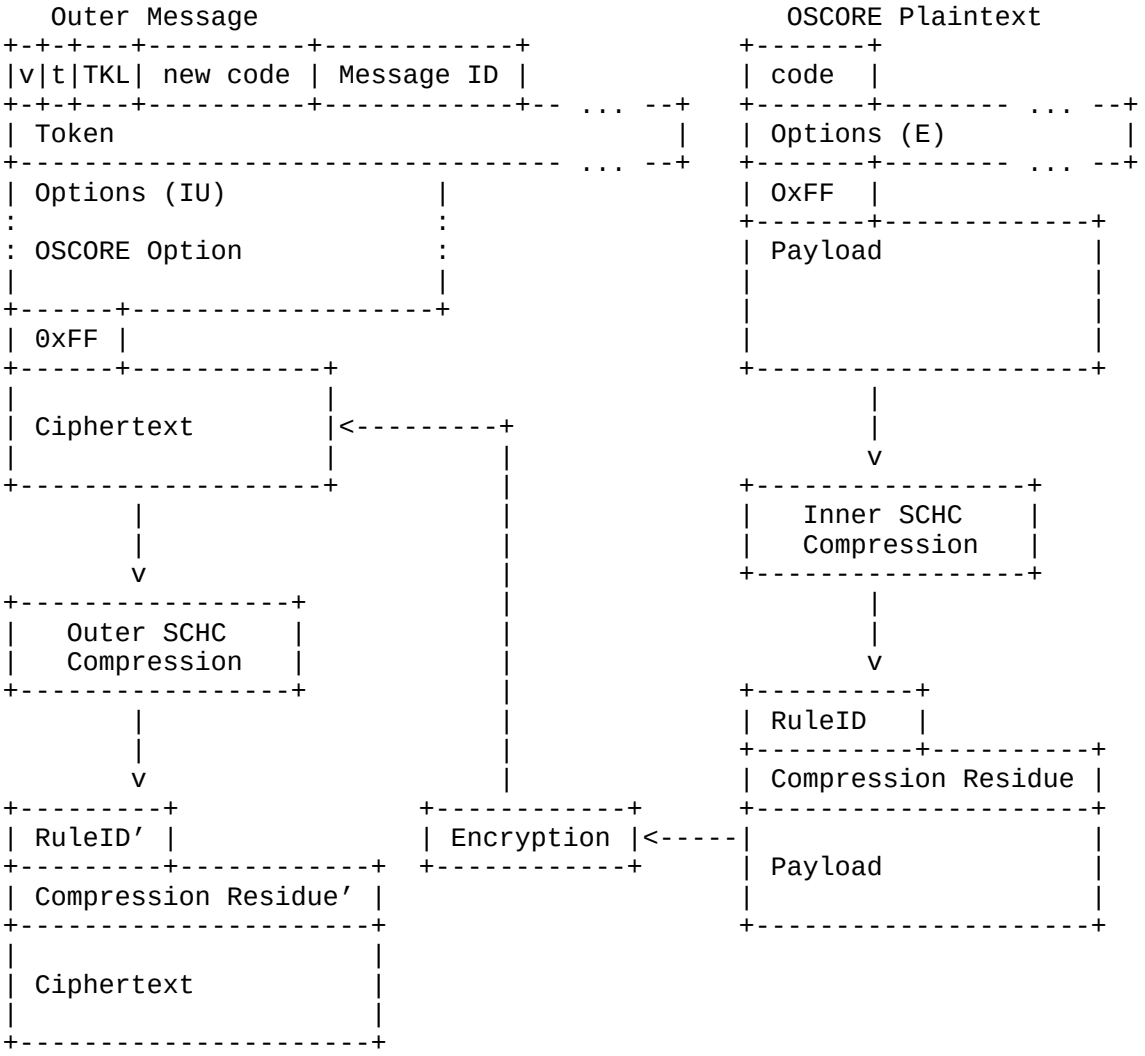


Figure 8: OSCORE Compression Diagram.

The OSCORE message translates into a segmented process where SCHC compression is applied independently in two stages, each with its corresponding set of Rules, i.e., the Inner SCHC Rules and the Outer SCHC Rules. By doing so, compression is applied to all the fields of the original CoAP message.

As to the compression of the 0xFF payload marker, the same rationale described in Section 7 applies to both the Inner SCHC Compression and the Outer SCHC Compression. 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.

8.3. Example OSCORE Compression

This section provides an example with a GET request and a corresponding Content response, exchanged between a Device-based CoAP client and a cloud-based CoAP server. The example also describes a possible set of Rules for Inner SCHC Compression and Outer SCHC Compression. A dump of the results and a contrast between SCHC + OSCORE performance and SCHC + CoAP performance are also listed. This example shows an estimate of the cost of security with SCHC-OSCORE.

Our first CoAP message is the GET request in Figure 9.

```
Original message:
=================
0x4101000182bb74656d7065726174757265

Header:
0x4101
01   Ver
  00   CON
    0001   TKL
        00000001   Request Code 1 "GET"

0x0001 = mid
0x82 = token

Options:

0xbb74656d7065726174757265
Option 11: Uri-Path
Value = temperature

Original message length: 17 bytes
```

Figure 9: CoAP GET Request.

Its corresponding response is the Content response in Figure 10.

```
Original message:
=================
0x6145000182ff32332043

Header:
0x6145
01   Ver
  10   ACK
    0001   TKL
        01000101 Successful Response Code 69 "2.05 Content"

0x0001 = mid
0x82 = token

0xFF  Payload marker

Payload:
0x32332043

Original message length: 10 bytes
```

Figure 10: CoAP Content Response.

The SCHC Rules for the Inner Compression include all the fields that are present in a regular CoAP message. The methods described in Section 4 apply to these fields. Table 4 provides an example.

RuleID 0

FID	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP. Code	8	1	Up	1	equal	not-sent	
CoAP. Code	8	1	Dw	[69, 132]	match-mapping	mapping-sent	C
CoAP. option(11)		1	Up	"temperature"	equal	not-sent	

Table 4: Inner SCHC Rule. CoAP Option Numbers: 11 (Uri-Path).

Figure 11 shows the Plaintext obtained for the example GET request. The packet follows the process of Inner Compression and encryption until the payload. The Outer OSCORE message adds the result of the Inner process.

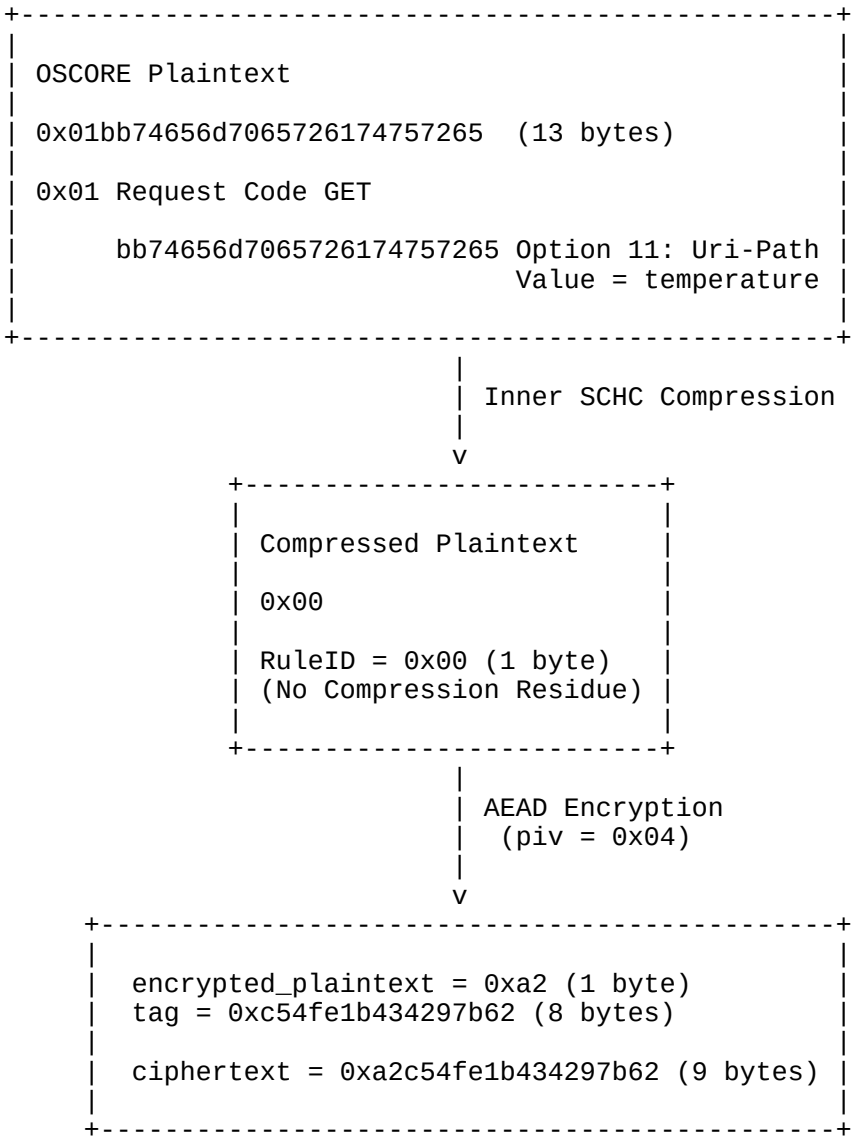


Figure 11: Plaintext Compression and Encryption for GET Request.

In this case, the original message has no payload and its resulting Plaintext is compressed up to only 1 byte (the size of the RuleID). The AEAD algorithm preserves this length in its first output and yields a fixed-size tag. SCHC cannot compress the tag, and the OSCORE message must include it without compression. The use of integrity protection translates into an overhead on the total message

length, thus limiting the amount of compression that can be achieved and contributing to the cost of applying security to the message exchange.

Figure 12 shows the process for the example Content response. The Compression Residue is 1 bit long. Note that since SCHC adds padding after the payload, this misalignment causes the hexadecimal code from the payload to differ from the original, even if SCHC cannot compress the tag. The overhead for the tag bytes limits SCHC’s performance but adds security to the message exchange.

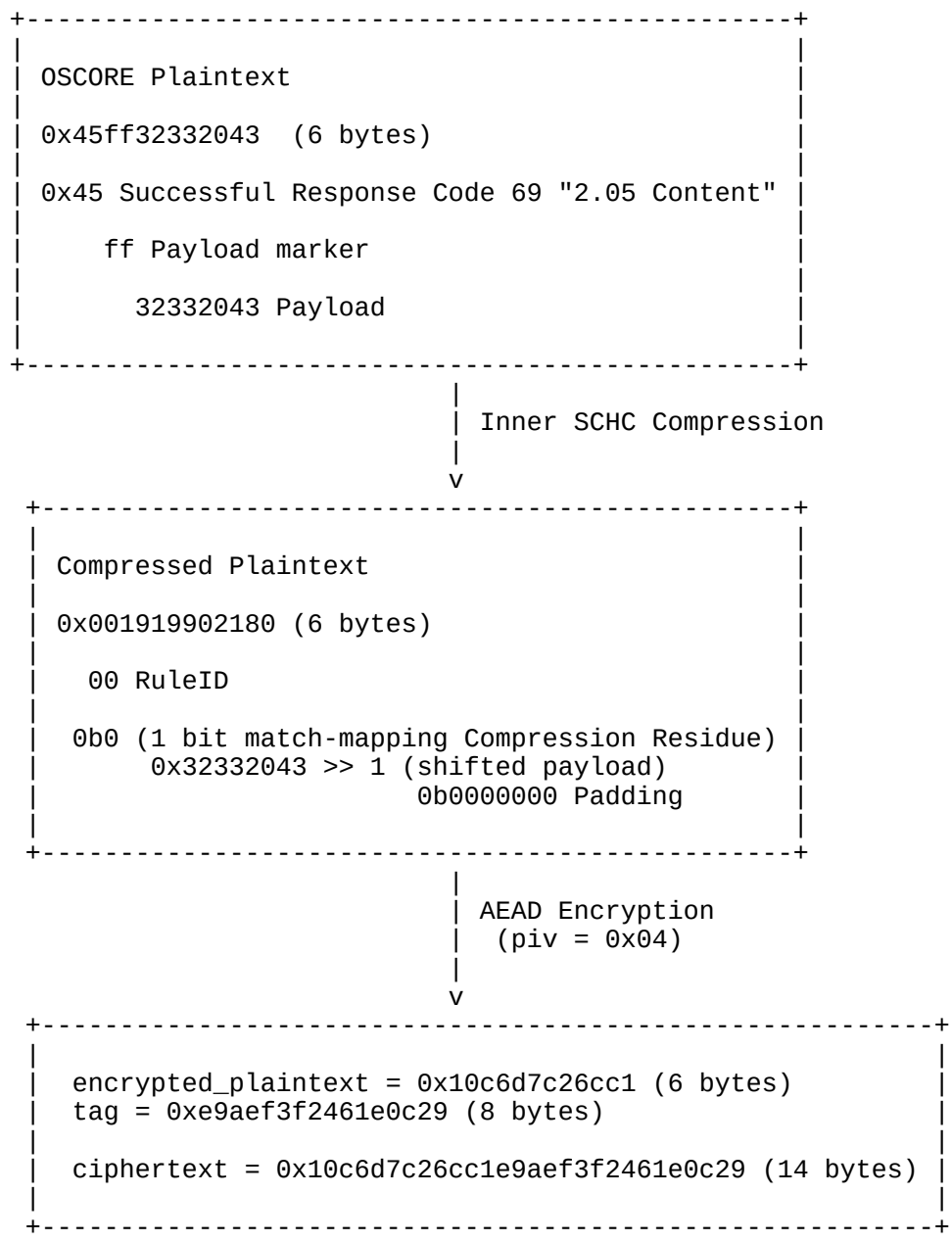


Figure 12: Plaintext Compression and Encryption for Content Response

The Outer SCHC Rule shown in Table 5 is used, also to process the OSCORE Option fields. Figure 13 and Figure 14 show a dump of the OSCORE messages generated from the example messages, also including the Inner Compressed ciphertext in the payload. These are the messages that have to be compressed via the Outer SCHC Compression scheme.

Table 5 shows a possible set of Outer Rule items to compress the Outer header.

RuleID 1

FID	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP. Version	2	1	Bi	1	equal	not-sent	
CoAP. Type	2	1	Up	0	equal	not-sent	
CoAP. Type	2	1	Dw	2	equal	not-sent	
CoAP. TKL	4	1	Bi	1	equal	not-sent	
CoAP. Code	8	1	Up	2	equal	not-sent	
CoAP. Code	8	1	Dw	68	equal	not-sent	
CoAP. MID	16	1	Bi	0x0000	MSB(12)	LSB	MMMM
CoAP. Token	tkl	1	Bi	0x80	MSB(5)	LSB	TTT
CoAP. option(9). flags	var	1	Up	0x09	equal	not-sent	
CoAP. option(9).	var	1	Dw	b''	equal	not-sent	

flags							
CoAP. option(9). piv	osc.piv	1	Up	0x00	MSB(4)	LSB	PPPP
CoAP. option(9). piv	var	1	Dw	b''	equal	not-sent	
CoAP. option(9). kid_ctx	var	1	Bi	b''	equal	not-sent	
CoAP. option(9). x	8	1	Bi	b''	equal	not-sent	
CoAP. option(9). nonce	osc.x.m	1	Bi	b''	equal	not-sent	
CoAP. option(9). kid	var (bit)	1	Up	0x636c69 656e70	MSB(44)	LSB	KKKK
CoAP. option(9). kid	var	1	Dw	b''	equal	not-sent	

Table 5: Outer SCHC Rule. CoAP Option Numbers: 9 (OSCORE).

```
Protected message:
==================
0x41020001829809046 36c69656e74ffa2c54fe1b434297b62
(24 bytes)

Header:
0x4102
01   Ver
  00   CON
    0001   TKL
        00000010   Request Code 2 "POST"

0x0001 = mid
0x82 = token

Options:

0x98 0904636c69656e74 (9 bytes)
Option 9: OSCORE
Value = 0x0904636c69656e74
          09 = 000 0 1 001 flag byte
                   h k  n
            04 piv
              636c69656e74 kid

0xFF  Payload marker

Payload:
0xa2c54fe1b434297b62 (9 bytes)
```

Figure 13: Protected and Inner SCHC Compressed GET Request.

```
Protected message:
==================
0x614400018290ff10c6d7c26cc1e9aef3f2461e0c29
(21 bytes)

Header:
0x6144
01   Ver
  10   ACK
    0001   TKL
        01000100   Successful Response Code 68 "2.04 Changed"

0x0001 = mid
0x82 = token

Options:

0x90 (1 byte)
Option 9: OSCORE
Value = b''

0xFF  Payload marker

Payload:
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)
```

Figure 14: Protected and Inner SCHC Compressed Content Response.

For the flag bits, some SCHC compression methods are useful, depending on the application. The most straightforward alternative is to provide a fixed value for the flags, combining an "equal" MO and a "not-sent" CDA. This SCHC description saves most bits but could prevent flexibility. Otherwise, SCHC could use a "match-mapping" MO to choose from several configurations for the exchange. If not, the SCHC description may use an MSB MO to mask off the three hard-coded most significant bits.

Note that fixing a flag bit will limit the possible OSCORE options that can be used in the exchange, since the value of the flag bits plays a role in determining a specific OSCORE option.

The piv field lends itself to having some bits masked off with an MSB MO and an LSB CDA. This SCHC description could be useful in applications where the message transmission frequency is low, such as with LPWAN technologies. Note that compressing the piv field may reduce the maximum number of sequence numbers that can be used in an exchange. Once the sequence number exceeds the maximum value, the OSCORE keys need to be re-established.

The size, s, that is included in the kid_ctx field MAY be masked off with an LSB CDA. The rest of the kid_ctx field could have additional bits masked off, or the whole field could be fixed in accordance with an "equal" MO and a "not-sent" CDA. The same holds for the kid field.

The Outer Rule of Table 5 is applied to the example GET request and Content response. Figure 15 and Figure 16 show the resulting messages.

```
Compressed message:
==================
0x0114889458a9fc3686852f6c40 (13 bytes)
0x01 RuleID
    148894 compression residue
          58a9fc3686852f6c40 padded payload

Compression Residue:
0b0001 010 0100  0100 0100
   mid tkn  piv   kid (residue size and residue)

  (19 bits -> 3 bytes with padding)

Payload
0xa2c54fe1b434297b62 (9 bytes)

Compressed message length: 13 bytes
```

Figure 15: SCHC-OSCORE Compressed GET Request.

```
Compressed message:
==================
0x0114218daf84d983d35de7e48c3c1852 (16 bytes)
0x01 RuleID
    14 Compression Residue
      218daf84d983d35de7e48c3c1852 Padded payload

Compression Residue:
0b0001 010 (7 bits -> 1 byte with padding)
   mid tkn

Payload
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)
```

Figure 16: SCHC-OSCORE Compressed Content Response.

In contrast, the following compares these results with what would be obtained by SCHC compressing the original CoAP messages without protecting them with OSCORE, according to the SCHC Rule in Table 6.

RuleID 2

FID	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP. Version	2	1	Bi	1	equal	not-sent	
CoAP. Type	2	1	Up	0	equal	not-sent	
CoAP. Type	2	1	Dw	2	equal	not-sent	
CoAP. TKL	4	1	Bi	1	equal	not-sent	
CoAP. Code	8	1	Up	2	equal	not-sent	
CoAP. Code	8	1	Dw	[69, 132]	match-mapping	mapping-sent	C
CoAP. MID	16	1	Bi	0x0000	MSB(12)	LSB	MMMM
CoAP. Token	tkl	1	Bi	0x80	MSB(5)	LSB	TTT
CoAP. option(11)		1	Up	"temperature"	equal	not-sent	

Table 6: SCHC-CoAP Rule (No OSCORE). CoAP Option Numbers: 11 (Uri-Path).

The Rule in Table 6 yields the SCHC compression results shown in Figure 17 for the request and in Figure 18 for the response.

```
Compressed message:
==================
0x0214
0x02 = RuleID

Compression Residue:
0b00010100 (1 byte)

Compressed message length: 2 bytes
```

Figure 17: CoAP GET Compressed without OSCORE.

```
Compressed message:
==================
0x020a32332043
0x02 = RuleID

Compression Residue:
0b00001010 (1 byte)

Payload
0x32332043

Compressed message length: 6 bytes
```

Figure 18: CoAP Content Compressed without OSCORE.

As it can be seen, the difference between applying SCHC + OSCORE as compared to regular SCHC + CoAP is about 10 bytes.

9. CoAP Header Compression with Proxies

This section defines how SCHC Compression/Decompression is performed when CoAP proxies are deployed. The following refers to the origin client and origin server as application endpoints.

Note that SCHC Compression/Decompression of CoAP headers is not necessarily used between each pair of hops in the communication chain. For example, if a proxy is deployed between an origin client and an origin server, SCHC might be used on the communication leg between the origin client and the proxy, but not on the communication leg between the proxy and the origin server.

9.1. 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.

9.2. With End-to-End Security

In case OSCORE is used end-to-end between client and server (see Section 8.2), 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.

10. Examples of CoAP Header Compression with Proxies

This section provides examples of SCHC Compression/Decompression in the presence of a CoAP proxy.

The presented examples refer to the same deployment considered in Section 2, including a Device communicating over LPWAN with a Network Gateway (NGW), which in turn communicates with an Application Server over the Internet. The Application Server and the Device exchange CoAP messages through the NGW.

The following also applies in the presented examples.

* CoAP request messages are sent only by the Device as targeting the Application Server (uplink traffic), which replies to the Device with corresponding CoAP response messages (downlink traffic). That is, the Device acts as CoAP client, while the Application Server acts as CoAP server.

* A CoAP proxy is co-located on the Network Gateway (NGW) deployed between the Application Server and the Device.

* SCHC is used also on the communication leg between the Application Server and the proxy.

Like in Section 2, the presented examples focus on SCHC Compression/Decompression of CoAP headers, i.e., irrespective of possible SCHC Compression/Decompression applied to further protocol headers.

The example in Section 10.1 considers an exchange of two unprotected messages, while the example in Section 10.2 considers an exchange of two messages protected end-to-end with OSCORE. In the examples, the character | denotes bit concatenation.

Figure 19 and Figure 20 show the two CoAP messages exchanged between the Device and the Application Server via the proxy. The figures show the two messages as originally generated by the application at the two origin endpoints, i.e., before they are possibly protected end-to-end with OSCORE as considered by the example in Section 10.2.

Note that:

* On the communication leg between the Device and the proxy, the CoAP Message ID has value 0x0001 and the CoAP Token has value 0x82.

* On the communication leg between the proxy and the Application Server, the CoAP Message ID has value 0x0004 and the CoAP Token has value 0x75.

```
Original message:
=================
0x41010001823b6578616d706c652e636f6d
  8b74656d7065726174757265d40f636f6170

Header:
0x4101
01   Ver
  00   CON
    0001   TKL
        00000001   Request Code 1 "GET"

0x0001 = mid
0x82 = token

Options:

0x3b6578616d706c652e636f6d
Option 3: Uri-Host
Value = example.com

0x8b74656d7065726174757265
Option 11: Uri-Path
Value = temperature

0xd40f636f6170
Option 39: Proxy-Scheme
Value = coap

Original message length: 35 bytes
```

Figure 19: CoAP GET Request.

```
Original message:
=================
0x6145000475ff32332043

Header:
0x6145
01   Ver
  10   ACK
    0001   TKL
        01000101 Successful Response Code 69 "2.05 Content"

0x0004 = mid
0x75 = token


0xFF Payload marker

Payload:
0x32332043

Original message length: 10 bytes
```

Figure 20: CoAP Content Response.

10.1. Without End-to-End Security

In case OSCORE is not used end-to-end between the Device and the Application Server, the following SCHC Rules are shared between the different entities. Based on those Rules, the SCHC Compression/ Decompression is performed as per Section 9.1.

The Device and the proxy share the SCHC Rule shown in Table 7, with RuleID 0.

```
+----------+
| RuleID 0 |
+----------+
```

FID	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP. Version	2	1	Bi	1	equal	not-sent	
CoAP. Type	2	1	Up	0	equal	not-sent	
CoAP. Type	2	1	Dw	[0, 2]	match-mapping	mapping-sent	T
CoAP. TKL	4	1	Bi	1	equal	not-sent	
CoAP. Code	8	1	Up	[1, 2, 3, 4]	match-mapping	mapping-sent	CC
CoAP. Code	8	1	Dw	[65, 68, 69, 132]	match-mapping	mapping-sent	CC
CoAP. MID	16	1	Bi	0x0000	MSB(12)	LSB	MMMM
CoAP. Token	tkl	1	Bi	0x80	MSB(5)	LSB	TTT
CoAP. option(3)	var (B)	1	Up		ignore	value-sent	
CoAP. option(11)	var	1	Up	"temperature"	equal	not-sent	
CoAP. option(39)	var	1	Up	"coap"	equal	not-sent	

Table 7: SCHC Rule between the Device and the Proxy. CoAP Option Numbers: 3 (Uri-Host), 11 (Uri-Path), 39 (Proxy-Scheme).

Instead, the proxy and the Application Server share the SCHC Rule shown in Table 8, with RuleID 1.

RuleID 1

FID	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP. Version	2	1	Bi	1	equal	not-sent	
CoAP. Type	2	1	Up	0	equal	not-sent	
CoAP. Type	2	1	Dw	[0, 2]	match-mapping	mapping-sent	T
CoAP. TKL	4	1	Bi	1	equal	not-sent	
CoAP. Code	8	1	Up	[1, 2, 3, 4]	match-mapping	mapping-sent	CC
CoAP. Code	8	1	Dw	[65, 68, 69, 132]	match-mapping	mapping-sent	CC
CoAP. MID	16	1	Bi	0x0000	MSB(12)	LSB	MMMM
CoAP. Token	tkl	1	Bi	0x70	MSB(5)	LSB	TTT
CoAP. option(3)	var (B)	1	Up		ignore	value-sent	
CoAP. option(11)	var	1	Up	"temperature"	equal	not-sent	

Table 8: SCHC Rule between the Proxy and the Application Server.
CoAP Option Numbers: 3 (Uri-Host), 11 (Uri-Path).

First, the Device applies the Rule in Table 7 shared with the proxy to the CoAP request in Figure 19. The result is the compressed CoAP request in Figure 21 that the Device sends to the proxy.

```
Compressed message:
==================
0x00055b2bc30b6b836329731b7b68 (14 bytes)
0x00 RuleID
    055b2bc30b6b836329731b7b68 compression residue
                                and padded payload

Compression Residue (101 bits -> 13 bytes with padding)
0b   00 0001 010      1011  |  0x6578616d706c652e636f6d
   code  mid tkn  Uri-Host (residue size and residue)

Compressed message length: 14 bytes
```

Figure 21: CoAP GET Request Compressed for the Proxy.

Upon receiving the message in Figure 21, the proxy decompresses it with the Rule in Table 7 shared with the Device and obtains the same CoAP request in Figure 19.

After that, the proxy removes the Proxy-Scheme Option from the decompressed CoAP request. Also, the proxy replaces the values of the CoAP Message ID and of the CoAP Token to 0x0004 and 0x75, respectively. The result is the CoAP request shown in Figure 22.

```
Message to forward:
==================
0x41010004753b6578616d706c652e636f6d
  8b74656d7065726174757265

Header:
0x4101
01   Ver
  00   CON
    0001   TKL
        00000001   Request Code 1 "GET"

0x0004 = mid
0x75 = token

Options:

0x3b6578616d706c652e636f6d
Option 3: Uri-Host
Value = example.com

0x8b74656d7065726174757265
Option 11: Uri-Path
Value = temperature

Original message length: 29 bytes
```

Figure 22: CoAP GET Request to be Compressed by the Proxy.

Then, the proxy applies the Rule in Table 8 shared with the Application Server to the CoAP request in Figure 22.

The result is the compressed CoAP request in Figure 23 that the proxy forwards to the Application Server.

```
Compressed message to forward:
=================
0x0112db2bc30b6b836329731b7b68 (14 bytes)
0x01 RuleID
    12db2bc30b6b836329731b7b68 compression residue
                                and padded payload


Compression Residue (101 bits -> 13 bytes with padding)
0b   00 0100 101      1011  |  0x6578616d706c652e636f6d
   code  mid tkn  Uri-Host (residue size and residue)

Compressed message length: 14 bytes
```

Figure 23: CoAP GET Request Forwarded by the Proxy.

Upon receiving the message in Figure 23, the Application Server decompresses it using the Rule in Table 8 shared with the proxy. The result is the same CoAP request in Figure 22 that the Application Server delivers to the application.

After that, the Application Server produces the CoAP response in Figure 20 and compresses it using the Rule in Table 8 shared with the proxy. The result is the compressed CoAP response shown in Figure 24 that the Application Server sends to the proxy.

```
Compressed message:
==================
0x01c94c8cc810c0 (7 bytes)
0x01 RuleID
    c94c8cc810c0 compression residue
                 and padded payload


Compression Residue (10 bits -> 2 bytes with padding)
0b    1   10 0100 101
   type code  mid tkn

Payload
0x32332043 (4 bytes)

Compressed message length: 7 bytes
```

Figure 24: CoAP Content Response Compressed for the Proxy.

Upon receiving the message in Figure 24, the proxy decompresses it using the Rule in Table 8 shared with the Application Server. The result is the same CoAP response in Figure 20.

Then, the proxy replaces the values of the CoAP Message ID and of the CoAP Token to 0x0001 and 0x82, respectively. The result is the CoAP response shown in Figure 25.

```
Message to forward:
==================
0x6145000182ff32332043

Header:
0x6145
01   Ver
  10   ACK
    0001   TKL
        01000101 Successful Response Code 69 "2.05 Content"

0x0001 = mid
0x82 = token


0xFF Payload marker

Payload:
0x32332043

Original message length: 10 bytes
```

Figure 25: CoAP Content Response to be Compressed by the Proxy.

Then, the proxy compresses the CoAP response in Figure 25 with the Rule in Table 7 shared with the Device. The result is the compressed CoAP response shown in Figure 26 that the proxy forwards to the Device.

```
Compressed message:
==================
0x00c28c8cc810c0 (7 bytes)
0x00 RuleID
    c28c8cc810c0 compression residue
                 and padded payload


Compression Residue (10 bits -> 2 bytes with padding)
0b    1   10 0001 010
   type code  mid tkn

Payload
0x32332043 (4 bytes)

Compressed message length: 7 bytes
```

Figure 26: CoAP Content Response Forwarded by the Proxy.

Upon receiving the message in Figure 26, the Device decompresses it using the Rule in Table 7 shared with the proxy. The result is the same CoAP request in Figure 25 that the Device delivers to the application.

10.2. With End-to-End Security

In case OSCORE is used end-to-end between the Device and the Application Server, the following SCHC Rules are shared between the different entities. Based on those Rules, the SCHC Compression/Decompression is performed as per Section 9.2.

The Device and the Application Server share the SCHC Rule shown in Table 9, with RuleID 2. The Device and the Application Server use this Rule to perform the Inner SCHC Compression/Decompression end-to-end.

RuleID 2

FID	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP. Code	8	1	Up	[1, 2, 3, 4]	match-mapping	mapping-sent	CC
CoAP. Code	8	1	Dw	[65, 68, 69, 132]	match-mapping	mapping-sent	CC
CoAP. option(11)	var	1	Up	"temperature"	equal	not-sent	

Table 9: Inner SCHC Rule between the Device and the Application Server. CoAP Option Numbers: 11 (Uri-Path).

The Device and the proxy share the SCHC Rule shown in Table 10, with RuleID 3. The Device and the proxy use this Rule to perform the Outer SCHC Compression/Decompression hop-by-hop on their communication leg.

RuleID 3

FID	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP. Version	2	1	Bi	1	equal	not-sent	
CoAP. Type	2	1	Up	0	equal	not-sent	
CoAP. Type	2	1	Dw	[0, 2]	match-mapping	mapping-sent	T
CoAP. TKL	4	1	Bi	1	equal	not-sent	
CoAP. Code	8	1	Up	2	equal	not-sent	
CoAP. Code	8	1	Dw	68	equal	not-sent	
CoAP. MID	16	1	Bi	0x0000	MSB(12)	LSB	MMMM
CoAP. Token	tkl	1	Bi	0x80	MSB(5)	LSB	TTT
CoAP. option(3)	var (B)	1	Up		ignore	value-sent	
CoAP. option(9). flags	var	1	Up	0x09	equal	not-sent	
CoAP. option(9). flags	var	1	Dw	b''	equal	not-sent	
CoAP. option(9). piv	osc.piv	1	Up	0x00	MSB(4)	LSB	PPPP
CoAP. option(9). piv	var	1	Dw	b''	equal	not-sent	
CoAP.	var	1	Bi	b''	equal	not-sent	

option(9). kid_ctx							
CoAP. option(9). x	8	1	Bi	b''	equal	not-sent	
CoAP. option(9). nonce	osc.x.m	1	Bi	b''	equal	not-sent	
CoAP. option(9). kid	var (bit)	1	Up	0x0000	MSB(12)	LSB	KKKK
CoAP. option(9). kid	var	1	Dw	b''	equal	not-sent	
CoAP. option(39)	var	1	Up	"coap"	equal	not-sent	

Table 10: Outer SCHC Rule between the Device and the Proxy. CoAP Option Numbers: 3 (Uri-Host), 9 (OSCORE), 39 (Proxy-Scheme).

The proxy and the Application Server share the SCHC Rule shown in Table 11, with RuleID 4. The proxy and the Application Server use this Rule to perform the Outer SCHC Compression/Decompression hop-by-hop on their communication leg.

RuleID 4

FID	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP. Version	2	1	Bi	1	equal	not-sent	
CoAP. Type	2	1	Up	0	equal	not-sent	
CoAP. Type	2	1	Dw	[0, 2]	match-mapping	mapping-sent	T

CoAP. TKL	4	1	Bi	1	equal	not-sent	
CoAP. Code	8	1	Up	2	equal	not-sent	
CoAP. Code	8	1	Dw	68	equal	not-sent	
CoAP. MID	16	1	Bi	0x0000	MSB(12)	LSB	MMMM
CoAP. Token	tkl	1	Bi	0x70	MSB(5)	LSB	TTT
CoAP. option(3)	var (B)	1	Up		ignore	value-sent	
CoAP. option(9). flags	var	1	Up	0x09	equal	not-sent	
CoAP. option(9). flags	var	1	Dw	b''	equal	not-sent	
CoAP. option(9). piv	osc.piv	1	Up	0x00	MSB(4)	LSB	PPPP
CoAP. option(9). piv	var	1	Dw	b''	equal	not-sent	
CoAP. option(9). kid_ctx	var	1	Bi	b''	equal	not-sent	
CoAP. option(9). x	8	1	Bi	b''	equal	not-sent	
CoAP. option(9). nonce	osc.x.m	1	Bi	b''	equal	not-sent	
CoAP.	var	1	Up	0x0000	MSB(12)	LSB	KKKK

option(9). kid	(bit)							
CoAP. option(9). kid	var	1	Dw	b''	equal	not-sent		

Table 11: Outer SCHC Rule between the Proxy and the Application Server. CoAP Option Numbers: 3 (Uri-Host), 9 (OSCORE).

When the Device applies the Rule in Table 9 shared with the Application Server to the CoAP request in Figure 19, this results in the Compressed Plaintext shown in Figure 27.

As per Section 8.2, the message follows the process of SCHC Inner Compression and encryption until the payload (if any). The ciphertext resulting from the overall Inner process is used as payload of the Outer OSCORE message.

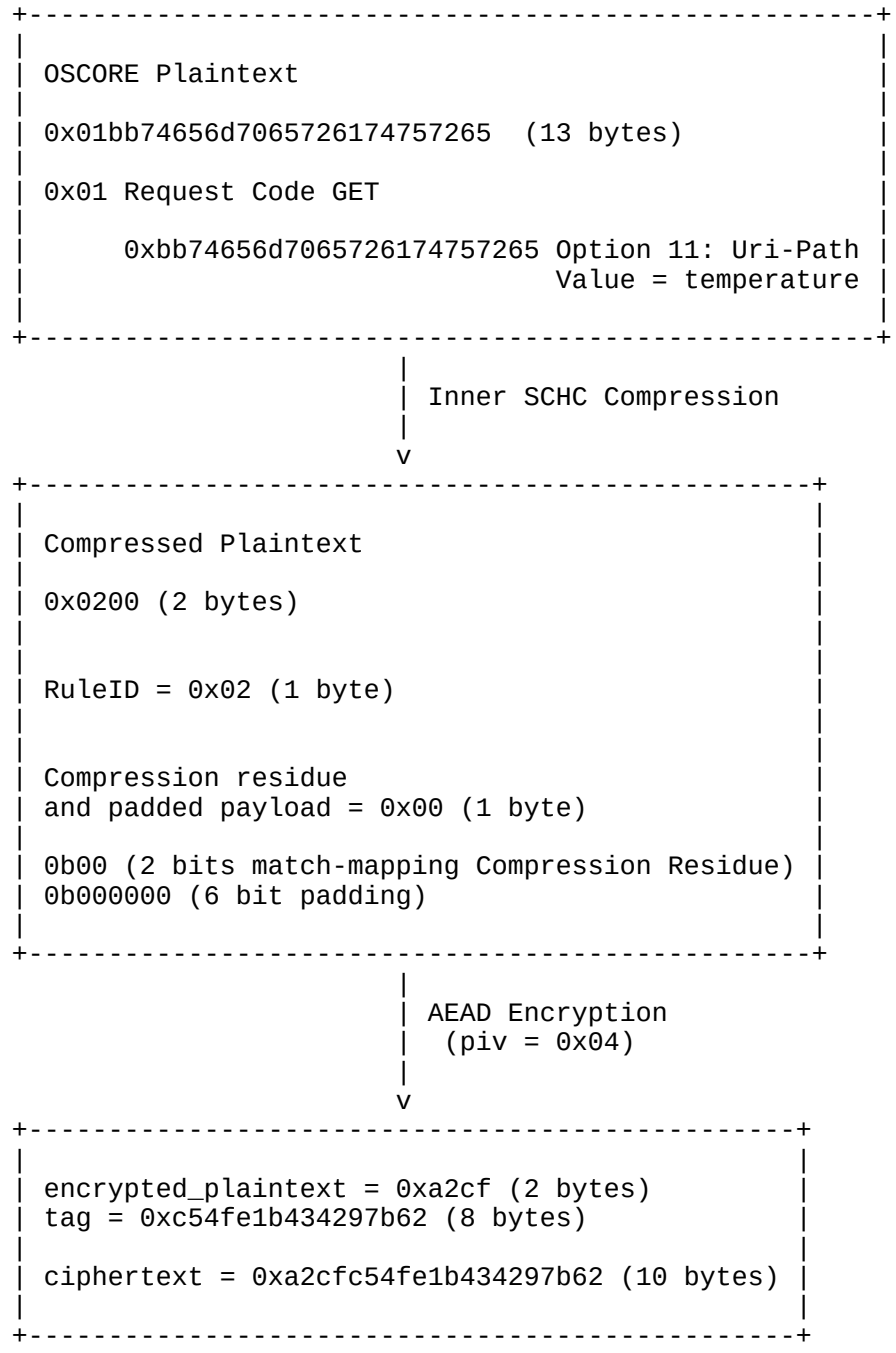
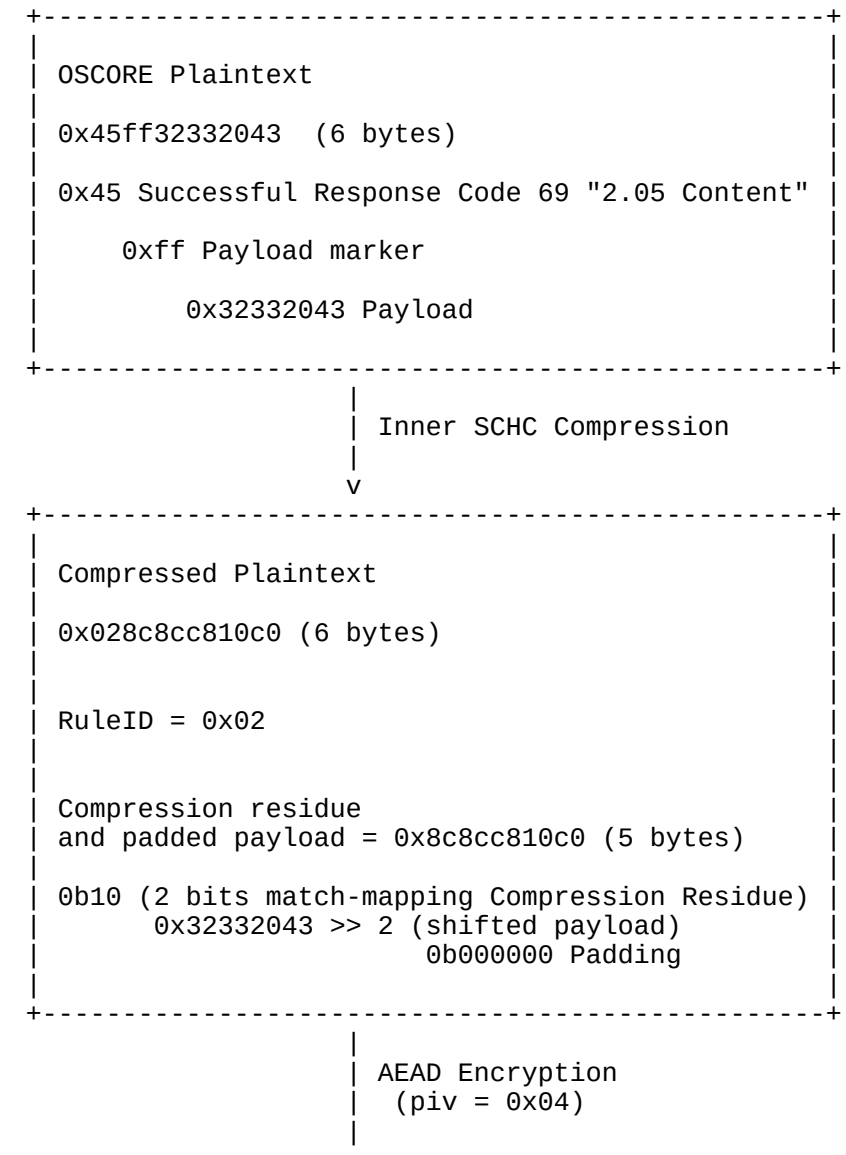


Figure 27: Plaintext Compression and Encryption for the GET Request.

When the Application Server applies the Rule in Table 9 shared with the Device to the CoAP response in Figure 20, this results in the Compressed Plaintext shown in Figure 28.

As per Section 8.2, the message follows the process of SCHC Inner Compression and encryption until the payload (if any). The ciphertext resulting from the overall Inner process is used as payload of the Outer OSCORE message.



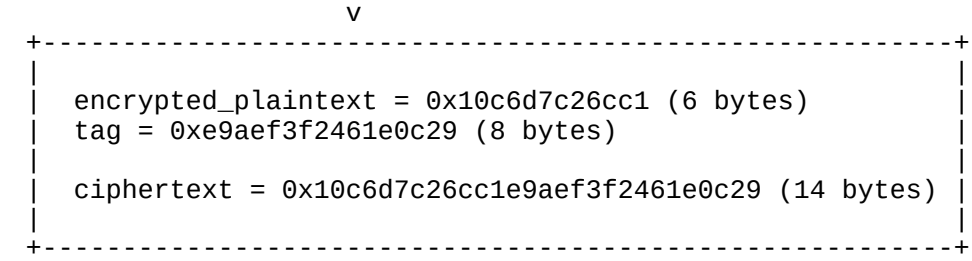


Figure 28: Plaintext Compression and Encryption for the Content Response.

After having performed the SCHC Inner Compression of the CoAP request in Figure 19, the Device protects it with OSCORE by considering the Compressed Plaintext in Figure 27. The result is the OSCORE-protected CoAP request shown in Figure 29.

```
Protected message:
==================
0x41020001823b6578616d706c652e636f6d
  6409040005d411636f6170ffa2cfc54fe1b434297b62
(39 bytes)

Header:
0x4102
01   Ver
  00   CON
    0001   TKL
        00000010   Request Code 2 "POST"

0x0001 = mid
0x82 = token

Options:

0x3b6578616d706c652e636f6d
Option 3: Uri-Host
Value = example.com

0x6409040005
Option 9: OSCORE
Value = 0x09040005
          09 = 000 0 1 001 flag byte
                   h k  n
            04 piv
              0005 kid

0xd411636f6170
Option 39: Proxy-Scheme
Value = coap


0xFF Payload marker

Payload:
0xa2cfc54fe1b434297b62 (10 bytes)
```

Figure 29: Protected and Inner SCHC Compressed CoAP GET Request.

Then, the Device applies the Rule in Table 10 shared with the proxy to the OSCORE-protected CoAP request in Figure 29, thus performing the SCHC Outer Compression of such request. The result is the OSCORE-protected and Outer Compressed CoAP request shown in Figure 30 that the Device sends to the proxy.

```
Compressed message:
==================
0x03156caf0c2dae0d8ca5cc6deda88b459f8a9fc3686852f6c4 (25 bytes)
0x03 RuleID
    156caf0c2dae0d8ca5cc6deda88b compression residue
                                459f8a9fc3686852f6c4 padded payload

Compression Residue
0b0001 010      1011 | 0x6578616d706c652e636f6d |
   mid tkn  Uri-Host (residue size and residue)

0b0100  0100 0101
   piv   kid (residue size and residue)

   (111 bits -> 14 bytes with padding)

Payload
0xa2cfc54fe1b434297b62 (10 bytes)

Compressed message length: 25 bytes
```

Figure 30: SCHC-OSCORE CoAP GET Request Compressed for the Proxy.

Upon receiving the message in Figure 30, the proxy decompresses it with the Rule in Table 10 shared with the Device, thus performing the SCHC Outer Decompression. The result is the same OSCORE-protected CoAP request in Figure 29.

After that, the proxy removes the Proxy-Scheme Option from the decompressed OSCORE-protected CoAP request. Also, the proxy replaces the values of the CoAP Message ID and of the CoAP Token to 0x0004 and 0x75, respectively. The result is the OSCORE-protected CoAP request shown in Figure 31.

```
Protected message:
==================
0x41020004753b6578616d706c652e636f6d
  6409040005ffa2cfc54fe1b434297b62
(33 bytes)

Header:
0x4102
01   Ver
  00   CON
    0001   TKL
        00000010   Request Code 2 "POST"

0x0004 = mid
0x75 = token

Options:

0x3b6578616d706c652e636f6d
Option 3: Uri-Host
Value = example.com

0x6409040005
Option 9: OSCORE
Value = 0x09040005
          09 = 000 0 1 001 flag byte
                   h k  n
            04 piv
              0005 kid


0xFF Payload marker

Payload:
0xa2cfc54fe1b434297b62 (10 bytes)
```

Figure 31: SCHC-OSCORE CoAP GET Request to be Compressed by the Proxy.

Then, the proxy applies the Rule in Table 11 shared with the Application Server to the OSCORE-protected CoAP request in Figure 31, thus performing the SCHC Outer Compression of such request. The result is the OSCORE-protected and Outer Compressed CoAP request shown in Figure 32 that the proxy forwards to the Application Server.

```
Compressed message:
==================
0x044b6caf0c2dae0d8ca5cc6deda88b459f8a9fc3686852f6c4 (25 bytes)
0x04 RuleID
    4b6caf0c2dae0d8ca5cc6deda88b compression residue
                               459f8a9fc3686852f6c4 padded payload


Compression Residue
0b0100 101      1011 | 0x6578616d706c652e636f6d |
   mid tkn  Uri-Host (residue size and residue)

0b0100  0100 0101
   piv   kid (residue size and residue)

   (111 bits -> 14 bytes with padding)


Payload
0xa2cfc54fe1b434297b62 (10 bytes)

Compressed message length: 25 bytes
```

Figure 32: SCHC-OSCORE CoAP GET Request Forwarded by the Proxy.

Upon receiving the message in Figure 32, the Application Server decompresses it using the Rule in Table 11 shared with the proxy, thus performing the SCHC Outer Decompression. The result is the same OSCORE-protected CoAP request in Figure 31.

The Application Server decrypts and verifies such a request, which results in the same Compressed Plaintext in Figure 27. Then, the Application Server applies the Rule in Table 9 shared with the Device to such a Compressed Plaintext, thus performing the SCHC Inner Decompression. The result is used to rebuild the same CoAP request in Figure 19 that the Application Server delivers to the application.

After having performed the SCHC Inner Compression of the CoAP response in Figure 20, the Application Server protects it with OSCORE by considering the Compressed Plaintext in Figure 28. The result is the OSCORE-protected CoAP response shown in Figure 33.

```
Protected message:
==================
0x614400047590ff10c6d7c26cc1e9aef3f2461e0c29
(21 bytes)

Header:
0x6144
01   Ver
  10   ACK
    0001   TKL
        01000100   Successful Response Code 68 "2.04 Changed"

0x0004 = mid
0x75 = token

Options:

0x90
Option 9: OSCORE
Value = b''


0xFF Payload marker

Payload:
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)
```

Figure 33: Protected and Inner SCHC Compressed CoAP Content Response.

Then, the Application Server applies the Rule in Table 11 shared with the proxy to the OSCORE-protected CoAP response in Figure 33, thus performing the SCHC Outer Compression of such response. The result is the OSCORE-protected and Outer Compressed CoAP response shown in Figure 34 that the Application Server sends to the proxy.

```
Compressed message:
==================
0x04a510c6d7c26cc1e9aef3f2461e0c29  (16 bytes)
0x04 RuleID
    a510c6d7c26cc1e9aef3f2461e0c29 compression residue
                                   and padded payload


Compression Residue (8 bits -> 1 byte with padding)
0b    1 0100 101
   type  mid tkn

Payload
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)

Compressed message length: 16 bytes
```

Figure 34: SCHC-OSCORE CoAP Content Response Compressed for the Proxy.

Upon receiving the message in Figure 34, the proxy decompresses it with the Rule in Table 11 shared with the Application Server, thus performing the SCHC Outer Decompression. The result is the same OSCORE-protected CoAP response in Figure 33.

After that, the proxy replaces the values of the CoAP Message ID and of the CoAP Token to 0x0001 and 0x82, respectively. The result is the OSCORE-protected CoAP response shown in Figure 35.

```
Protected message:
==================
0x614400018290ff10c6d7c26cc1e9aef3f2461e0c29
(21 bytes)

Header:
0x6144
01   Ver
  10   ACK
    0001   TKL
        01000100   Successful Response Code 68 "2.04 Changed"

0x0001 = mid
0x82 = token

Options:

0x90
Option 9: OSCORE
Value = b''


0xFF Payload marker

Payload:
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)
```

Figure 35: SCHC-OSCORE CoAP Content Response to be Compressed by the Proxy.

Then, the proxy applies the Rule in Table 10 shared with the Device to the OSCORE-protected CoAP response in Figure 35, thus performing the SCHC Outer Compression of such response. The result is the OSCORE-protected and Outer Compressed CoAP response shown in Figure 36 that the proxy forwards to the Device.

```
Compressed message:
==================
0x038a10c6d7c26cc1e9aef3f2461e0c29 (16 bytes)
0x03 RuleID
    8a10c6d7c26cc1e9aef3f2461e0c29 compression residue
                                   and padded payload


Compression Residue (8 bits -> 1 byte with padding)
0b    1 0001 010
   type  mid tkn

Payload
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)

Compressed message length: 16 bytes
```

Figure 36: SCHC-OSCORE CoAP Content Response Forwarded by the Proxy.

Upon receiving the message in Figure 36, the Device decompresses it using the Rule in Table 10 shared with the proxy, thus performing the SCHC Outer Decompression. The result is the same OSCORE-protected CoAP response in Figure 35.

The Device decrypts and verifies such a response, which results in the same Compressed Plaintext shown in Figure 28. Then, the Device applies the Rule in Table 9 shared with the Application Server to such a Compressed Plaintext, thus performing the SCHC Inner Decompression. The result is used to rebuild the same CoAP response in Figure 20 that the Device delivers to the application.

11. CoAP Fields

Table 12 lists the CoAP fields and subfields for which SCHC compression has been defined or revised in this document.

Editor’s note: Before publication, confirm or amend the option numbers associated with the CoAP options Proxy-Cri and Proxy-Scheme-Number defined in [I-D.ietf-core-href], to ensure that they are consistent with those registered in the "CoAP Option Numbers" registry.

Field	Description
CoAP.Version	CoAP header field Version [RFC7252]
CoAP.Type	CoAP header field Type [RFC7252]

CoAP.TKL	CoAP header field Token Length (TKL) [RFC7252][RFC8974]
CoAP.Code	CoAP header field Code [RFC7252]
CoAP.Code.Class	CoAP header field Code (subfield Class) [RFC7252]
CoAP.Code.Detail	CoAP header field Code (subfield Detail) [RFC7252]
CoAP.MID	CoAP header field Message ID [RFC7252]
CoAP.Token	CoAP field Token [RFC7252][RFC8974]
CoAP.option(1)	CoAP option If-Match [RFC7252]
CoAP.option(3)	CoAP option Uri-Host [RFC7252]
CoAP.option(4)	CoAP option ETag [RFC7252]
CoAP.option(5)	CoAP option If-None-Match [RFC7252]
CoAP.option(6)	CoAP option Observe [RFC7641]
CoAP.option(7)	CoAP option Uri-Port [RFC7252]
CoAP.option(8)	CoAP option Location-Path [RFC7252]
CoAP.option(9)	CoAP option OSCORE [RFC8613][I-D.ietf-core-oscore-key-update]
CoAP.option(9).flags	CoAP option OSCORE (subfield flags) [RFC8613][I-D.ietf-core-oscore-key-update]
CoAP.option(9).piv	CoAP option OSCORE (subfield piv) [RFC8613]
CoAP.option(9).kid_ctx	CoAP option OSCORE (subfield kid_ctx) [RFC8613]
CoAP.option(9).x	CoAP option OSCORE (subfield x) [I-D.ietf-core-oscore-key-update]
CoAP.option(9).nonce	CoAP option OSCORE (subfield nonce) [I-D.ietf-core-oscore-key-update]

CoAP.option(9).kid	CoAP option OSCORE (subfield kid) [RFC8613]
CoAP.option(11)	CoAP option Uri-Path [RFC7252]
CoAP.option(12)	CoAP option Content-Format [RFC7252]
CoAP.option(14)	CoAP option Max-Age [RFC7252]
CoAP.option(15)	CoAP option Uri-Query [RFC7252]
CoAP.option(16)	CoAP option Hop-Limit [RFC8768]
CoAP.option(17)	CoAP option Accept [RFC7252]
CoAP.option(19)	CoAP option Q-Block1 [RFC9177]
CoAP.option(20)	CoAP option Location-Query [RFC7252]
CoAP.option(21)	CoAP option EDHOC [RFC9668]
CoAP.option(23)	CoAP option Block2 [RFC7959][RFC8323]
CoAP.option(27)	CoAP option Block1 [RFC7959][RFC8323]
CoAP.option(28)	CoAP option Size2 [RFC7959]
CoAP.option(31)	CoAP option Q-Block2 [RFC9177]
CoAP.option(35)	CoAP option Proxy-Uri [RFC7252]
CoAP.option(39)	CoAP option Proxy-Scheme [RFC7252]
CoAP.option(60)	CoAP option Size1 [RFC7252]
CoAP.option(235)	CoAP option Proxy-Cri [I-D.ietf-core-href]
CoAP.option(239)	CoAP option Proxy-Scheme-Number [I-D.ietf-core-href]
CoAP.option(252)	CoAP option Echo [RFC9175]
CoAP.option(258)	CoAP option No-Response [RFC7967]
CoAP.option(292)	CoAP option Request-Tag [RFC9175]

Table 12: CoAP Fields.

12. Security Considerations

The use of SCHC header compression for CoAP header fields only affects the representation of the header information. SCHC header compression itself does not increase or decrease the overall level of security of the communication. When the connection does not use a security protocol (e.g., OSCORE or DTLS), it is necessary to use a Layer 2 security mechanism to protect the SCHC packets.

If an LPWAN is the Layer 2 technology being used, the SCHC security considerations discussed in [RFC8724] continue to apply. When using another Layer 2 protocol, the use of a cryptographic integrity-protection mechanism to protect the SCHC headers is REQUIRED. Such cryptographic integrity protection is necessary in order to continue to provide the properties that [RFC8724] relies upon.

When SCHC is used with OSCORE, the security considerations discussed in [RFC8613] continue to apply. When SCHC is used with Group OSCORE, the security considerations discussed in [I-D.ietf-core-oscore-groupcomm] 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 the OSCORE Outer headers, the Initialization Vector (IV) size in the Compression Residue must be carefully selected. There is a trade-off between compression efficiency (with a longer MSB MO prefix) and the frequency at which the Device must renew its key material (in order to prevent the IV from expanding to an incompressible value). The key-renewal operation itself may require several message exchanges and result in energy-intensive computation, but the optimal trade-off will depend on the specifics of the Device and expected usage patterns. In order to renew its key material with another OSCORE endpoint, the Device can rely on the lightweight key update protocol KUDOS defined in [I-D.ietf-core-oscore-key-update].

If an attacker can introduce a corrupted SCHC-compressed packet onto a link, DoS attacks can be mounted by causing excessive resource consumption at the decompressor. However, an attacker able to inject packets at the link layer is also capable of other, potentially more damaging, attacks.

SCHC compression emits variable-length Compression Residues for some CoAP fields. In the representation of the compressed header, the length field that is sent is not the length of the original header field but rather the length of the Compression Residue that is being transmitted. If a corrupted packet arrives at the decompressor with a longer or shorter length than that of the original compressed representation, the SCHC decompression procedures will detect an error and drop the packet.

SCHC header compression Rules MUST remain tightly coupled between the compressor and the decompressor. If the compression Rules get out of sync, a Compression Residue might be decompressed differently at the receiver, thus yielding a result different than the initial message submitted to compression procedures. Accordingly, any time the context Rules are updated on an OSCORE endpoint, that endpoint MUST trigger OSCORE key re-establishment, e.g., by running the lightweight key update protocol KUDOS [I-D.ietf-core-oscore-key-update]. Similar procedures may be appropriate to signal Rule updates when other message-protection mechanisms are in use.

12.1. YANG Module

The YANG data model defined in Appendix A extends the ietf-schc module defined in [RFC9363].

Therefore, all the security considerations compiled in Section 8 of [RFC9363] also apply to the resulting extended YANG data model.

13. IANA Considerations

This document has the following actions for IANA.

Note to RFC Editor: Please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

13.1. IETF XML

IANA is asked to register the following entry in the "IETF XML" registry [RFC3688].

* URI: urn:ietf:params:xml:ns:yang:ietf-schc-coap

* Registrant Contact: The IESG.

* XML: N/A; the requested URI is an XML namespace.

13.2. YANG Module Names

IANA is asked to register the following entry in the "YANG Module Names" registry [RFC6020][RFC8407] within the "YANG Parameters" registry group.

* Name: ietf-schc-coap
* Namespace: urn:ietf:params:xml:ns:yang:ietf-schc-coap
* Prefix: schc-coap
* Reference: [RFC-XXXX]

13.3. SCHC Compression of CoAP Fields

IANA is asked to establish the "SCHC Compression of CoAP Fields" IANA registry.

As registration policy, the registry uses "Specification Required" per Section 4.6 of [RFC8126]. Expert Review guidelines are provided in Section 13.4.

13.3.1. Intended Use

The objective of this registry is to collect a list of CoAP fields and subfields, for which it has been defined how to perform SCHC compression.

Such a definition does not necessarily have to be in the same documentation that defines the CoAP fields and subfields in question. While that can be the case, it is also possible to provide that definition in a separate specification.

Each entry of the registry is intended to include references to the documentation that defines the associated CoAP field or subfield, as well as references to the specifications that define the SCHC compression of that CoAP field or subfield.

When a specification defines how to perform SCHC compression of a CoAP field, the following applies.

* If a registry entry for the CoAP field does not already exist, it is strongly encouraged to register an associated new entry.
* If a registry entry for the CoAP field already exists, it is strongly encouraged to update its list of references. The update is intended to add references to the specification that provides

the new or updated SCHC compression of the CoAP field, as well as to any documentation that updates the definition of the CoAP field itself.

If the defined SCHC compression considers the CoAP field as composed of subfields, it is strongly encouraged that the same as above is also performed for each subfield and the associated registry entry.

13.3.2. Structure of Entries

The columns of this registry are:

* Field: a unique identifier of the CoAP field or subfield associated with this entry. This identifier is used as FID in a Field Descriptor of a SCHC compression Rule for compressing/decompressing CoAP messages.

  This identifier has two possible formats:

  - "CoAP.X", where X is the name of the CoAP field.

  - "CoAP.X.Y", where X is the name of the CoAP field and Y is the name of a subfield of X.

  If the CoAP field in question is specifically a CoAP option, then X has the format "option(N)", where N is the option number of the CoAP option. The value N is taken from the "Number" column of the corresponding entry in the "CoAP Option Numbers" IANA registry [CoAP.Option.Numbers].

  This identifier must have a corresponding item or set of items in the YANG data model for the CoAP field or subfield associated with this entry, as specified in Section 6 of [RFC9363] or in Appendix A of [RFC-XXXX].

* Description: a short description of the CoAP field or subfield associated with this entry, together with public references to the resources that define it.

* Reference: public references to the resources that define how a SCHC compression Rule works for the CoAP field or subfield associated with this entry.

This registry has been initially populated with the values in Table 12. The "Reference" column for all these entries refers to this document.

13.4. Expert Review Instructions

The IANA registry established in this document is defined as "Specification Required". This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

* Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments.

  Specifically, for every CoAP field, only one corresponding registry entry is allowed. Also, for every CoAP subfield, only one corresponding registry entry is allowed.

* Consistent with the "Specification Required" registration policy, specifications should exist, but early assignment before a specification is available is considered to be permissible. When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for.

If the expert becomes aware of a definition for SCHC compression of CoAP fields and subfields that is deployed and in use, the expert may also initiate a registration or update an existing one on their own, if they deem important that the definition in question gains visibility through the registry entry.

14. References

14.1. Normative References

[CoAP.Option.Numbers]
IANA, "CoAP Option Numbers", <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#option-numbers>.

[I-D.ietf-core-href]
Bormann, C. and H. Birkholz, "Constrained Resource Identifiers", Work in Progress, Internet-Draft, draft-ietf-core-href-22, 20 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-href-22>.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and R. H旦glund, "Group Object Security for Constrained RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-26, 5 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-26>.

[I-D.ietf-core-oscore-key-update]
H旦glund, R. and M. Tiloca, "Key Update for OSCORE (KUDOS)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-key-update-11, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-key-update-11>.

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/rfc/rfc3688>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/rfc/rfc5116>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/rfc/rfc6020>.

[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>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/rfc/rfc7641>.

[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/rfc/rfc7959>.

[RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <https://www.rfc-editor.org/rfc/rfc7967>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/rfc/rfc8323>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/rfc/rfc8407>.

[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>.

[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>.

[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/rfc/rfc8768>.

[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>.

[RFC8974] Hartke, K. and M. Richardson, "Extended Tokens and Stateless Clients in the Constrained Application Protocol (CoAP)", RFC 8974, DOI 10.17487/RFC8974, January 2021, <https://www.rfc-editor.org/rfc/rfc8974>.

[RFC9175] Ams端ss, C., Preu 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/rfc/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/rfc/rfc9177>.

[RFC9363] Minaburo, A. and L. Toutain, "A YANG Data Model for Static Context Header Compression (SCHC)", RFC 9363, DOI 10.17487/RFC9363, March 2023, <https://www.rfc-editor.org/rfc/rfc9363>.

[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>.

14.2. Informative References

[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-14, 2 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-14>.

[I-D.ietf-schc-universal-option]
Lampin, Q., Minaburo, A., Tiloca, M., and L. Toutain, "Options representation in SCHC YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-schc-universal-option-00, 15 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-schc-universal-option-00>.

[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/rfc/rfc8824>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[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>.

Appendix A. YANG Data Model

This appendix defines the ietf-schc-coap module, which extends the ietf-schc module defined in [RFC9363] to include the new CoAP options as defined in the present document.

```
<CODE BEGINS> file "ietf-schc-coap@2025-07-07.yang"
module ietf-schc-coap {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-schc-coap";
  prefix schc-coap;

  import ietf-schc {
      prefix schc;
  }

  organization
    "IETF Static Context Header Compression (schc) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/schc/about/>
     WG List:  <mailto:schc@ietf.org>
     Editor:   Marco Tiloca
       <mailto:marco.tiloca@ri.se>";
  description
    "Copyright (c) 2021 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.
     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).
```

```
      This version of this YANG module is part of RFC XXXX
      (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
      for full legal notices.
      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 (RFC 2119) (RFC 8174) when, and only when,
      they appear in all capitals, as shown here.
      ****************************************************************

      This module extends the ietf-schc module defined in RFC 9363 to
      include the new CoAP options as defined in RFC YYYY.";

    revision 2025-07-07 {
      description
        "New CoAP extensions and extended OSCORE fields.";
      reference
        "RFC YYYY Static Context Header Compression (SCHC) for the
                  Constrained Application Protocol (CoAP) (see
                  Sections 5 and 6)";
    }

    // Field ID

    identity fid-coap-option-proxy-cri {
      base "schc:fid-coap-option";
      description
        "Proxy-Cri option.";
      reference
        "RFC XXXX Constrained Resource Identifiers";
    }

    identity fid-coap-option-proxy-scheme-number {
      base "schc:fid-coap-option";
      description
        "Proxy-Scheme-Number option.";
      reference
        "RFC XXXX Constrained Resource Identifiers";
    }

    identity fid-coap-option-hop-limit {
      base "schc:fid-coap-option";
      description
        "Hop Limit option to avoid infinite forwarding loops.";
      reference
        "RFC 8768 Constrained Application Protocol (CoAP)
                  Hop-Limit Option";
    }
```

```
identity fid-coap-option-echo {
  base "schc:fid-coap-option";
  description
    "Echo option.";
  reference
    "RFC 9175 Constrained Application Protocol (CoAP):
              Echo, Request-Tag, and Token Processing";
}

identity fid-coap-option-request-tag {
  base "schc:fid-coap-option";
  description
    "Request-Tag option.";
  reference
    "RFC 9175 Constrained Application Protocol (CoAP):
              Echo, Request-Tag, and Token Processing";
}

identity fid-coap-option-q-block1 {
  base "schc:fid-coap-option";
  description
    "Q-Block1 option.";
  reference
    "RFC 9177 Constrained Application Protocol (CoAP)
              Block-Wise Transfer Options Supporting
              Robust Transmission";
}

identity fid-coap-option-q-block2 {
  base "schc:fid-coap-option";
  description
    "Q-Block2 option.";
  reference
    "RFC 9177 Constrained Application Protocol (CoAP)
              Block-Wise Transfer Options Supporting
              Robust Transmission";
}

identity fid-coap-option-edhoc {
  base "schc:fid-coap-option";
  description
    "EDHOC option.";
  reference
    "RFC 9668 Using Ephemeral Diffie-Hellman Over COSE (EDHOC)
              with the Constrained Application Protocol (CoAP)
              and Object Security for Constrained RESTful
              Environments (OSCORE)";
}
```

```
     identity fid-coap-option-oscore-x {
          base "schc:fid-coap-option";
          description
            "CoAP option OSCORE x field.";
          reference
            "RFC YYYY Static Context Header Compression (SCHC) for the
                      Constrained Application Protocol (CoAP) (see
                      Section 6.4)
             RFC XXXX Key Update for OSCORE (KUDOS)";
     }

     identity fid-coap-option-oscore-nonce {
          base "schc:fid-coap-option";
          description
            "CoAP option OSCORE nonce field.";
          reference
            "RFC YYYY Static Context Header Compression (SCHC) for the
                      Constrained Application Protocol (CoAP) (see
                      Section 6.4)
             RFC XXXX Key Update for OSCORE (KUDOS)";
     }

     // Function Length

     identity fl-oscore-oscore-piv-length {
          base "schc:fl-base-type";
          description
            "Size in bytes of the OSCORE Partial IV corresponding to n.";
          reference
            "RFC YYYY Static Context Header Compression (SCHC) for the
                      Constrained Application Protocol (CoAP) (see
                      Section 6.4)";
     }

     identity fl-oscore-oscore-nonce-length {
          base "schc:fl-base-type";
          description
            "Size in bytes of the OSCORE nonce corresponding to m+1.";
          reference
            "RFC YYYY Static Context Header Compression (SCHC) for the
                      Constrained Application Protocol (CoAP) (see
                      Section 6.4)
             RFC XXXX Key Update for OSCORE (KUDOS)";
     }
   }
   <CODE ENDS>
```

Figure 37: SCHC CoAP Extension YANG Data Model.

Appendix B. Document Updates

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

B.1. Version -04 to -05

* Compression of CoAP options:
  - Clarified definition of Field Descriptors in SCHC Rules.
  - Description of Option Value as possibly composed of sub-fields.
  - Both the syntactic approach and the semantics approach are possible (see draft-ietf-schc-universal-option).
  - Updated the FIDs to be consistent with the semantic approach.
* Compression of OSCORE Option:
  - Revised semantics of the x sub-field related to KUDOS.
  - Removed moot sub-fields related to KUDOS.
* Clarified OSCORE compression when using the group mode of Group OSCORE.
* Updated YANG data model.
* Updated author’s contact information.
* Fixes and editorial improvements.

B.2. Version -03 to -04

* Clarified the rationale for using the "tkl" function.
* Added the "osc.piv" function to determine the length of the OSCORE piv field.
* Consistent formulation of "tkl", "osc.x.m", and "osc.y.w".
* Explicitly stated that Field Descriptors have to be ordered in a determistic way.
* Fixed format of TV in Rule Descriptors for CoAP MID.
* Fixed order of OSCORE-related Field Descriptors in example Rules.

* Use "bit" instead of "b" as symbol for bit (per ISO/IEC 80000-13).
* Made YANG extractable.
* Updated references.
* Fixes and editorial improvements.

B.3. Version -02 to -03

* Consistent representation of "CoAP Version" 1 in example Rules.
* Split the compression of Token Length and Token into two sections.
* Disambiguated example of Rule on eliding a Uri-Path option.
* Fixed compression examples with OSCORE.
* Inherited security considerations on the YANG module from RFC 9363.
* Fixes and editorial improvements.

B.4. Version -01 to -02

* Added compression for the CoAP options Proxy-Cri and Proxy-Scheme-Number.
* Defined new IANA registry "SCHC Compression of CoAP Fields".
* Updated the YANG data model.
* Fixes and editorial improvements.

B.5. Version -00 to -01

* Fixed an example, as per the erratum with Errata ID 7623.
* Clarified building of Field Descriptor for CoAP options.
* Clarified what SCHC compression considers for CoAP options.
* Revised SCHC compression of the ETag and If-Match CoAP option.
* Revised SCHC compression of the If-None-Match CoAP option.
* Added YANG data model for the YANG module.

* Added IANA considerations.

* Fixes and editorial improvements.

Acknowledgments

The authors sincerely thank Christian Amsss, Quentin Lampin, John Preu Mattsson, Carles Gomez Montenegro, Gran Selander, Pascal Thubert, and ric Vyncke for their comments and feedback.

This work was supported by the Sweden's Innovation Agency VINNOVA within the EUREKA CELTIC-NEXT project CYPRESS; and 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

Ivn Martnez
IRISA
263 Av. Gnral Leclerc
35000 Rennes
France
Email: ivan-marino.martinez-bolivar@irisa.fr

Ana Minaburo
Consultant
Rue de Rennes
35510 Cesson-Sevigne
France
Email: anaminaburo@gmail.com