ï»¿

ï»¿

Internet Engineering Task Force (IETF)   A. Minaburo
Request for Comments: 8824   Acklio
Category: Standards Track   L. Toutain
ISSN: 2070-1721   IMT Atlantique
                                                          R. Andreasen
                                         Universidad de Buenos Aires
                                                            June 2021

             Static Context Header Compression (SCHC) for the
                  Constrained Application Protocol (CoAP)

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/UDP headers,
   this document applies SCHC to CoAP headers. The CoAP header
   structure differs from IPv6 and UDP, since CoAP uses a flexible
   header with a variable number of options, themselves of variable
   length. The CoAP message format is asymmetric: the request messages
   have a header format different from the format in the response
   messages. This specification gives guidance on applying SCHC to
   flexible headers and how to leverage the asymmetry for more efficient
   compression Rules.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc8824.

Table of Contents

1. Introduction

   The Constrained Application Protocol (CoAP) [RFC7252] is a command/
   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 this document's scope.

   CoAP is an application protocol, so CoAP compression requires
   installing common Rules between the two SCHC instances. SCHC
   compression may apply at two different levels: at IP and UDP in the
   LPWAN and another 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]. Thereby, each
   description contains the Field ID (FID), Field Length (FL), and Field

Position (FP), as well as a Direction Indicator (DI) (upstream, downstream, and bidirectional) and some associated Target Values (TVs). The DI is used for compression to give the best TV to the FID when these values differ in their transmission direction. So, a field may be described several times.

A Matching Operator (MO) is associated with each header Field Descriptor. The Rule is selected if all the MOs fit the TVs for all fields of the incoming 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 gives the method to compress and decompress each 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.

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.

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 Figures 1, 2, and 3.

In the first example, 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 C/D (SCHC Compression/Decompression; see [RFC8724]). The application communicating with the Device does not implement SCHC C/D.

```
        (Device)              (NGW)                        (App)

      +--------+                                        +--------+
      |  CoAP  |                                        |  CoAP  |
      +--------+                                        +--------+
      |  UDP   |                                        |  UDP   |
      +--------+     +----------------+                 +--------+
      |  IPv6  |     |      IPv6      |                 |  IPv6  |
      +--------+     +--------+-------+                 +--------+
      |  SCHC  |     |  SCHC  |       |                 |        |
      +-------+      +-------+        +                 +        +
      | LPWAN |      | LPWAN |        |                 |        |
      +-------+      +-------+-------+                  +--------+
        ((((LPWAN))))              ------  Internet  ------
```
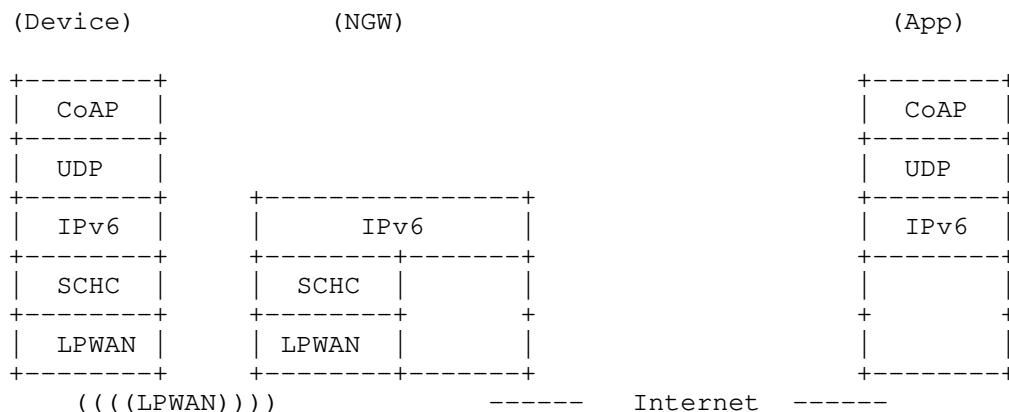
                Figure 1: Compression/Decompression at the LPWAN Boundary

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, 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. 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.

```
         (Device)                (NGW)                     (App)

         +--------+                                       +--------+
         | CoAP   |                                       | CoAP   |
         +--------+                                       +--------+
         | SCHC   |                                       | SCHC   |
         +--------+                                       +--------+
         | DTLS   |                                       | DTLS   |
         +--------+                                       +--------+
         . udp    .                                       . udp    .
         ..........       ....................            ..........
         . ipv6   .       .      ipv6        .            . ipv6   .
         ..........       ....................            ..........
         . schc   .       . schc   .        .             .        .
         ..........       ..........         .            .        .
         . lpwan  .       . lpwan  .         .            .        .
         ..........       ....................            ..........
            ((((LPWAN))))           ------   Internet  ------
```
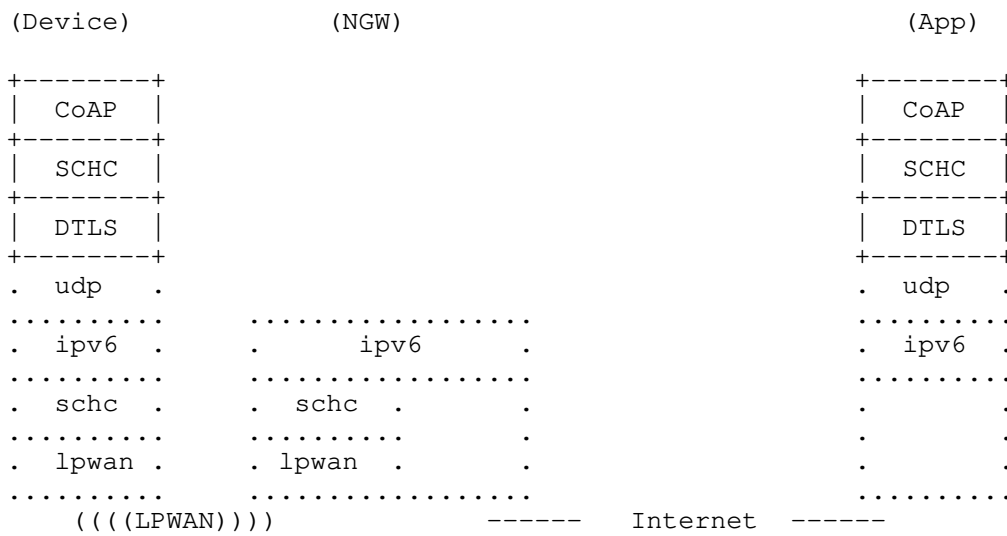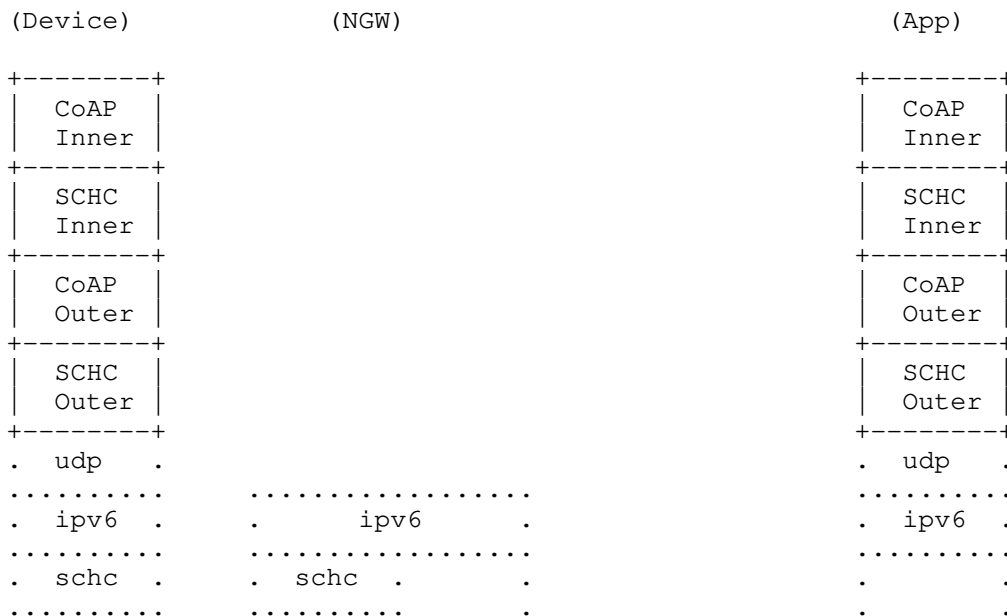
          Figure 2: Standalone CoAP End-to-End Compression/Decompression

The third example, Figure 3, shows the use of Object Security for
Constrained RESTful Environments (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 the OSCORE mechanism. Then, a second Rule compresses
the Outer header, including the OSCORE options.

```
         (Device)                (NGW)                     (App)

         +--------+                                       +--------+
         |  CoAP  |                                       |  CoAP  |
         |  Inner |                                       |  Inner |
         +--------+                                       +--------+
         |  SCHC  |                                       |  SCHC  |
         |  Inner |                                       |  Inner |
         +--------+                                       +--------+
         |  CoAP  |                                       |  CoAP  |
         |  Outer |                                       |  Outer |
         +--------+                                       +--------+
         |  SCHC  |                                       |  SCHC  |
         |  Outer |                                       |  Outer |
         +--------+                                       +--------+
         . udp    .                                       . udp    .
         ..........       ....................            ..........
         . ipv6   .       .      ipv6        .            . ipv6   .
         ..........       ....................            ..........
         . schc   .       . schc   .        .             .        .
         ..........       ..........         .            .        .
```

```
          .  lpwan  .        .  lpwan  .      .                    .        .
          ..........        .................              ..........
            ((((LPWAN))))              ------    Internet  ------

                 Figure 3: OSCORE Compression/Decompression
```
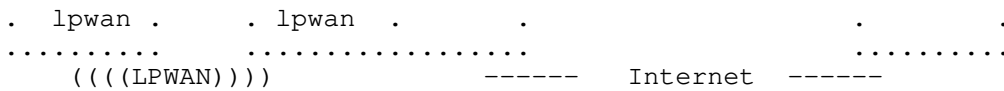
In the case of several SCHC instances, as shown in Figures 2 and 3,
the Rules may come from different provisioning domains.

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 applies the same description and
compression/decompression techniques as the technique used for IP and
UDP, 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.

As 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, and
where 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; the headers are different
   for a request or a response. For example, the Uri-Path option is
   mandatory in the request, and it might not be present in the
   response. A request might contain an Accept option, and the
   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 by compression,
   and the answer may use one single bit to carry either the ACK or
   Reset (RST) type. The field Code has the same behavior: the 0.0X
   code format value in the request and the Y.ZZ code format in the
   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, in a Uri-Path or Uri-
   Query, the Token size may vary from 0 to 8 bytes, and the CoAP
   options use the Type-Length-Value encoding format.

   When doing SCHC compression of a variable-length field,
   Section 7.4.2 of [RFC8724] offers the option of defining a
   function for the Field Length in the Field Descriptor to know 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 the CoAP headers. It is found
  typically for elements of a URI (path or queries). 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, thereby removing the MO's ambiguity.

* 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 compression of the different CoAP header
   fields. CoAP compression with SCHC follows the information provided
   in Section 7.1 of [RFC8724].

4.1. CoAP Version Field

   The CoAP version is bidirectional and MUST be elided during SCHC
   compression, since it always contains the same value. In the future,
   or if a new version of CoAP is defined, new Rules will be needed to
   avoid ambiguities between versions.

4.2. CoAP Type Field

   CoAP [RFC7252] has four types of messages: two requests (CON, NON),
   one response (ACK), and one empty message (RST).

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

4.3. CoAP Code Field

   The Code field, defined in an IANA registry [RFC7252], indicates the
   Request Method used in CoAP. 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 request codes with the 0 class and (2) the
   response values. SCHC will use the DI to identify the correct value
   in the packet.

   If the Device only implements a CoAP client, SCHC compression may
   reduce the request code to the set of requests the client can
   process.

   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.

4.4. 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].

4.5. CoAP Token Fields

   CoAP defines the Token using two CoAP fields: Token Length in the
   mandatory header and Token Value directly following the mandatory
   CoAP header.

   SCHC processes the Token Length as it would any header field. If the
   value does not change, the size can be stored in the TV and elided
   during the transmission. Otherwise, SCHC will send the Token Length

in the Compression Residue.

For the Token Value, SCHC MUST NOT send it as variable-length data in
the Compression Residue, to avoid ambiguity with the Token Length.
Therefore, SCHC MUST use the Token Length value to define the size of
the Compression Residue. 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.

5. CoAP Options

CoAP defines options placed after the basic header, ordered by option
number; see [RFC7252]. Each Option instance in a message uses the
format Delta-Type (D-T), Length (L), Value (V). The SCHC Rule builds
the description of the option by using the following:

* in the FID: the option number built from the D-T;

* in the TV: the option value; and

* for the Option Length: the information provided in Sections 7.4.1
   and 7.4.2 of [RFC8724].

When the Option Length has a well-known size, the Rule may keep the
length value. Therefore, SCHC compression does not send it.
Otherwise, SCHC compression carries the length of the Compression
Residue, in addition to the Compression Residue value.

CoAP requests and responses do not include the same options. So,
compression Rules may reflect this asymmetry by tagging the DI.

Note that length coding differs between CoAP options and SCHC
variable size Compression Residue.

The following sections present how SCHC compresses some specific CoAP
options.

If CoAP introduces a new option, the SCHC Rules MAY be updated, and
the new FID description MUST be assigned to allow its compression.
Otherwise, if no Rule describes this new option, SCHC compression is
not achieved, and SCHC sends the CoAP header without compression.

5.1. CoAP Content and Accept Options

If the client expects a single value, it can be stored in the TV and
elided during the transmission. 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 (fixed or variable length).

5.2. 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, SCHC will use the "value-sent" MO, and the Compression
Residue will send these options' values.

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

The Uri-Path and Uri-Query fields are repeatable options; this means
that in the CoAP header, they may appear several times with different
values. The SCHC Rule description uses the FP to distinguish the
different instances in the path.

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

In Table 1, SCHC can use a single bit in the Compression Residue to code one of the two paths. If regrouping were not allowed, 2 bits in the Compression Residue would be needed. SCHC sends the third path element as a variable size in the Compression Residue.

+==========+=====+====+====+==========+=========+=============+
| Field | FL | FP | DI | TV | MO | CDA |
+==========+=====+====+====+==========+=========+=============+
| Uri-Path | | 1 | Up | ["/a/b", | match- | mapping-sent|
| | | | | "/c/d"] | mapping | |
+----------+-----+----+----+----------+---------+-------------+
| Uri-Path | var | 3 | Up | | ignore | value-sent |
+----------+-----+----+----+----------+---------+-------------+

                   Table 1: Complex Path Example

The length of Uri-Path and Uri-Query may be known when the Rule is defined. In any case, SCHC MUST set the Field Length to a variable value. The Compression Residue size is expressed in bytes.

SCHC compression can use the MSB MO to a Uri-Path or Uri-Query element. However, attention to the length is important because the MSB value is in bits, and the size MUST always be a multiple of 8 bits.

The length sent at the beginning of a variable-length Compression Residue indicates the LSB's size in bytes.

For instance, for a CORECONF path /c/X6?k=eth0, the Rule description can be as follows (Table 2):

+===========+=====+====+====+======+=========+============+
| Field | FL | FP | DI | TV | MO | CDA |
+===========+=====+====+====+======+=========+============+
| Uri-Path | | 1 | Up | "c" | equal | not-sent |
+-----------+-----+----+----+------+---------+------------+
| Uri-Path | var | 2 | Up | | ignore | value-sent |
+-----------+-----+----+----+------+---------+------------+
| Uri-Query | var | 1 | Up | "k=" | MSB(16) | LSB |
+-----------+-----+----+----+------+---------+------------+

                 Table 2: CORECONF URI Compression

Table 2 shows the Rule description for a Uri-Path and a Uri-Query. SCHC compresses the first part of the Uri-Path with a "not-sent" CDA. SCHC will send the second element of the Uri-Path with the length (i.e., 0x2 "X6") followed by the query option (i.e., 0x4 "eth0").

5.3.1. Variable Number of Path or Query Elements

SCHC fixed the number of Uri-Path or Uri-Query elements in a Rule at the Rule creation time. If the number 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].

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

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

5.5. CoAP Option ETag, If-Match, If-None-Match, Location-Path, and Location-Query Fields

A Rule entry cannot store these fields' values. The Rule description MUST always send these values in the Compression Residue.

6. SCHC Compression of CoAP Extensions

6.1. Block

When a packet uses a Block option [RFC7959], SCHC compression MUST send its content in the Compression Residue. The SCHC Rule describes an empty TV with the MO set to "ignore" and the CDA set to "value-sent". The Block option allows 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 will not define the TV but will set 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 Observe Option MAY use a RST message to inform a server that the client does not require the Observe response, a specific SCHC Rule SHOULD exist to allow the message's compression with the RST type.

6.3. No-Response

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

Otherwise, if the value is changing over time, the SCHC Rule will set the MO to "ignore" and the CDA to "value-sent". The Rule may also use a "match-mapping" MO to compress this option.

6.4. OSCORE

OSCORE [RFC8613] defines end-to-end protection for CoAP messages. This section describes how SCHC Rules can be applied to compress OSCORE-protected messages.

Figure 4 shows the OSCORE option value encoding defined in Section 6.1 of [RFC8613], where the first byte specifies the content of the OSCORE options using flags. The three most significant bits of this byte are reserved and always set to 0. Bit h, when set, indicates the presence of the kid context field in the option. Bit k, when set, indicates the presence of a kid field. The three least significant bits, n, indicate the length of the piv (Partial Initialization Vector) field in bytes. When n = 0, no piv is present.

```
        0 1 2 3 4 5 6 7 <--------- n bytes ------------->
       +-+-+-+-+-+-+-+-+-------------------------------
       |0 0 0|h|k|  n  |     Partial IV (if any) ...
       +-+-+-+-+-+-+-+-+-------------------------------
       |             |                              |
       |<-- CoAP  -->|<------ CoAP OSCORE_piv ------> |
          OSCORE_flags

       <- 1 byte -> <------ s bytes ----->
       +------------+---------------------+----------------------+
       | s (if any) | kid context (if any) | kid (if any)    ... |
```

```
    +------------+--------------------+--------------------+
    |            |                    |                    |
    | <------ CoAP OSCORE_kidctx ----->|<-- CoAP OSCORE_kid -->|
    |            |                    |                    |
```
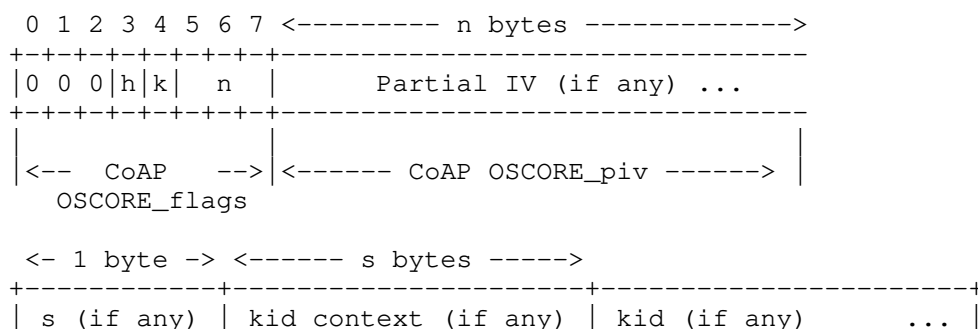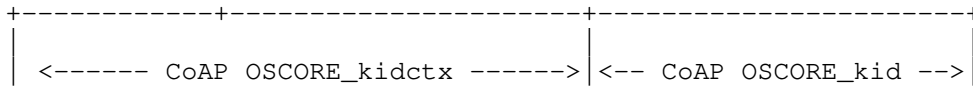
                    Figure 4: OSCORE Option

The flag byte is followed by the piv field, the kid context field,
and the kid field, in that order, and, if present, the kid context
field's length (in bytes) is encoded in the first byte, denoted by
"s".

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

* CoAP OSCORE_flags

* CoAP OSCORE_piv

* CoAP OSCORE_kidctx

* CoAP OSCORE_kid

Figure 4 shows the OSCORE option format with those four fields
superimposed on it. Note that the CoAP OSCORE_kidctx field directly
includes the size octet, s.

7. Examples of CoAP Header Compression

7.1. 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.

RuleID 1							
Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP version	2	1	Bi	01	equal	not-sent	
CoAP Type	2	1	Dw	CON	equal	not-sent	
CoAP Type	2	1	Up	[ACK, RST]	match-mapping	matching-sent	T
CoAP TKL	4	1	Bi	0	equal	not-sent	
CoAP Code	8	1	Bi	[0.00, ... 5.05]	match-mapping	matching-sent	CC CCC
CoAP MID	16	1	Bi	0000	MSB(7)	LSB	MID
CoAP Uri-Path	var	1	Dw	path	equal 1	not-sent	

        Table 3: CoAP Context to Compress Header without Token

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 indicated in the TV and elided with the CDA

"not-sent".

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

   Note that a client located in an Application Server sending a request
   to a server located in the Device 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.

7.2. OSCORE Compression

   OSCORE aims to solve the problem of end-to-end encryption for CoAP
   messages. Therefore, the goal is to hide the message as much as
   possible while still enabling proxy operation.

   Conceptually, this is achieved by splitting the CoAP message into an
   Inner Plaintext and Outer OSCORE message. The Inner Plaintext
   contains sensitive information that is not necessary for proxy
   operation. However, it is part of the message that can be encrypted
   until it reaches its end destination. The Outer Message acts as a
   shell matching the regular CoAP message format and includes all
   options and information needed for proxy operation and caching.
   Figure 5 below illustrates this analysis.

   CoAP arranges the options into one of three classes, each granted a
   specific type of protection by the protocol:

   Class E: Encrypted options moved to the Inner Plaintext.

   Class I: Integrity-protected options included in the Additional
      Authenticated Data (AAD) for the encryption of the Plaintext but
      otherwise left untouched in the Outer Message.

   Class U: Unprotected options left untouched in the Outer Message.

   These classes point out that the Outer option contains the OSCORE
   option and that the message is OSCORE protected; this option carries
   the information necessary to retrieve the Security Context. The
   endpoint will use this Security Context to decrypt the message
   correctly.

```
                       Original CoAP Packet
                 +-+-+---+-------+--------------+
                 |v|t|TKL| code  | Message ID   |
                 +-+-+---+-------+--------------+....+
                 | Token                             |
                 +------------------------------.....+
                 | Options (IEU)            |
                 .                          .
                 .                          .
                 +------+------------------+
                 | 0xFF |
                 +------+----------------------+
                 |                             |
                 |                             |
                 |      Payload                |
                 |                             |
                 +-----------------------------+
                        /              \
                       /                \
                      /                  \
                     /                    \
       Outer Header       v                    v  Plaintext
    +-+-+---+--------+--------------+         +-------+
    |v|t|TKL|new code| Message ID   |         | code  |
    +-+-+--+--------+--------------+....+     +-------+-----......+
    | Token                             |     | Options (E)      |
    +------------------------------.....+     +-------+------.....+
    | Options (IU)             |               | 0xFF |
```

```
.                             .        +-------+-----------+
. OSCORE Option               .        |       |           |
+------+-------------------+            |  Payload          |
| 0xFF |                               |       |           |
+------+                               +-------------------+
```
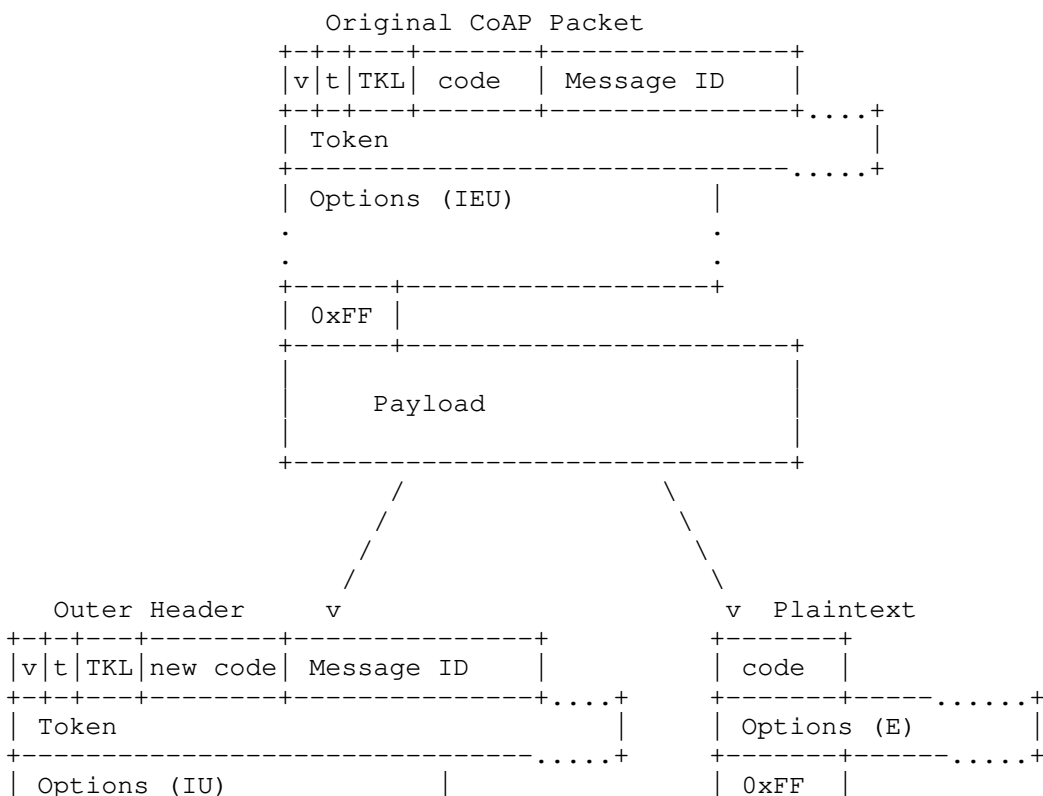
Figure 5: CoAP Packet Split into OSCORE Outer Header and Plaintext

Figure 5 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 default dummy value. As seen in Sections 4.1.3.5 and 4.2 of
[RFC8613], the message code is replaced by POST for requests and
Changed for responses when CoAP is not using the Observe Option. If
CoAP uses Observe, the OSCORE message code is replaced by FETCH for
requests and Content for responses.

The first byte of the Plaintext contains the original packet code,
followed by the message code, the class E options, and, if present,
the original message payload preceded by its payload marker.

An Authenticated Encryption with Associated Data (AEAD) algorithm now
encrypts the Plaintext. This integrity-protects the Security Context
parameters and, eventually, any class I options from the Outer
header. The resulting ciphertext becomes the new payload of the
OSCORE message, as illustrated in Figure 6.

As defined in [RFC5116], this ciphertext is the encrypted Plaintext's
concatenation of 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.

```
     Outer Header
  +-+-+---+--------+--------------+
  |v|t|TKL|new code| Message ID   |
  +-+-+---+--------+--------------+....+
  | Token                             |
  +------------------------------.....+
  | Options (IU)          |
  .                       .
  . OSCORE Option         .
  +------+-------------------+
  | 0xFF |
  +------+--------------------------+
  |                                 |
  |  Ciphertext: Encrypted Inner    |
  |          Header and Payload     |
  |          + Authentication Tag   |
  |                                 |
  +---------------------------------+
```
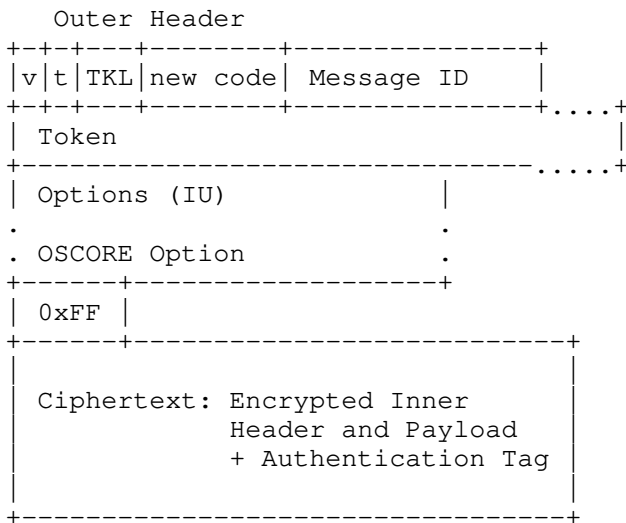
Figure 6: OSCORE Message

The SCHC compression scheme consists of compressing both the
Plaintext before encryption and the resulting OSCORE message after
encryption; see Figure 7.

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

```
     Outer Message                        OSCORE Plaintext
  +-+-+---+--------+--------------+        +-------+
  |v|t|TKL|new code| Message ID   |        | code  |
  +-+-+---+--------+--------------+....+    +-------+-----......+
  | Token                             |    | Options (E)      |
  +------------------------------.....+    +-------+------.....+
```

```
 | Options (IU)                 |           | 0xFF  |
 .                              .           +-------+-----------+
 . OSCORE Option                .           |                   |
 +------+-------------------+                | Payload           |
 | 0xFF |                                    |                   |
 +------+-----------+                        +-------------------+
 |   Ciphertext     |<---------\             |
 |                  |          |             v
 +------------------+          |      +------------------+
          |                    |      | Inner SCHC       |
          v                    |      | Compression      |
     +------------------+      |      +------------------+
     |    Outer SCHC    |      |             |
     |    Compression   |      |             v
     +------------------+      |      +-------+
          |                    |      |RuleID |
          v                    |      +-------+-----------+
     +--------+         +------------+ |Compression Residue|
     |RuleID' |         | Encryption | <--  +----------+--------+
     +--------+-----------+ +------------+   |                   |
     |Compression Residue'|                  | Payload           |
     +-----------+--------+                  |                   |
     |   Ciphertext      |                   +-------------------+
     |                   |
     +-------------------+
```
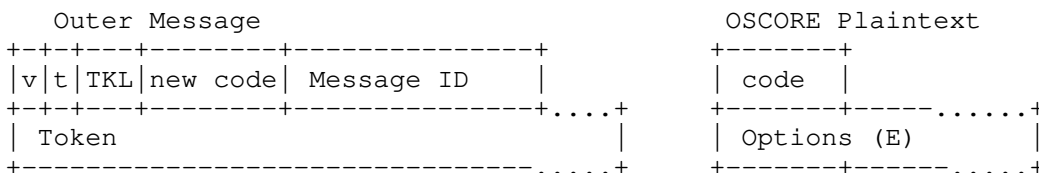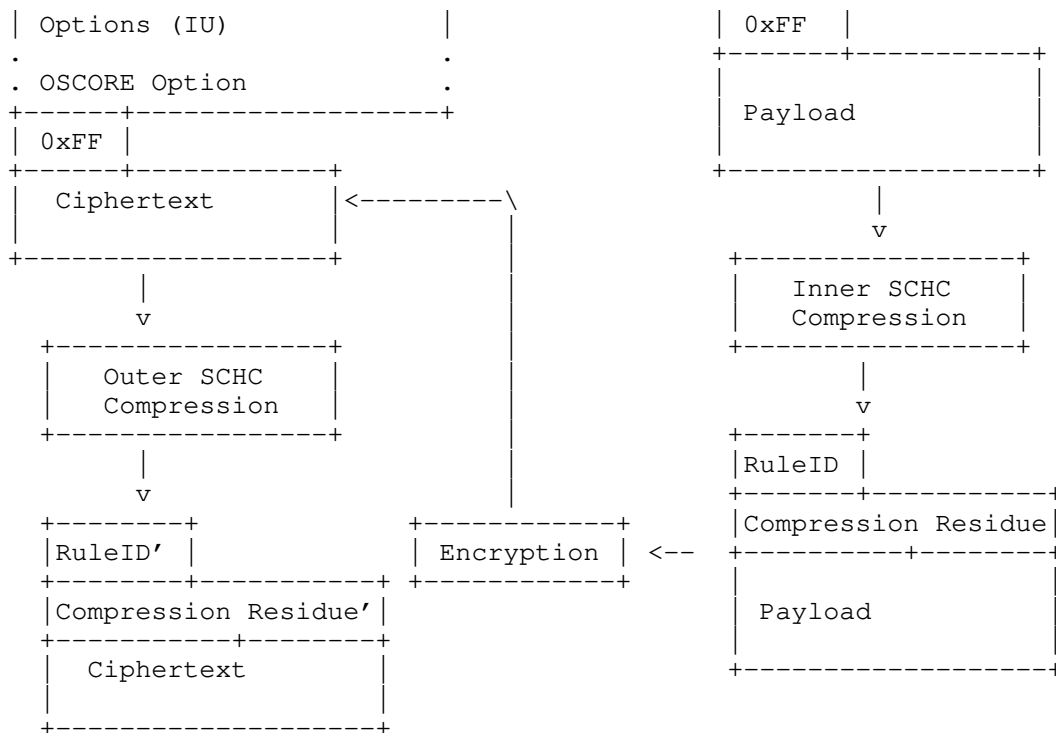
                    Figure 7: OSCORE Compression Diagram

   Note that since the corresponding endpoint can only decrypt the Inner
   part of the message, this endpoint will also have to implement Inner
   SCHC Compression/Decompression.

7.3. Example OSCORE Compression

   This section gives an example with a GET request and its consequent
   Content response from a Device-based CoAP client to 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 with SCHC +
   CoAP performance are also listed. This example gives an
   approximation of the cost of security with SCHC-OSCORE.

   Our first CoAP message is the GET request in Figure 8.

   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 8: CoAP GET Request

   Its corresponding response is the Content response in Figure 9.

   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:
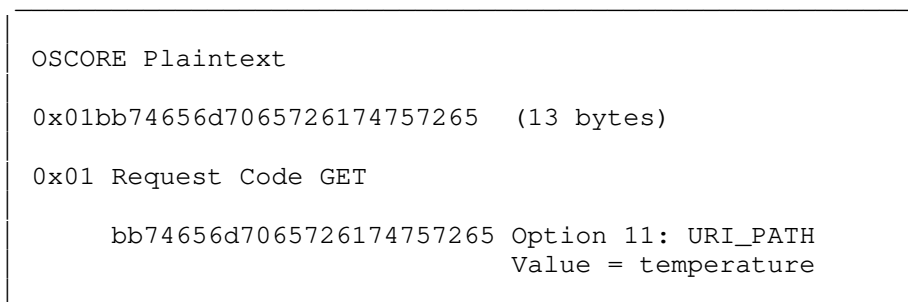0x32332043

Original message length:    10 bytes
```

Figure 9: CoAP Content Response

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

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
RuleID 0							
CoAP Code	8	1	Up	1	equal	not-sent	
CoAP Code	8	1	Dw	[69,132]	match-mapping	mapping-sent	c
CoAP Uri-Path		1	Up	temperature	equal	not-sent	

Table 4: Inner SCHC Rule

Figure 10 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.

```
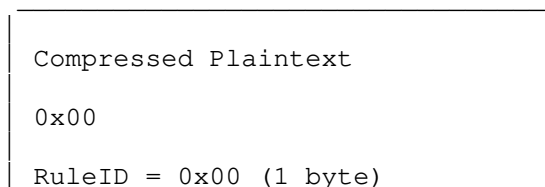 _____
|                                        |
|  OSCORE Plaintext                      |
|                                        |
|  0x01bb74656d7065726174757265  (13 bytes)
|                                        |
|  0x01 Request Code GET                 |
|                                        |
|      bb74656d7065726174757265 Option 11: URI_PATH
|                               Value = temperature
|_____|

                      |
                      |
                      |   Inner SCHC Compression
                      |
                      v

          _____
         |                                  |
         |  Compressed Plaintext            |
         |                                  |
         |  0x00                            |
         |                                  |
         |  RuleID = 0x00 (1 byte)          |
```

```
          |        (No Compression Residue)        |
          |_____|

                            |
                            |  AEAD Encryption
                            |    (piv = 0x04)
                            v
       _____
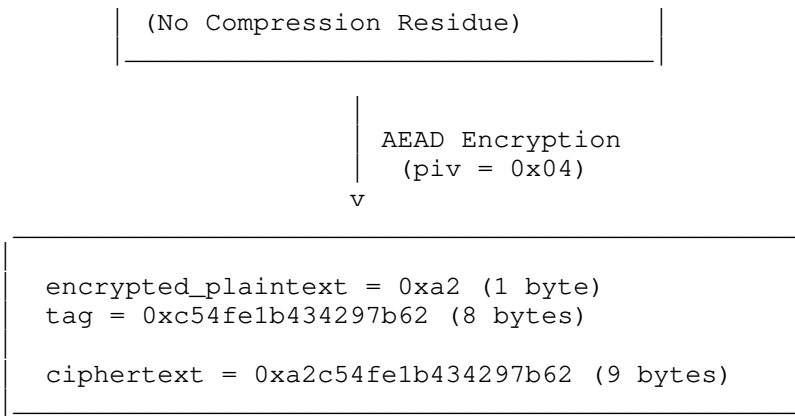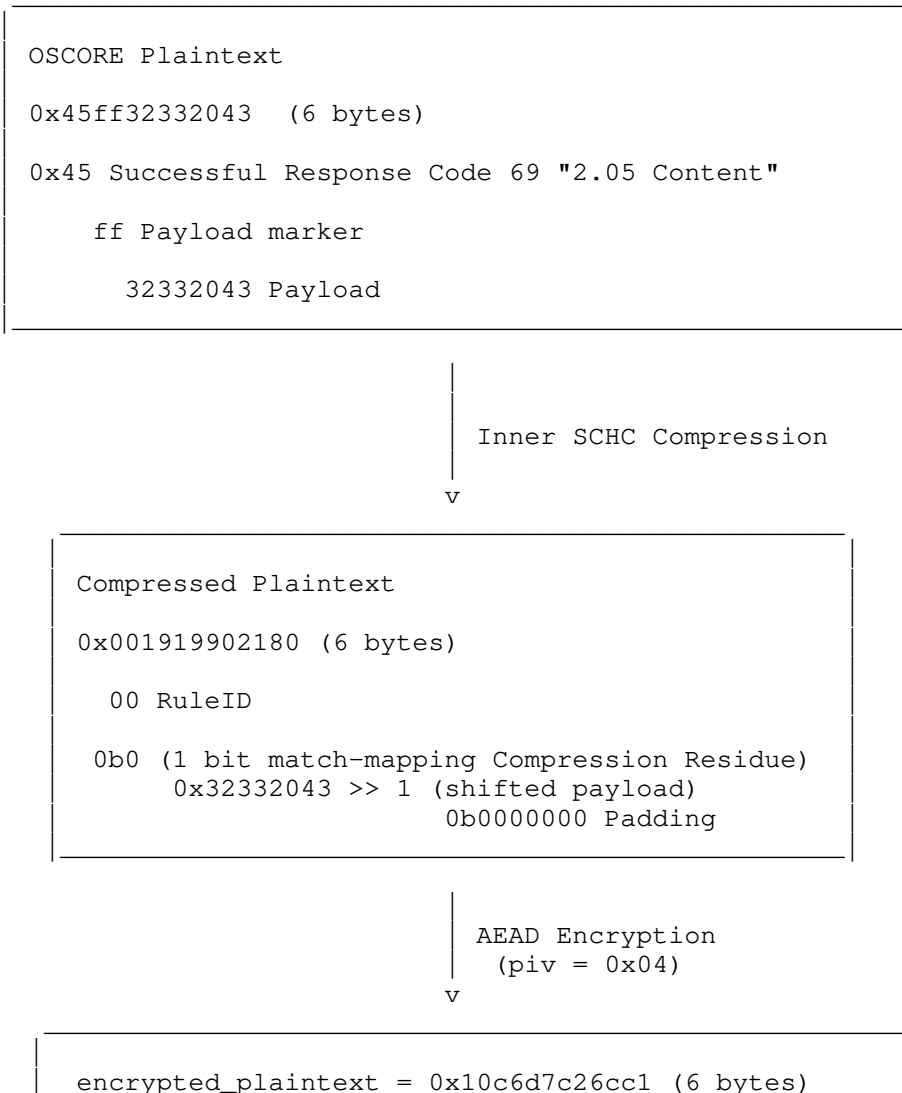      |                                                 |
      |    encrypted_plaintext = 0xa2 (1 byte)          |
      |    tag = 0xc54fe1b434297b62 (8 bytes)           |
      |                                                 |
      |    ciphertext = 0xa2c54fe1b434297b62 (9 bytes)  |
      |_____|
```

       Figure 10: 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 in total message
length, limiting the amount of compression that can be achieved and
playing into the cost of adding security to the exchange.

Figure 11 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 brings security to the transmission.

```
       _____
      |                                                |
      |  OSCORE Plaintext                              |
      |                                                |
      |  0x45ff32332043   (6 bytes)                    |
      |                                                |
      |  0x45 Successful Response Code 69 "2.05 Content"|
      |                                                |
      |     ff Payload marker                          |
      |                                                |
      |       32332043 Payload                         |
      |_____|

                            |
                            |
                            |  Inner SCHC Compression
                            |
                            v
       _____
      |                                                |
      |  Compressed Plaintext                          |
      |                                                |
      |  0x001919902180 (6 bytes)                      |
      |                                                |
      |    00 RuleID                                   |
      |                                                |
      |   0b0 (1 bit match-mapping Compression Residue)|
      |        0x32332043 >> 1 (shifted payload)       |
      |                      0b0000000 Padding         |
      |_____|

                            |
                            |  AEAD Encryption
                            |    (piv = 0x04)
                            v
       _____
      |                                                     |
      |  encrypted_plaintext = 0x10c6d7c26cc1 (6 bytes)     |
```

```
|   tag = 0xe9aef3f2461e0c29 (8 bytes)                    |
|                                                         |
|   ciphertext = 0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)|
|_____|
```

Figure 11: Plaintext Compression and Encryption for Content Response

The Outer SCHC Rule (Table 5) must process the OSCORE options fields.
Figures 12 and 13 show a dump of the OSCORE messages generated from
the example messages. They include 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.

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
RuleID 0							
CoAP version	2	1	Bi	01	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	0000	MSB(12)	LSB	MMMM
CoAP Token	tkl	1	Bi	0x80	MSB(5)	LSB	TTT
CoAP OSCORE_flags	8	1	Up	0x09	equal	not-sent	
CoAP OSCORE_piv	var	1	Up	0x00	MSB(4)	LSB	PPPP
CoAP OSCORE_kid	var	1	Up	0x636c69656e70	MSB(52)	LSB	KKKK
CoAP OSCORE_kidctx	var	1	Bi	b''	equal	not-sent	
CoAP OSCORE_flags	8	1	Dw	b''	equal	not-sent	
CoAP OSCORE_piv	var	1	Dw	b''	equal	not-sent	
CoAP OSCORE_kid	var	1	Dw	b''	equal	not-sent	

Table 5: Outer SCHC Rule

Protected message:
==================
0x4102000182d8080904636c69656e74ffa2c54fe1b434297b62
(25 bytes)

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

0x0001 = mid

```
0x82 = token

Options:
0xd8080904636c69656e74 (10 bytes)
Option 21: OBJECT_SECURITY
Value = 0x0904636c69656e74
          09 = 000 0 1 001 flag byte
                 h k   n
            04 piv
               636c69656e74 kid

0xFF  Payload marker
Payload:
0xa2c54fe1b434297b62 (9 bytes)
```

          Figure 12: Protected and Inner SCHC Compressed GET Request

```
Protected message:
==================
0x6144000182d008ff10c6d7c26cc1e9aef3f2461e0c29
(22 bytes)


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

0x0001 = mid
0x82 = token

Options:
0xd008 (2 bytes)
Option 21: OBJECT_SECURITY
Value = b''

0xFF  Payload marker
Payload:
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)
```

    Figure 13: 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 a MO of "equal"
and a CDA of "not-sent". This SCHC definition 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 choices of CoAP options
that can be used in the exchange, since the values of these choices
are dependent on specific options.

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 frequency is low, such as LPWAN
technologies. Note that compressing the sequence numbers 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 context field MAY be masked
off with an "LSB" CDA. The rest of the field could have additional
bits masked off or have the whole field fixed with a MO of "equal"
and a CDA of "not-sent". The same holds for the kid field.

The Outer Rule of Table 5 is applied to the example GET request and
Content response. Figures 14 and 15 show the resulting messages.

```
Compressed message:
=================
0x001489458a9fc3686852f6c4 (12 bytes)
0x00 RuleID
    1489 Compression Residue
        458a9fc3686852f6c4 Padded payload

Compression Residue:
0b 0001 010 0100 0100 (15 bits -> 2 bytes with padding)
    mid tkn piv  kid

Payload
0xa2c54fe1b434297b62 (9 bytes)

Compressed message length: 12 bytes
```

              Figure 14: SCHC-OSCORE Compressed GET Request

```
Compressed message:
==================
0x0014218daf84d983d35de7e48c3c1852 (16 bytes)
0x00 RuleID
    14 Compression Residue
      218daf84d983d35de7e48c3c1852 Padded payload
Compression Residue:
0b0001 010 (7 bits -> 1 byte with padding)
  mid  tkn

Payload
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)

Compressed message length: 16 bytes
```

            Figure 15: SCHC-OSCORE Compressed Content Response

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

RuleID 1							
Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP version	2	1	Bi	01	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	0000	MSB(12)	LSB	MMMM
CoAP Token	tkl	1	Bi	0x80	MSB(5)	LSB	TTT
CoAP Uri-Path		1	Up	temperature	equal	not-sent	

```
+--------+---+--+--+----------+--------------+-------------+======+
```

<div align="center">Table 6: SCHC-CoAP Rule (No OSCORE)</div>

The Rule in Table 6 yields the SCHC compression results as shown in
Figure 16 for the request and Figure 17 for the response.

Compressed message:
==================
0x0114
0x01 = RuleID

Compression Residue:
0b00010100 (1 byte)

Compressed message length: 2 bytes

<div align="center">Figure 16: CoAP GET Compressed without OSCORE</div>

Compressed message:
==================
0x010a32332043
0x01 = RuleID

Compression Residue:
0b00001010 (1 byte)

Payload
0x32332043

Compressed message length: 6 bytes

<div align="center">Figure 17: CoAP Content Compressed without OSCORE</div>

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

8. IANA Considerations

    This document has no IANA actions.

9. 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 (OSCORE, DTLS, etc.), it is necessary to use a
    Layer 2 security mechanism to protect the SCHC messages.

    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 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 uncompressible value). The key-renewal operation itself requires
    several message exchanges and requires energy-intensive computation,
    but the optimal trade-off will depend on the specifics of the Device
    and expected usage patterns.

    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 the original compressed
representation possessed, 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 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. Similar procedures may be appropriate to signal Rule
updates when other message-protection mechanisms are in use.

10. Normative References

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

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

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

   [RFC7641] Hartke, K., "Observing Resources in the Constrained
              Application Protocol (CoAP)", RFC 7641,
              DOI 10.17487/RFC7641, September 2015,
              <https://www.rfc-editor.org/info/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/info/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/info/rfc7967>.

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

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

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

Acknowledgements

Authors' Addresses

Ana Minaburo
Acklio
1137A avenue des Champs Blancs
35510 Cesson-Sevigne Cedex
France

Email: ana@ackl.io

Laurent Toutain
Institut MINES TELECOM; IMT Atlantique
CS 17607
2 rue de la Chataigneraie
35576 Cesson-Sevigne Cedex
France

Email: Laurent.Toutain@imt-atlantique.fr

Ricardo Andreasen
Universidad de Buenos Aires
Av. Paseo Colon 850
C1063ACV Ciudad Autonoma de Buenos Aires
Argentina

Email: randreasen@fi.uba.ar