lpwan Working Group A. Minaburo
Internet-Draft Acklio
Intended status: Standards Track L. Toutain
Expires: April 23, 2021 Institut MINES TELECOM; IMT Atlantique
R. Andreasen
Universidad de Buenos Aires
October 20, 2020

LPWAN Static Context Header Compression (SCHC) for CoAP
draft-ietf-lpwan-coap-static-context-hc-16

Abstract

This draft defines how Static Context Header Compression (SCHC) can be applied to the Constrained Application Protocol (CoAP). SCHC is a header compression mechanism adapted for constrained devices. SCHC uses a static description of the header to reduce the redundancy and size of the header's information. While RFC 8724 describes the SCHC compression and fragmentation framework, and its application for IPv6/UDP headers, this document applies SCHC for 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 protocol messages format is asymmetric: the request messages have a header format different from the one 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

Table of Contents

1. Introduction

CoAP [rfc7252] is a command/response protocol designed for micro-controllers with a small amount of RAM and ROM and is optimized for REST-based (Representational state transfer) services. Although CoAP was designed for Low-Power Wireless Personal Area Networks (6LoWPAN), a CoAP header's size is still too large for LPWAN (Low Power Wide Area Networks) and some compression of the CoAP header is required either to increase performances or allow CoAP other some LPWAN technologies.

The [rfc8724] defines SCHC, a header compression mechanism for the LPWAN network based on a static context. Section 5 of the [rfc8724] explains the architecture where compression and decompression are done. The SCHC compression scheme assumes as a prerequisite that the static context is known to both endpoints 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: one to compress IP and UDP in the LPWAN network and another at the application level for CoAP. These two compressions may be independent. Both follow the same principle described in RFC8724. SCHC rules driving the compression/decompression are different and may be managed by different entities. The [rfc8724] describes how the IP and UDP headers may be compressed. This document specifies how the SCHC compression rules can be applied to CoAP traffic.

SCHC compresses and decompresses headers based on shared contexts between devices.
Each context consists of multiple Rules. Each Rule can match header fields and specific values or ranges of values.
If a Rule matches, the matched header fields are replaced by the RuleID and some residual bits. Thus, different Rules may correspond to divers protocols packets that a device expects to send or receive.

A Rule describes the packets's entire header with an ordered list of fields descriptions; see section 7 of [rfc8724]. Thereby each description contains the field ID (FID), its length (FL), and its position (FP), a direction indicator (DI) (upstream, downstream, and bidirectional), and some associated Target Values (TV). The

direction indicator is used for compression to give the best TV to the FID when these values differ in the transmission direction. So a field may be described several times depending on the asymmetry of its possible TVs.

A Matching Operator (MO) is associated with each header field description.
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 unknown to the SCHC compressor.

In that case, a Compression/Decompression Action (CDA) associated with each field give the method to compress and decompress each field. Compression mainly results in one of 4 actions:

o send the field value,

o send nothing,

o send some least significant bits of the field or

o send an index.

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

SCHC is a general mechanism applied to different protocols, the exact Rules to be used depending on the protocol and the application. Section 10 of the [rfc8724] describes the compression scheme for IPv6 and UDP headers.
This document targets the 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

The SCHC Compression Rules can be applied to CoAP headers. SCHC Compression of the CoAP header 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

In the first example, Figure 1, a Rule compresses the complete header stack from IPv6 to CoAP. In this case, SCHC C/D (Static Context Header Compression Compressor/Decompressor) is performed at the device and the application. The host 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

The SCHC can be viewed as a layer above layer 2. This layer received non-encrypted packets and can apply compression rule to all the headers. On the other end, the NGW receives the SCHC packet and reconstructs the headers from the rule, identified by its ID and the header residues. The result is a regular IPv6 packet that can be forwarded toward the destination. The same process applies in the other direction. A not encrypted packet arrived at the NGW, thanks to 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, the SCHC compression is applied in the CoAP layer, compressing the CoAP header independently of the other layers. The RuleID, the 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] document (represented in dotted lines).

This use case needs an end-to-end context initialization between the device and the application and is out-of-scope of 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

In the third example, Figure 3, the Object Security for Constrained RESTful Environments (OSCORE) [rfc8613] is used. In this case, two rulesets are used to compress the CoAP message. A first ruleset focused on the inner header compresses it. The result is encrypted using the OSCORE mechanism. A second ruleset 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 Figure 3 and Figure 3, the rulesets may come from different provisioning domains.

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

3. CoAP Headers compressed with SCHC

The use of SCHC over the CoAP header uses the same description and compression/decompression techniques like the one for IP and UDP explained in the [rfc8724]. For CoAP, SCHC Rules description uses the direction information to optimize the compression by reducing the number of Rules needed to compress headers. The field description MAY define both request/response headers and target values in the same Rule, using the DI (direction indicator) to make the difference.

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. 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 on the following aspects:

- The CoAP protocol 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 may not be present in the response. A request may contain an Accept option, and the response may include a Content-Format option. In comparison, IPv6 and UDP returning path swap the value of some fields in the header. But all the directions have the same fields (e.g., source and destination address fields).

  The [rfc8724] defines the use of a Direction Indicator (DI) in the Field Descriptor, which allows a single Rule to process a message headers 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 matching list in the TV to limit the range of expected values in a particular direction and therefore reduce the Compression Residue's size. Through the Direction Indicator (DI), a field description in the Rules splits the possible field value into two parts, one for each direction. For instance, if a client sends only CON requests, the type can be elided by compression, and the answer may use one single bit to carry either the ACK or RST type. The field Code has the same behavior, the 0.0X code format value in the request, and Y.ZZ code format in the response.

- Headers in IPv6 and UDP have a fixed size. The size is not sent as part of the Compression Residue but is defined in the Rule. Some CoAP header fields have variable lengths, so the length is also specified in the Field Description. For example, the Token size may vary from 0 to 8 bytes. And the CoAP options have a variable length since they use the Type-Length-Value encoding format, as URI-path or URI-query.

  Section 7.5.2 from [rfc8724] offers the possibility to define a function for the Field length in the Field Description to know the length before compression. When doing SCHC compression of a variable-length field,
  if the field size is unknown, the Field Length in the Rule is set as variable, and the size is sent with the Compression Residue.

- A field can appear several times in the CoAP headers. This is typical for elements of a URI (path or queries). The SCHC specification [rfc8724] allows a Field ID to appear several times in the Rule and uses the Field Position (FP) to identify the correct instance, and thereby removing the ambiguity of the matching operation.

- Field sizes defined in the CoAP protocol can be too large regarding LPWAN traffic constraints. This is particularly true for the Message-ID field and the Token field. SCHC uses different Matching operators (MO) to perform the compression. See section 7.4 of [rfc8724]. In this case, the Most Significant Bits (MSB) MO can be applied to reduce the information carried on LPWANs.

4. Compression of CoAP header fields

This section discusses the compression of the different CoAP header fields. The CoAP compression with SCHC follows the Section 7.1 of [rfc8724].

4.1. CoAP version field

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

4.2. CoAP type field

The CoAP Protocol [rfc7252] has four types of messages: two requests (CON, NON), one response (ACK), and one empty message (RST).

The field SHOULD be elided if, for instance, a client is sending only NON or only CON messages. For the RST message, a dedicated Rule may be needed. For other usages, a mapping list can be used.

4.3. CoAP code field

The code field indicates the Request Method used in CoAP, an IANA registry [rfc7252]. 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, the set of code values can be split into two parts, the request codes with the 0 class and the response values.

If the device only implements a CoAP client, the request code can be reduced to the set of requests the client can to process.

A mapping list can be used for known values. The field cannot be compressed for other values, and the value needs to be sent in the Compression Residue.

4.4. CoAP Message ID field

The Message ID field can be compressed with the MSB(x) MO and the Least Significant Bits (LSB) CDA. See section 7.4 of [rfc8724].

4.5. CoAP Token fields

A Token is defined through two CoAP fields, Token Length in the mandatory header and Token Value directly following the mandatory CoAP header.

Token Length is processed as any protocol field. If the value does not change, the size can be stored in the TV and elided during the transmission. Otherwise, it will have to be sent in the Compression Residue.

Token Value MUST NOT be sent as a variable-length residue to avoid ambiguity with Token Length. Therefore, the Token Length value MUST be used to define the size of the Compression Residue. A specific function designated as "TKL" MUST be used in the Rule. During the decompression, this function returns the value contained in the Token Length field.

5. CoAP options

CoAP defines options that are placed after the based header in Option Numbers order, see [rfc7252]. Each Option instance in a message uses the format Delta-Type (D-T), Length (L), Value (V). When applying SCHC compression to the Option, the D-T, L, and V format serve to make the Rule description of the Option. The SCHC compression builds the description of the Option by using in the Field ID the Option Number built from D-T; in TV, the Option Value; and the Option Length uses section 7.4 of [rfc8724]. When the Option Length has a wellknown size, it can be stored in the Rule. 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 direction indicator.

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 a new option is introduced in CoAP, a new Field ID has to be assigned in the Rules to allow its compression. Otherwise, if no Rule describes this Option, the SCHC compression is not possible, and the CoAP header is sent 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 matching-list SHOULD be used to limit the Compression Residue's size. Otherwise, the value has to be sent as a Compression Residue (fixed or variable length).

5.2. CoAP option Max-Age, Uri-Host, and Uri-Port fields

If both ends know the value, the value can be elided.

A matching list can be used if some well-known values are defined.

Otherwise, these options can be sent as a Compression Residue.

5.3. CoAP option Uri-Path and Uri-Query fields

Uri-Path and Uri-Query elements are repeatable options. The Field Position (FP) gives the position in the path.

A Mapping list can be used to reduce the size of variable Paths or Queries. In that case, to optimize the compression, several elements can be regrouped into a single entry. The Numbering of elements do not change; MO comparison is set with the first element of the matching.

Field	FL	FP	DI	Target Value	Match Opera.	CDA
Uri-Path		1	up	["/a/b", "/c/d"]	equal	not-sent
Uri-Path	var	3	up		ignore	value-sent

Figure 4: complex path example

In Figure 4, a single bit residue can be used to code one of the 2 paths. If regrouping were not allowed, a 2 bits residue would be needed. The third path element is sent as a variable size residue.

5.3.1. Variable-length Uri-Path and Uri-Query

When the length is not known at the Rule creation, the Field Length MUST be set to variable, and the unit is set to bytes.

The MSB MO can be applied to a Uri-Path or Uri-Query element. Since MSB value is given in bit, the size MUST always be a multiple of 8 bits.

The length sent at the beginning of a variable-length residue indicates the size of the LSB in bytes.

For instance, for a CORECONF path /c/X6?k="eth0" the Rule can be set to:

```
+-------------+---+--+--+--------+---------+-------------+
| Field       |FL |FP|DI| Target | Match   |     CDA     |
|             |   |  |  | Value  | Opera.  |             |
+-------------+---+--+--+--------+---------+-------------+
|Uri-Path     |  8| 1|up|"c"     |equal    |not-sent     |
|Uri-Path     |var| 2|up|        |ignore   |value-sent   |
|Uri-Query    |var| 1|up|"k="    |MSB(16)  |LSB          |
+-------------+---+--+--+--------+---------+-------------+
```

Figure 5: CORECONF URI compression

Figure 5 shows the parsing and the compression of the URI, where c is not sent. The second element is sent with the length (i.e., 0x2 X 6) followed by the query option (i.e. 0x05 "eth0").

5.3.2. Variable number of Path or Query elements

The number of Uri-Path or Uri-Query elements in a Rule is fixed at the Rule creation time. If the number varies, several Rules SHOULD be created to cover all the possibilities. Another possibility is to define the length of Uri-Path to variable and send a Compression Residue with a length of 0 to indicate that this Uri-Path is empty. This adds 4 bits to the variable Residue size. See section 7.5.2 [rfc8724]

5.4. CoAP option Size1, Size2, Proxy-URI and Proxy-Scheme fields

If the field value has to be sent, TV is not set, MO is set to "ignore", and CDA is set to "value-sent." A mapping MAY also be used.

Otherwise, the TV is set to the value, MO is set to "equal", and CDA is set to "not-sent".

5.5. CoAP option ETag, If-Match, If-None-Match, Location-Path, and Location-Query fields

These fields' values cannot be stored in a Rule entry. They MUST always be sent with the Compression Residues.

6. SCHC compression of CoAP extension RFCs

6.1. Block

Block [rfc7959] allows a fragmentation at the CoAP level. SCHC also includes a fragmentation protocol. They can be both used. If a block option is used, its content MUST be sent as a Compression Residue.

6.2. Observe

The [rfc7641] defines the Observe option. The TV is not set, MO is set to "ignore", and the CDA is set 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 an RST message may be sent to inform a server that the client does not require Observe response; a Rule SHOULD exist to allow the message's compression with the RST type.

6.3. No-Response

The [rfc7967] defines a No-Response option limiting the responses made by a server to a request. If both ends know the value, then TV is set to this value, MO is set to "equal", and CDA is set to "not-sent".

Otherwise, if the value is changing over time, TV is not set, MO is set to "ignore", and CDA to "value-sent". A matching list can also be used to reduce the size.

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.

```
      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 6: OSCORE Option

The encoding of the OSCORE Option Value defined in Section 6.1 of [rfc8613] is repeated in Figure 6.

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.

The flag byte is followed by the piv field, kid context field, and kid field in this order, and if present, the length of the kid context field is encoded in the first byte denoting by s the length of the kid context in bytes.

This specification recommends identifying the OSCORE Option and the fields it contains. Conceptually, it discerns up to 4 distinct pieces of information within the OSCORE option: the flag bits, the piv, the kid context, and the kid. The SCHC Rule splits into four field descriptions the OSCORE option to compress them:

o CoAP OSCORE_flags,

- o CoAP OSCORE_piv,
- o CoAP OSCORE_kidctx,
- o CoAP OSCORE_kid.

Figure 6 shows the OSCORE Option format with those four fields superimposed on it. Note that the CoAP OSCORE_kidctx field includes directly the size octet s.

7. Examples of CoAP header compression

7.1. Mandatory header with CON message

In this first scenario, the LPWAN Compressor at the Network Gateway side receives from an Internet client a POST message, which is immediately acknowledged by the Device. For this simple scenario, the Rules are described in Figure 7.

RuleID 1

Field	FL	FP	DI	Target Value	Match Opera.	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-map	matching-sent	T
CoAP TKL	4	1	bi	0	equal	not-sent	
CoAP Code	8	1	bi	[0.00,			
				...			
				5.05]	match-map	matching-sent	CC CCC
CoAP MID	16	1	bi	0000	MSB(7)	LSB	M-ID
CoAP Uri-Path	var	1	dw	path	equal 1	not-sent	

Figure 7: CoAP Context to compress header without token

The version and Token Length fields are elided. The 26 method and response codes defined in [rfc7252] has been shrunk to 5 bits using a matching list. Uri-Path contains a single element indicated in the matching operator.

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

Note that a request sent by a client located in an Application Server to a server located in the device, may not be compressed through this Rule since the MID will not start with 7 bits equal to 0. A CoAP proxy, before the core SCHC C/D can rewrite the message ID to a value matched by the Rule.

7.2. OSCORE Compression

OSCORE aims to solve the problem of end-to-end encryption for CoAP messages. The goal, therefore, is to hide as much of the message 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. This, in turn, is the part of the message which 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. This decomposition is illustrated in Figure 8.

CoAP options are sorted into one of 3 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 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.

Additionally, the OSCORE Option is added as an Outer option, signaling that the message is OSCORE protected. This option carries the information necessary to retrieve the Security Context with which the message was encrypted to be correctly decrypted at the other end-point.

```
                         Original CoAP Message
                      +-+-+---+-------+---------------+
                      |v|t|tkl| code  |  Msg Id.      |
                      +-+-+---+-------+---------------+....+
                      | Token                              |
                      +-------------------------------.....+
                      | Options (IEU)            |
                      .                          .
                      .                          .
                      +------+-------------------+
                      | 0xFF |
                      +------+------------------------+
                      |                               |
                      |     Payload                   |
                      |                               |
                      +-------------------------------+
                             /                \
                            /                  \
                           /                    \
                          /                      \
        Outer Header     v                        v  Plaintext
     +-+-+---+--------+---------------+          +-------+
     |v|t|tkl|new code|  Msg Id.      |          | code  |
     +-+-+---+--------+---------------+....+     +-------+-----......+
     | Token                               |     | Options (E)       |
     +--------------------------------.....+     +-------+------.....+
     | Options (IU)            |                 | 0xFF  |
     .                          .                +-------+-----------+
     . OSCORE Option            .                |                   |
     +------+-------------------+                | Payload           |
     | 0xFF |                                    |                   |
     +------+                                    +-------------------+
```

Figure 8: A CoAP message is split into an OSCORE outer and plaintext

Figure 8 shows the message format for the OSCORE Message and Plaintext.

In the Outer Header, the original message 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 Observe is not used. If Observe is used, the message code is replaced by FETCH for requests and Content for responses.

The original message code is put into the first byte of the Plaintext. Following the message code, the class E options come,

and, if present, the original message Payload is preceded by its payload marker.

The Plaintext is now encrypted by an AEAD algorithm which integrity protects Security Context parameters and, eventually, any class I options from the Outer Header. Currently, no CoAP options are marked class I. The resulting Ciphertext becomes the new Payload of the OSCORE message, as illustrated in Figure 9.

As defined in [rfc5116], this Ciphertext is the concatenation of the encrypted Plaintext and its authentication tag. Note that Inner Compression only affects the Plaintext before encryption. Thus only the first variable-length of the Ciphertext can be reduced. The authentication tag is fixed in length and is considered part of the cost of protection.

```
     Outer Header
  +-+-+---+--------+---------------+
  |v|t|tkl|new code|  Msg Id.      |
  +-+-+---+--------+---------------+....+
  | Token                               |
  +--------------------------------.....+
  | Options (IU)              |
  .                           .
  . OSCORE Option             .
  +------+-------------------+
  | 0xFF |
  +------+------------------------+
  |                               |
  | Ciphertext: Encrypted Inner   |
  |             Header and Payload |
  |             + Authentication Tag |
  |                               |
  +-------------------------------+
```

Figure 9: OSCORE message

The SCHC Compression scheme consists of compressing both the Plaintext before encryption and the resulting OSCORE message after encryption, see Figure 10.

This translates into a segmented process where SCHC compression is applied independently in 2 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.

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

```
        Outer Message                             OSCORE Plaintext
     +-+-+---+--------+---------------+          +-------+
     |v|t|tkl|new code|  Msg Id.      |          | code  |
     +-+-+---+--------+---------------+....+     +-------+-----......+
     | Token                               |     | Options (E)       |
     +--------------------------------.....+     +-------+------.....+
     | Options (IU)               |              | OxFF  |
     .                            .              +-------+-----------+
     . OSCORE Option              .              |                   |
     +------+-------------------+                | Payload           |
     | 0xFF |                                    |                   |
     +------+------------+                       +-------------------+
     |  Ciphertext       |<---------\                      |
     |                   |          |                      v
     +-------------------+          |             +-----------------+
             |                      |             |   Inner SCHC    |
             v                      |             |   Compression   |
       +-----------------+          |             +-----------------+
       |   Outer SCHC    |          |                      |
       |   Compression   |          |                      v
       +-----------------+          |              +-------+
             |                      |              |RuleID |
             v                      |              +-------+--+
         +--------+           +------------+       | Residue  |
         |RuleID' |           | Encryption | <---  +----------+--------+
         +--------+--+        +------------+       |                   |
         | Residue'  |                             | Payload           |
         +-----------+-------+                     |                   |
         |  Ciphertext       |                     +-------------------+
         |                   |
         +-------------------+
```

Figure 10: OSCORE Compression Diagram

7.3. Example OSCORE Compression

An example is given with a GET Request and its consequent Content Response from a device-based CoAP client to a cloud-based CoAP server. A possible set of Rules for the Inner and Outer SCHC Compression is shown. A dump of the results and a contrast between SCHC + OSCORE performance with SCHC + COAP performance is also listed. This gives an approximation to the cost of security with SCHC-OSCORE.

Our first example CoAP message is the GET Request in Figure 11

```
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 msg length:   17 bytes.
```

Figure 11: CoAP GET Request

Its corresponding response is the CONTENT Response in Figure 12.

```
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 msg length:   10
```

Figure 12: 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. As an example, see Figure 13.

```
 RuleID 0
+--------------+--+--+--+-----------+----------+----------++------+
| Field        |FL|FP|DI|  Target   |    MO    |    CDA   || Sent |
|              |  |  |  |  Value    |          |          ||[bits]|
+--------------+--+--+--+-----------+----------+----------++------+
|CoAP Code     | 8| 1|up|   1       |  equal   |not-sent  ||      |
|CoAP Code     | 8| 1|dw|[69,132]   | match-map|match-sent|| c    |
|CoAP Uri-Path |88| 1|up|temperature|  equal   |not-sent  ||      |
+--------------+--+--+--+-----------+----------+----------++------+
```

Figure 13: Inner SCHC Rules

Figure 14 shows the Plaintext obtained for the example GET Request and follows the process of Inner Compression and Encryption until the end up with the Payload to be added in the outer OSCORE Message.

In this case, the original message has no payload, and its resulting Plaintext can be compressed up to only 1 byte (size of the RuleID). The AEAD algorithm preserves this length in its first output and yields a fixed-size tag that cannot be compressed and has to be included in the OSCORE message. This translates into an overhead in total message length, limiting the amount of compression that can be achieved and plays into the cost of adding security to the exchange.

```
       ________________________________________________________
      |                                                        |
      | 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 residue)                    |
                   |_________________________________|

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

Figure 14: Plaintext compression and encryption for GET Request

In Figure 15, the process is repeated for the example CONTENT Response. The 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 though it has not been compressed.

On top of this, the overhead from the tag bytes is incurred as before.

```
   ________________________________________________________
  |                                                        |
  | 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-map 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 15: Plaintext compression and encryption for CONTENT Response

The Outer SCHC Rules (Figure 18) must process the OSCORE Options fields. The Figure 16 and Figure 17 show a dump of the OSCORE Messages generated from the example messages once they have been provided with the Inner Compressed Ciphertext in the payload. These are the messages that have to be compressed by the Outer SCHC Compression.

```
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 16: 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 17: Protected and Inner SCHC Compressed CONTENT Response

For the flag bits, some SCHC compression methods are useful, depending on the application. The simplest alternative is to provide a fixed value for the flags, combining MO equal and CDA not- sent. This saves most bits but could prevent flexibility. Otherwise, match-mapping could be used to choose from an interesting number of configurations for the exchange.
Otherwise, MSB could be used to mask off the 3 hard-coded most significant bits.

Note that fixing a flag bit will limit CoAP Options choice that can be used in the exchange since their values are dependent on certain options.

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

The size s included in the kid context field MAY be masked off with CDA MSB. The rest of the field could have additional bits masked off

or have the whole field be fixed with MO equal and CDA not-sent. The same holds for the kid field.

Figure 18 shows a possible set of Outer Rules to compress the Outer Header.

RuleID 0

Field	FL	FP	DI	Target Value	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	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	

Figure 18: Outer SCHC Rules

These Outer Rules are applied to the example GET Request and CONTENT Response. The resulting messages are shown in Figure 19 and Figure 20.

```
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 19: 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 msg length: 16 bytes
```

Figure 20: 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 Rules in Figure 21.

```
RuleID 1
+---------------+--+--+--+-----------+---------+-----------++-------+
| Field         |FL|FP|DI|  Target   |   MO    |     CDA   ||  Sent |
|               |  |  |  |  Value    |         |           || [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-map|map-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  |88| 1|up|temperature|equal    |not-sent   ||       |
+---------------+--+--+--+-----------+---------+-----------++-------+
```

Figure 21: SCHC-CoAP Rules (No OSCORE)

This yields the results in Figure 22 for the Request, and Figure 23 for the Response.

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

Compression Residue:
0b00010100 (1 byte)

Compressed msg length: 2
```

Figure 22: CoAP GET Compressed without OSCORE

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

Compression Residue:
0b00001010 (1 byte)

Payload
0x32332043

Compressed msg length: 6
```

Figure 23: CoAP CONTENT Compressed without OSCORE

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 request to IANA.

9. Security considerations

When applied to LPWAN, the Security Considerations of SCHC header compression [rfc8724] are valid for SCHC CoAP header compression. When CoAP uses OSCORE, the security considerations defined in [rfc8613] does not change when SCHC header compression is applied.

The definition of SCHC over CoAP header fields permits the compression of header information only. The SCHC header compression itself does not increase or reduce the level of security in the communication. When the connection does not use any security protocol as OSCORE, DTLS, or other, it is highly necessary to use a layer two security.

DoS attacks are possible if an intruder can introduce a compressed SCHC corrupted packet onto the link and cause a compression efficiency reduction. However, an intruder having the ability to add corrupted packets at the link layer raises additional security issues than those related to the use of header compression.

SCHC compression returns variable-length Residues for some CoAP fields. In the compressed header, the length sent is not the original header field length but the length of the Residue. So if a corrupted packet comes to the decompressor with a longer or shorter

length than the one in the original header, SCHC decompression will detect an error and drops the packet.

OSCORE compression is also based on the same compression method described in [rfc8724]. The size of the Initialisation Vector (IV) residue must be considered carefully. A residue size obtained with LSB CDA over the IV impacts on the compression efficiency and the frequency the device will renew its key. This operation requires several exchanges and is energy-consuming.

SCHC header and compression Rules MUST remain tightly coupled. Otherwise, an encrypted residue may be decompressed differently by the receiver. To avoid this situation, if the Rule is modified in one location, the OSCORE keys MUST be re-established.

10. Acknowledgements

The authors would like to thank (in alphabetic order): Christian Amsuss, Dominique Barthel, Carsten Bormann, Theresa Enghardt, Thomas Fossati, Klaus Hartke, Francesca Palombini, Alexander Pelov and Goran Selander.

11. 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. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <https://www.rfc-editor.org/info/rfc8724>.

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
2 rue de la Chataigneraie
CS 17607
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