lpwan Working Group A. Minaburo
Internet-Draft Acklio
Intended status: Informational L. Toutain
Expires: August 9, 2019 Institut MINES TELECOM; IMT Atlantique
R. Andreasen
Universidad de Buenos Aires
February 05, 2019

LPWAN Static Context Header Compression (SCHC) for CoAP
draft-ietf-lpwan-coap-static-context-hc-06

Abstract

This draft defines the way SCHC header compression can be applied to CoAP headers. The CoAP header structure differs from IPv6 and UDP protocols since CoAP
use a flexible header with a variable number of options themselves of a variable length. The CoAP protocol is asymmetric in its format messages, the format of the header packet in the request messages is different than in the response messages. Most of the compression mechanisms have been introduced in [I-D.ietf-lpwan-ipv6-static-context-hc], this document explains how to use the SCHC compression for CoAP.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

CoAP [rfc7252] is an implementation of the REST architecture for constrained devices. Nevertheless, if limited, the size of a CoAP header may be too large for LPWAN constraints and some compression may be needed to reduce the header size.

[I-D.ietf-lpwan-ipv6-static-context-hc] defines a header compression mechanism for LPWAN network based on a static context. The context is said static since the field description composing the Rules and the context are not learned during the packet exchanges but are previously defined. The context(s) is(are) known by both ends before transmission.

A context is composed of a set of rules that are referenced by Rule IDs (identifiers). A rule contains an ordered list of the fields descriptions containing a field ID (FID), its length (FL) and its position (FP), a direction indicator (DI) (upstream, downstream and bidirectional) and some associated Target Values (TV). Target Value indicates the value that can be expected. TV can also be a list of values. A Matching Operator (MO) is associated to each header field description. The rule is selected if all the MOs fit the TVs for all fields. In that case, a Compression/Decompression Action (CDA) associated to each field defines the link between the compressed and decompressed value for each of the header fields. Compression results mainly in 4 actions: send the field value, send nothing, send less significant bits of a field, send an index. Values sent are called Compression Residues and follows the rule ID.

2. SCHC Compression Process

The SCHC Compression rules can be applied to CoAP flows. SCHC Compression of the CoAP header may be done in conjunction with the above layers (IPv6/UDP) or independently. The SCHC adaptation layers as described in [I-D.ietf-lpwan-ipv6-static-context-hc] may be used as as shown in the Figure 1.

```
    ^   +------------+    ^  +------------+        ^  +------------+
    |   |    CoAP    |    |  |    CoAP    |  inner |  |    CoAP    |
    |   +------------+    v  +------------+        x  |    OSCORE  |
    |   |    UDP     |       |    DTLS    |  outer |  +------------+
    |   +------------+       +------------+        |  |    UDP     |
    |   |    IPv6    |       |    UDP     |        |  +------------+
    v   +------------+       +------------+        |  |    IPv6    |
                             |    IPv6    |        v  +------------+
                             +------------+
```

Figure 1: rule scope for CoAP

Figure 1 shows some examples for CoAP architecture and the SCHC rule's scope. A rule can covers all headers from IPv6 to CoAP, SCHC C/D is done in the device and at the LPWAN boundary. If an end-to-end encryption mechanisms is used between the device and the application. CoAP must be compressed independently of the other layers. The rule ID and the compression residue are encrypted using a mechanism such as DTLS. Only the other end can decipher the information.
Layers below may also be compressed using other SCHC rules (this is out of the scope of this document). OSCORE [I-D.ietf-core-object-security] can also define 2 rules to compress the CoAP message. A first rule focuses on the inner header and is end to end, a second rule may compress the outer header and the layer above. SCHC C/D for inner header is done by both ends, SCHC C/D for outer header and other headers is done between the device and the LPWAN boundary.

3. CoAP Compression with SCHC

CoAP differs from IPv6 and UDP protocols on the following aspects:

- IPv6 and UDP are symmetrical protocols. The same fields are found in the request and in the response, only the location in the header may vary (e.g. source and destination fields). A CoAP request is different from a response. For example, the URI-path option is mandatory in the request and is not found in the response, a request may contain an Accept option and the response a Content option.

  [I-D.ietf-lpwan-ipv6-static-context-hc] defines the use of a message direction (DI) when processing the rule which allows the description of message header format in both directions.

- Even when a field is "symmetric" (i.e. found in both directions) the values carried in each direction are different. Combined with a matching list in the TV, this will allow to reduce the range of expected values in a particular direction and therefore reduce the size of a compression residue. For instance, if a client sends only CON request, the type can be elided by compression and the answer may use one bit to carry either the ACK or RST type. Same behavior can be applied to the CoAP Code field (0.0X code are present in the request and Y.ZZ in the answer). The direction allows to split in two parts the possible values for each direction.

- In IPv6 and UDP header fields have a fixed size. In CoAP, Token size may vary from 0 to 8 bytes, length is given by a field in the header. More systematically, the CoAP options are described using the Type-Length-Value.

  [I-D.ietf-lpwan-ipv6-static-context-hc] offers the possibility to define a function for the Field Length in the Field Description.

- In CoAP headers, a field can be duplicated several times, for instances, elements of an URI (path or queries). The position defined in a rule, associated to a Field ID, can be used to identify the proper element.

  [I-D.ietf-lpwan-ipv6-static-context-hc] allows a Field id to appears several times in the rule, the Field Position (FP) removes ambiguities for the matching operation.

- Field size defined in the CoAP protocol can be too large regarding LPWAN traffic constraints. This is particularly true for the message ID field or Token field. The use of MSB MO can be used to reduce the information carried on LPWANs.

- CoAP also obeys to the client/server paradigm and the compression rate can be different if the request is issued from an LPWAN node or from an non LPWAN device. For instance a Device (Dev) aware of LPWAN constraints can generate a 1 byte token, but a regular CoAP client will certainly send a larger token to the Thing. SCHC compression will not modify the values to offer a better compression rate. Nevertheless a proxy placed before the compressor may change some field values to offer a better compression rate and maintain the necessary context for interoperability with existing CoAP implementations.

4. Compression of CoAP header fields

This section discusses of the compression of the different CoAP header fields.

4.1. CoAP version field

This field is bidirectional and must be elided during the SCHC compression, since it always contains the same value. In the future, if new version of CoAP are defined, new rules ID will be defined avoiding ambiguities between versions.

4.2. CoAP type field

[rfc7252] defines 4 types of messages: CON, NON, ACK and RST. The latter two ones are a response of the two first ones. If the device plays a specific role, a rule can exploit these property with the mapping list: [CON, NON] for one direction and [ACK, RST] for the other direction. Compression residue is reduced to 1 bit.

The field must be elided if for instance a client is sending only NON or CON messages.

In any case, a rule must be defined to carry RST to a client.

4.3. CoAP code field

The compression of the CoAP code field follows the same principle as for the CoAP type field. If the device plays a specific role, the set of code values can be split in two parts, the request codes with the 0 class and the response values.

If the device implement only a CoAP client, the request code can be reduced to the set of request the client is able to process.

All the response codes should be compressed with a SCHC rule.

4.4. CoAP Message ID field

This field is bidirectional and is used to manage acknowledgments. Server memorizes the value for a EXCHANGE_LIFETIME period (by default 247 seconds) for CON messages and a NON_LIFETIME period (by default 145 seconds) for NON messages. During that period, a server receiving the same Message ID value will process the message as a retransmission. After this period, it will be processed as a new messages.

In case the Device is a client, the size of the message ID field may the too large regarding the number of messages sent. Client may use only small message ID values, for instance 4 bit long. Therefore a MSB can be used to limit the size of the compression residue.

In case the Device is a server, client may be located outside of the LPWAN area and view the device as a regular device connected to the internet. The client will generate Message ID using the 16 bits space offered by this field. A CoAP proxy can be set before the SCHC C/D to reduce the value of the Message ID, to allow its compression with the MSB matching operator and LSB CDA.

4.5. CoAP Token fields

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 remains the same during all the transaction, the size can be stored in the context and elided during the transmission. Otherwise it will have to the send as a compression residue.

Token Value size should not be defined directly in the rule in the Field Length (FL). Instead a specific function designed as "TKL" must be used and length do not have to the sent with the residue. During the decompression, this function returns the value contained in the Token Length field.

5. CoAP options

5.1. CoAP Content and Accept options.

These field are both unidirectional and must not be set to bidirectional in a rule entry.

If single value is expected by the client, it can be stored in the TV and elided during the transmission. Otherwise, if several possible values are expected by the client, a matching-list should be used to limit the size of the residue. If is not possible, the value has to be sent as a residue (fixed or variable length).

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

This field is unidirectional and must not be set to bidirectional in a rule entry. It is used only by the server to inform of the caching duration and is never found in client requests.

If the duration is known by both ends, value can be elided on the LPWAN.

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

Otherwise these options should be sent as a residue (fixed or variable length).

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

This fields are unidirectional and must not be set to bidirectional in a rule entry. They are used only by the client to access to a specific resource and are never found in server responses.

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

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

FID	FL	FP	DI	TV	MO	CDA
URI-Path		1	up	["/a/b", "/c/d"]	equal	not-sent
URI-Path		3	up		ignore	value-sent

Figure 2: complex path example

In Figure 2 a single bit residue can be used to code one of the 2 paths. If regrouping was not allowed, a 2 bits residue is needed.

5.3.1. Variable length Uri-Path and Uri-Query

When the length is 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 apply 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 and the LSB CDA must not carry any value.

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

For instance for a CoMi path /c/X6?k="eth0" the rule can be set to:

FID	FL	FP	DI	TV	MO	CDA
URI-Path		1	up	"c"	equal	not-sent
URI-Path		2	up		ignore	value-sent
URI-Query		1	up	"k="	MSB (16)	LSB

Figure 3: CoMi URI compression

Figure 3 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 element 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 possibilities 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 add 4 bits to the compression residue.

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

These fields are unidirectional and must not be set to bidirectional in a rule entry. They are used only by the client to access to a specific resource and are never found in server response.

If the field value must be sent, TV is not set, MO is set to "ignore" and CDA is set to "value-sent. A mapping can 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 are unidirectional.

These fields values cannot be stored in a rule entry. They must always be sent with the compression residues.

6. Other RFCs

6.1. Block

Block [rfc7959] allows a fragmentation at the CoAP level. SCHC includes also a fragmentation protocol. They are compatible. If a block option is used, its content must be sent as a compression residue.

6.2. Observe

[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 should limit the delta between two consecutive value or a proxy can modify the incrementation.

Since RST message may be sent to inform a server that the client does not require Observe response, a rule must allow the transmission of this message.

6.3. No-Response

[rfc7967] defines an No-Response option limiting the responses made by a server to a request. If the value is not known by both ends, 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. Time Scale

Time scale [I-D.toutain-core-time-scale] option allows a client to inform the server that it is in a slow network and that message ID should be kept for a duration given by the option.

If the value is not known by both ends, 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.

6.5. OSCORE

OSCORE [I-D.ietf-core-object-security] 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_kidctxt ----->|<-- CoAP OSCORE_kid -->|
```

Figure 4: OSCORE Option

The encoding of the OSCORE Option Value defined in Section 6.1 of [I-D.ietf-core-object-security] is repeated in Figure 4.

The first byte is used for flags that specify the contents of the OSCORE option. The 3 most significant bits 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 3 least significant bits n indicate the length of the piv field in bytes. When n = 0, no piv is present.

After the flag byte follow the piv field, kid context field and kid field in 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 draft recommends to implement a parser that is able to identify 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. It is thus recommended that the parser split the OSCORE option into the 4 subsequent fields:

- CoAP OSCORE_flags,

- CoAP OSCORE_piv,

- CoAP OSCORE_kidctxt,

- CoAP OSCORE_kid.

These fields are superposed on the OSCORE Option format in Figure 4, the CoAP OSCORE_kidctxt field including the size bits s. Their size may be reduced using the MSB matching operator.

7. Examples of CoAP header compression

7.1. Mandatory header with CON message

In this first scenario, the LPWAN compressor receives from outside client a POST message, which is immediately acknowledged by the Device. For this simple scenario, the rules are described Figure 5.

Rule ID 1

Field.	FL	FP	DI	Target Value	Match Opera.	CDA	Sent [bits]
CoAP version			bi	01	equal	not-sent	
CoAP version			bi	01	equal	not-sent	
CoAP Type			dw	CON	equal	not-sent	
CoAP Type			up	[ACK, RST]	match-map	matching-sent	T
CoAP TKL			bi	0	equal	not-sent	
CoAP Code			bi	ML1	match-map	matching-sent	CC CCC
CoAP MID			bi	0000	MSB(7)	LSB(9)	M-ID
CoAP Uri-Path			dw	path	equal 1	not-sent	

Figure 5: CoAP Context to compress header without token

The version and Token Length fields are elided. Code has shrunk to 5 bits using a matching list. Uri-Path contains a single element indicated in the matching operator.

Figure 6 shows the time diagram of the exchange. A client in the Application Server sends a CON request. It can go through a proxy which reduces the message ID to a smallest value, with at least the 9 most significant bits equal to 0. SCHC Compression reduces the header sending only the Type, a mapped code and the least 9 significant bits of Message ID.

```
                       Device     LPWAN      SCHC C/D
                          |                    |
                          |       rule id=1    |<--------------------
                          |<-------------------| +-+-+--+----+------+
     <------------------- | CCCCCMMMMMMMMM     | |1|0| 4|0.01|0x0034|
    +-+-+--+----+-------+ | 00001000110100     | |  0xb4   p   a   t|
    |1|0| 1|0.01|0x0034 | |                    | |  h   |
    |  0xb4   p   a   t | |                    | +------+
    |  h   |              |                    |
    +------+              |                    |
                          |                    |
                          |                    |
   ---------------------->|      rule id=1     |
   +-+-+--+----+--------+ |------------------->|
   |1|2| 0|2.05| 0x0034 | |  TCCCCCMMMMMMMMM   |---------------------->
   +-+-+--+----+--------+ |  001100000110100   | +-+-+--+----+------+
                          |                    | |1|2| 0|2.05|0x0034|
                          v                    v +-+-+--+----+------+
```

Figure 6: Compression with global addresses

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 sensible information which 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 format of a regular CoAP message, and includes all Options and information needed for proxy operation and caching. This decomposition is illustrated in Figure 7.

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 so that it may 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)             |                | OxFF  |
     .                          .                +-------+-----------+
     . OSCORE Option            .                |                   |
     +------+-------------------+                | Payload           |
     | 0xFF |                                    |                   |
     +------+                                    +-------------------+
```

Figure 7: OSCORE inner and outer header form a CoAP message

Figure 7 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

[I-D.ietf-core-object-security], 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 comes 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 8.

This Ciphertext is, as defined in RFC 5116, the concatenation of the encrypted Plaintext and its authentication tag. Note that Inner Compression only affects the Plaintext before encryption, thus we can only aim to reduce this first, variable length component of the Ciphertext. The authentication tag is fixed in length and considered part of the cost of protection.

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

Figure 8: OSCORE message

The SCHC Compression scheme consists of compressing both the Plaintext before encryption and the resulting OSCORE message after encryption, see Figure 9.

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 Inner part of the message can only be decrypted by the corresponding end-point, 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
       +-----------------+          |              +-------+
             |                      |              |Rule ID|
             v                      |              +-------+--+
         +--------+           +------------+       | Residue  |
         |Rule ID'|           | Encryption | <---  +----------+--------+
         +--------+--+        +------------+       |                   |
         | Residue'  |                             | Payload           |
         +-----------+-------+                     |                   |
         |  Ciphertext       |                     +-------------------+
         |                   |
         +-------------------+
```

Figure 9: OSCORE Compression Diagram

7.3. Example OSCORE Compression

An example is given with a GET Request and its consequent CONTENT Response. 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 10

```
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 10: CoAP GET Request

Its corresponding response is the CONTENT Response in Figure 11.

```
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 11: CoAP CONTENT Response

The SCHC Rules for the Inner Compression include all fields that are already present in a regular CoAP message, what is important is the order of appearance and inclusion of only those CoAP fields that go into the Plaintext, Figure 12.

Rule ID 0

Field	FP	DI	Target	MO	CDA	Sent
			Value			[bits]
CoAP Code		up	1	equal	not-sent	
CoAP Code		dw	[69,132]	match-map	match-sent	c
CoAP Uri-Path		up	temperature	equal	not-sent	
COAP Option-End		dw	0xFF	equal	not-sent	

Figure 12: Inner SCHC Rules

Figure 13 shows the Plaintext obtained for our example GET Request and follows the process of Inner Compression and Encryption until we 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 Rule ID). The AEAD algorithm preserves this length in its first output, but also yields a fixed-size tag which cannot be compressed and must be

included in the OSCORE message. This translates into an overhead in total message length, which limits 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                            |
                  |                                 |
                  | Rule ID = 0x00 (1 byte)         |
                  | (No residue)                    |
                  |_________________________________|

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

Figure 13: Plaintext compression and encryption for GET Request

In Figure 14 we repeat the process for the example CONTENT Response. In this case the misalignment produced by the compression residue (1 bit) makes it so that 7 bits of padding must be applied after the payload, resulting in a compressed Plaintext that is the same size as before compression. This misalignment also causes the hexcode 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 Rule ID                             |
           |                                          |
           |    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 14: Plaintext compression and encryption for CONTENT Response

The Outer SCHC Rules (Figure 17) must process the OSCORE Options fields. In Figure 15 and Figure 16 we show a dump of the OSCORE

Messages generated from our example messages once they have been provided with the Inner Compressed Ciphertext in the payload. These are the messages that are to go through Outer SCHC Compression.

```
   Protected message:
   ==================
   0x4102000182d7080904636c69656e74ffa2c54fe1b434297b62
   (25 bytes)

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

   0x0001 = mid
   0x82 = token

   Options:
   0xd7080904636c69656e74 (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 15: 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 16: Protected and Inner SCHC Compressed CONTENT Response

For the flag bits, a number of compression methods could prove to be 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 hinder flexibility. Otherwise, match-mapping could allow to choose from a number of configurations of interest to the exchange. If neither of these alternatives is desirable, MSB could be used to mask off the 3 hard-coded most significant bits.

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

The piv field lends itself to having a number of bits masked off with MO MSB and CDA LSB. This could prove useful in applications where the message frequency is low such as that found in LPWAN technologies. Note that compressing the sequence numbers effectively reduces the maximum amount of sequence numbers that can be used in an exchange. Once this amount is exceeded, the SCHC Context would 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 17 shows a possible set of Outer Rules to compress the Outer Header.

Rule ID 0

Field	FP	DI	Target Value	MO	CDA	Sent [bits]
CoAP version		bi	01	equal	not-sent	
CoAP Type		up	0	equal	not-sent	
CoAP Type		dw	2	equal	not-sent	
CoAP TKL		bi	1	equal	not-sent	
CoAP Code		up	2	equal	not-sent	
CoAP Code		dw	68	equal	not-sent	
CoAP MID		bi	0000	MSB(12)	LSB	MMMM
CoAP Token		bi	0x80	MSB(5)	LSB	TTT
CoAP OSCORE_flags		up	0x09	equal	not-sent	
CoAP OSCORE_piv		up	0x00	MSB(4)	LSB	PPPP
COAP OSCORE_kid		up	0x636c69656e70	MSB(52)	LSB	KKKK
COAP OSCORE_kidctxt		bi	b''	equal	not-sent	
CoAP OSCORE_flags		dw	b''	equal	not-sent	
CoAP OSCORE_piv		dw	b''	equal	not-sent	
CoAP OSCORE_kid		dw	b''	equal	not-sent	
COAP Option-End		dw	0xFF	equal	not-sent	

Figure 17: Outer SCHC Rules

These Outer Rules are applied to the example GET Request and CONTENT Response. The resulting messages are shown in Figure 18 and Figure 19.

```
Compressed message:
==================
0x001489458a9fc3686852f6c4 (12 bytes)
0x00 Rule ID
    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 18: SCHC-OSCORE Compressed GET Request

```
Compressed message:
==================
0x0014218daf84d983d35de7e48c3c1852 (16 bytes)
0x00 Rule ID
    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 19: SCHC-OSCORE Compressed CONTENT Response

For contrast, we compare these results with what would be obtained by SCHC compressing the original CoAP messages without protecting them with OSCORE. To do this, we compress the CoAP messages according to the SCHC rules in Figure 20.

Rule ID 1

Field	FP	DI	Target	MO	CDA	Sent
			Value			[bits]
CoAP version		bi	01	equal	not-sent	
CoAP Type		up	0	equal	not-sent	
CoAP Type		dw	2	equal	not-sent	
CoAP TKL		bi	1	equal	not-sent	
CoAP Code		up	2	equal	not-sent	
CoAP Code		dw	[69,132]	equal	not-sent	
CoAP MID		bi	0000	MSB(12)	LSB	MMMM
CoAP Token		bi	0x80	MSB(5)	LSB	TTT
CoAP Uri-Path		up	temperature	equal	not-sent	
COAP Option-End		dw	0xFF	equal	not-sent	

Figure 20: SCHC-CoAP Rules (No OSCORE)

This yields the results in Figure 21 for the Request, and Figure 22 for the Response.

```
Compressed message:
==================
0x0114
0x01 = Rule ID

Compression residue:
0b00010100 (1 byte)

Compressed msg length: 2
```

Figure 21: CoAP GET Compressed without OSCORE

```
Compressed message:
==================
0x010a32332043
0x01 = Rule ID

Compression residue:
0b00001010 (1 byte)

Payload
0x32332043

Compressed msg length: 6
```

Figure 22: 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 of cost.

8. IANA Considerations

This document has no request to IANA.

9. Security considerations

This document does not have any more Security consideration than the ones already raised on [I-D.ietf-lpwan-ipv6-static-context-hc]

10. Acknowledgements

Thanks to all the persons that have give us feedback

11. Normative References

[I-D.ietf-core-object-security]
Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", draft-ietf-core-object-security-15 (work in progress), August 2018.

[I-D.ietf-lpwan-ipv6-static-context-hc]
Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and J. Zuniga, "LPWAN Static Context Header Compression (SCHC) and fragmentation for IPv6 and UDP", draft-ietf-lpwan-ipv6-static-context-hc-18 (work in progress), December 2018.

[I-D.toutain-core-time-scale]
Minaburo, A. and L. Toutain, "CoAP Time Scale Option", draft-toutain-core-time-scale-00 (work in progress), October 2017.

[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>.

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