

lpwan Working Group
Internet-Draft
Intended status: Informational
Expires: 4 January 2024

D. Barthel
Orange SA
L. Toutain
IMT Atlantique
A. Kandasamy
Acklio
D. Dujovne
Universidad Diego Portales
JC. Zuniga
Cisco
3 July 2023

OAM for LPWAN using Static Context Header Compression (SCHC)
draft-barthel-schc-oam-schc-01

Abstract

This document describes ICMPv6 compression with SCHC and how basic OAM is performed on Low Power Wide Area Networks (LPWANs) by compressing ICMPv6/IPv6 headers and by protecting the LPWAN network and the Device from undesirable ICMPv6 traffic.

With IP protocols now generalizing to constrained networks, users expect to be able to Operate, Administer and Maintain them with the familiar tools and protocols they already use on less constrained networks.

OAM uses specific messages sent into the data plane to measure some parameters of a network. Most of the time, no explicit values are sent in these messages. Network parameters are obtained from the analysis of these specific messages.

This can be used:

- * To detect if a host is up or down.
- * To measure the RTT and its variation over time.
- * To learn the path used by packets to reach a destination.

OAM in LPWAN is a little bit trickier since the bandwidth is limited and extra traffic added by OAM can introduce perturbation on regular transmission.

Three main scenarios are investigated:

- * OAM reachability messages coming from internet. In that case, the SCHC core should act as a proxy and handle specifically the OAM traffic.
- * OAM messages initiated by LPWAN devices: They can be anticipated by the core SCHC.
- * OAM error messages coming from internet. In that case, the SCHC core may forward a compressed version to the device.

The primitive functionalities of OAM are achieved with the ICMPv6 protocol.

ICMPv6 defines messages that inform the source of IPv6 packets of errors during packet delivery. It also defines the Echo Request/Reply messages that are used for basic network troubleshooting (ping command). ICMPv6 messages are transported on IPv6.

This document also introduces the notion of actions in a SCHC rule, to perform locally some operations.

Status of This Memo

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

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

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

This Internet-Draft will expire on 4 January 2024.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Use cases	4
4. Detailed behavior	5
4.1. Device does a ping	5
4.1.1. Rule example	6
4.2. Device is ping'ed	7
4.2.1. Rule example	7
4.3. Device is the source of an ICMPv6 error message	8
4.4. Device is the destination of an ICMPv6 error message	9
4.4.1. ICMPv6 error message compression.	10
5. YANG identities and tree	13
6. YANG Module	14
7. Security considerations	18
8. IANA Considerations	18
9. References	18
9.1. Normative References	18
9.2. Informative References	19
Authors' Addresses	20

1. Introduction

The primitive functionalities of OAM [RFC6291] are achieved with the ICMPv6 protocol.

ICMPv6 [RFC4443] is a companion protocol to IPv6 [RFC8200].

[RFC4443] defines a generic message format. This format is used for messages to be sent back to the source of an IPv6 packet to inform it about errors during packet delivery.

More specifically, [RFC4443] defines 4 error messages: Destination Unreachable, Packet Too Big, Time Exceeded and Parameter Problem.

[RFC4443] also defines the Echo Request and Echo Reply messages, which provide support for the ping application.

Other ICMPv6 messages are defined in other RFCs, such as an extended format of the same messages [RFC4884] and other messages used by the Neighbor Discovery Protocol [RFC4861].

This document focuses on using Static Context Header Compression (SCHC) to compress [RFC4443] messages that need to be transmitted over the LPWAN network, and on having the LPWAN gateway proxying the Device to save it the unwanted traffic.

LPWANs salient characteristics are described in [RFC8376].

2. Terminology

This draft re-uses the Terminology defined in [RFC8724].

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.

3. Use cases

In the LPWAN architecture, we can distinguish the following cases:

- * the Device is the originator of an Echo Request message, and therefore the destination of the Echo Reply message. This message is compressed by the device through SCHC rules specifying ICMPv6 fields.
- * the Device is the destination of an Echo Request message, and therefore the purported source of an Echo Reply message. The core SCHC can either send a compressed SCHC message, or proxy the answer to avoid sending data on the constrained link. The proxy answer can be related to the device activity.
- * the Device is the (purported) source of an ICMP error message, mainly in response to an incorrect incoming IPv6 message, or in response to a ping request. In this case, as much as possible, the core SCHC C/D should act as a proxy and originate the ICMP Destination Unreachable message, so that the Device and the LPWAN network are protected from this unwanted traffic.
- * the Device is the destination of the ICMP message, mainly in response to a packet sent by the Device to the network that generates an error. In this case, we want the ICMP message to reach the Device, and this document describes in Section 4.4.1 what SCHC compression should be applied.

These cases are further described in Section 4.

4. Detailed behavior

4.1. Device does a ping

A Device may send some Echo Request message to check the availability of the network or the host running the Application.

If a ping request is generated by a Device, then SCHC compression applies.

The format of an ICMPv6 Echo Request message is described in Figure 1, with Type=128 and Code=0.

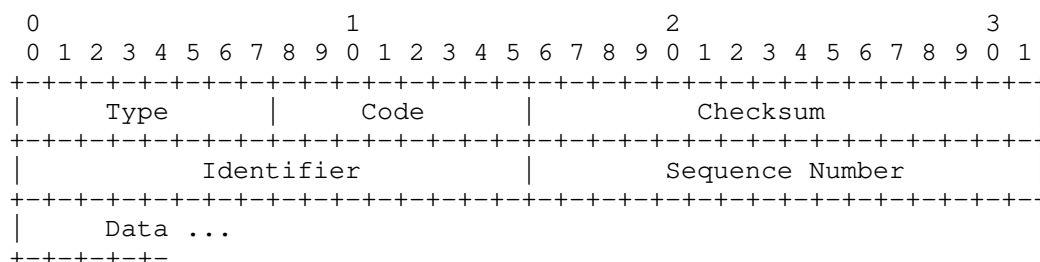


Figure 1: ICMPv6 Echo Request message format

If we assume that one rule will be devoted to compressing Echo Request messages, then Type and Code are known in the rule to be 128 and 0 and can therefore be elided with the not-sent CDA.

Checksum can be reconstructed with the compute-checksum CDA and therefore is not transmitted.

[RFC4443] states that Identifier and Sequence Number are meant to aid in matching Echo Replies to this Echo Request and that they may be zero. Data is zero or more bytes of arbitrary data.

For constrained devices or networks, we recommend that Identifier be zero, Sequence Number be a counter on 3 bits, and Data be zero bytes (absent). Therefore, Identifier is elided with the not-sent CDA, Sequence Number is transmitted on 3 bits with the LSB CDA and no Data is transmitted.

The transmission cost of the Echo Request message is therefore the size of the Rule Id + 3 bits. The rule ID length can be chosen to avoid adding padding.

When the destination receives the Echo Request message, it will respond back with a Echo Reply message. This message bears the same format as the Echo Request message but with Type = 129 (see Figure 1).

[RFC4443] states that the Identifier, Sequence Number and Data fields of the Echo Reply message shall contain the same values as the invoking Echo Request message. Therefore, a rule shall be used similar to that used for compressing the Echo Request message.

4.1.1. Rule example

The following rule gives an example of a SCHC compression. The type can be elided if the direction is taken into account. Identifier is ignored and generated as 0 at decompression. This implies that only one single ping can be launched at any given time on a device. Finally, only the least significant 8 bits of the sequence number are sent on the LPWAN, allowing a serie of 255 consecutive pings.

Field	FL	FP	DI	Target Value	Matching Operator	CDA	Sent bits
IPv6 Headers description							
ICMPv6 Type	8	1	Up	128	equal	not-sent	
ICMPv6 Type	8	1	Dw	129	equal	not-sent	
ICMPv6 Code	8	1	Bi	0	equal	not-sent	
ICMPv6 Checksum	16	1	Bi		ignore	compute-*	
ICMPv6 Identifier	16	1	Bi	0	ignore	not-sent	
ICMPv6 Sequence	16	1	Bi	0	MSB(8)	LSB	8

Table 1: Example of compression rule for a ping from the device

4.2. Device is ping'ed

If the Device is pinged (i.e., is the destination of an Echo Request message), the device receives the compress message and generate an Echo. In that case, the fields sequence number and identifier cannot be compressed if the source is not aware of the compression scheme.

But the default behavior is to avoid propagating the Echo Request message over the LPWAN.

This is done by proxying the ping request on the core SCHC C/D. This requires to introduce a new processing when the rule is selected. The selection of a compression rule triggers the compression and sends the SCHC packet to the other end. Specifying an Action, change this behavior. In our case, being processed by the compressor, the packet description is processed by a ping proxy. Since the rule is used for the selection, so CDAs are not necessary and set to "not-sent".

The ping-proxy takes a parameter in second, gives the interval during which the device is considered active. During this interval, the proxy-ping echoes ping requests, after this duration, the ping request will be discarded.

The resulting behavior is shown on Figure 2 and described below:

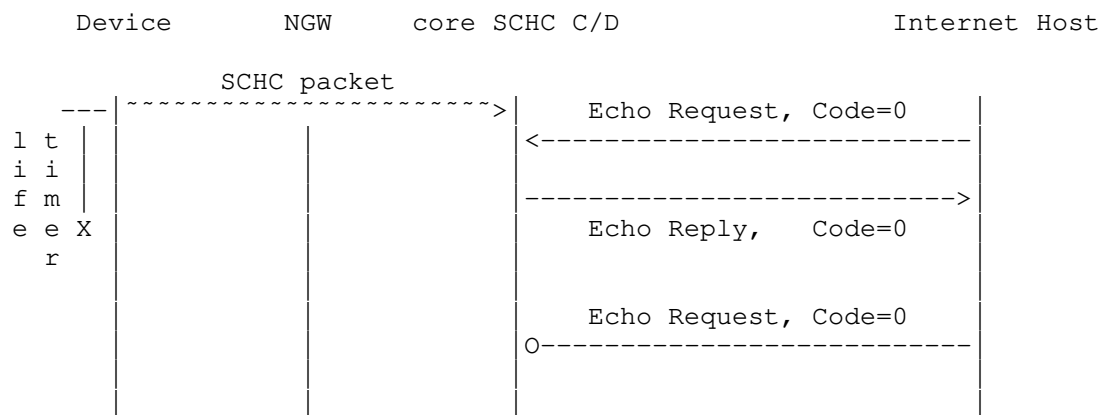


Figure 2: Examples of ICMPv6 Echo Request/Reply

4.2.1. Rule example

The following rule shows an example of a compression rule for pingging a device.

Field	FL	FP	DI	Target Value	Matching Operator	CDA	Sent bits
Action: proxy-ping(300)							
IPv6 Headers description							
ICMPv6 Type	8	1	Dw	128	equal	not-sent	
ICMPv6 Code	8	1	Dw	0	equal	not-sent	
ICMPv6 Checksum	16	1	Dw		ignore	compute-*	
ICMPv6 Identifier	16	1	Dw		ignore	not-sent	
ICMPv6 Sequence	16	1	Dw		ignore	not-sent	8

Table 2: Example of compression rule for a ping to a device

In this example, type and code are elided, the identifier has to be sent, and the sequence number is limited to one byte.

4.3. Device is the source of an ICMPv6 error message

As stated in [RFC4443], a node should generate an ICMPv6 message in response to an IPv6 packet that is malformed or which cannot be processed due to some incorrect field value.

The general intent of this document is to spare both the Device and the LPWAN network this un-necessary traffic. The incorrect packets should be caught at the core SCHC C/D and the ICMPv6 notification should be sent back from there.

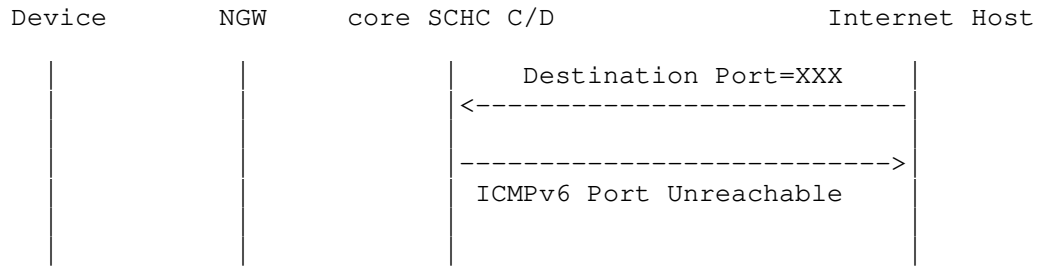


Figure 3: Example of ICMPv6 error message sent back to the Internet

Figure 3 shows an example of an IPv6 packet trying to reach a Device.

Let's assume that no rule matches the incoming packet (i.e. there is no co-compression rule)

Instead of sending the packet over the LPWAN and having this packet rejected by the Device, the core SCHC C/D issues an ICMPv6 error message Destination Unreachable (Type 1) with Code 1 (Port Unreachable) on behalf of the Device.

In that case the SCHC C/D MAY act as a router (i.e. it MUST have a routable IPv6 address to generate an ICMPv6 message). When compressing a packet containing an IPv6 header, no compression rules are found and: * if a rule contains some extension headers, a parameter problem may be generated (type 4), * no rule contains the IPv6 device address found in the incoming packet, a no route to destination ICMPv6 message (type 0, code 3) may be generated, * a device IPv6 address is found, but no port matches, a port unreachable ICMPv6 message (type 0, code 4) may be generated,

4.4. Device is the destination of an ICMPv6 error message

In this situation, we assume that a Device has been configured to send information to a server on the Internet. If this server becomes no longer accessible, an ICMPv6 message will be generated back towards the Device by either an intermediate router or the destination. This information can be useful to the Device, for example for reducing the reporting rate in case of periodic reporting of data. Therefore, we compress the ICMPv6 message using SCHC and forward it to the Device over the LPWAN. We also introduce new MO and CDA that can be used to test the presence and/or compress the returning payload.

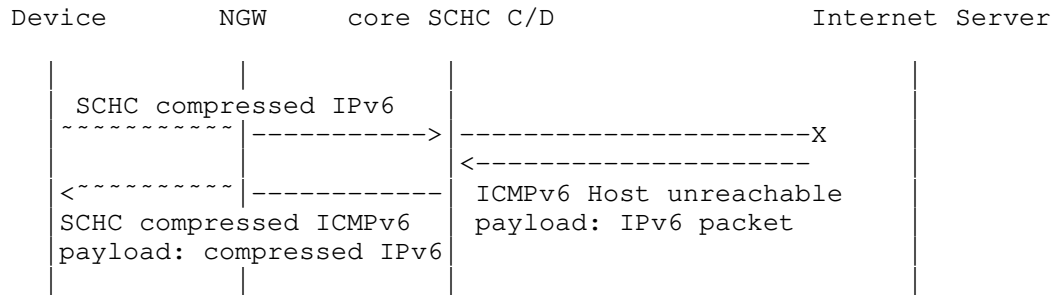


Figure 4: Example of ICMPv6 error message sent back to the Device

Figure 4 illustrates this behavior. The ICMPv6 error message is compressed as described in Section 4.4.1 and forwarded over the LPWAN to the Device.

The SCHC returning message contains the SCHC residue of the ICMPv6 message and MAY contain the compressed original message contained in the ICMP message. The compression can be done by the core SCHC by reversing the direction as if this message was issued by the device.

4.4.1. ICMPv6 error message compression.

The ICMPv6 error messages defined in [RFC4443] contain the fields shown in Figure 5.

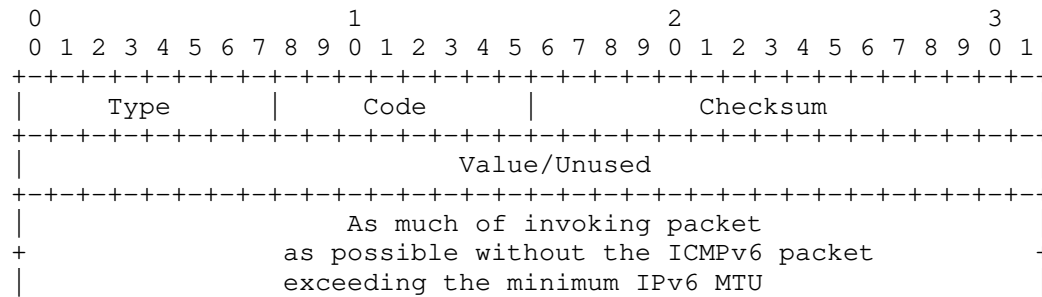


Figure 5: ICMPv6 Error Message format

[RFC4443] states that Type can take the values 1 to 4, and Code can be set to values between 0 and 6. Value is unused for the Destination Unreachable and Time Exceeded messages. It contains the MTU for the Packet Too Big message and a pointer to the byte causing the error for the Parameter Error message. Therefore, Value is never expected to be greater than 1280 in LPWAN networks.

The payload is viewed as a field. An unused field MUST not appear in the compressoin rules.

The source address of the message SHOULD be ignore, since it can be initiated by any router on the path.

The following generic rule can therefore be used to compress all ICMPv6 error messages as defined today. More specific rules can also be defined to achieve better compression of some error messages.

The Type field can be associated to a matching list [1, 2, 3, 4] and is therefore compressed down to 2 bits. Code can be reduced to 3 bits using the LSB CDA. Value can be sent on 11 bits using the LSB CDA, but if the Device is known to send smaller packets, then the size of this field can be further reduced.

The first rule example Table 3 just sends the ICMP type and code as residue to the device.

Field	FL	FP	DI	Target Value	Matching Operator	CDA	Sent bits
IPv6 Headers descript ion							
ICMPv6 Type	8	1	Dw	1	equal	not-sent	
ICMPv6 Code	8	1	Dw	[0,1,2,3,4,5,6]	match-mapping	mapping-sent	3
ICMPv6 Checksum	16	1	Dw		ignore	compute-*	
ICMPv6 Payload	var	1	Dw		ignore	not-sent	

Table 3: Example of compression rule for a ICMP error to a device

The second rule example Table 4 also only sends the ICMP type and code as residue to the device, but it introduces the new MO "rev-rule-match". This MO will check if a rule matches the payload.

Field	FL	FP	DI	Target Value	Matching Operator	CDA	Sent bits
IPv6 Headers description							
ICMPv6 Type	8	1	Dw	1	equal	not-sent	
ICMPv6 Code	8	1	Dw	[0, 1, 2, 3, 4, 5, 6]	match-mapping	mapping-sent	
ICMPv6 Checksum	16	1	Dw		ignore	compute-*	
ICMPv6 Payload	var	1	Dw		rev-rule-match	not-sent	

Table 4: Example of compression rule for a ICMP error to a device

By [RFC4443], the rest of the ICMPv6 message must contain as much as possible of the IPv6 offending (invoking) packet that triggered this ICMPv6 error message. This information is used to try and identify the SCHC rule that was used to decompress the offending IPv6 packet. If the rule can be found then the Rule Id is added at the end of the compressed ICMPv6 message. Otherwise the compressed packet ends with the compressed Value field.

The third rule example Table 5 also sends the ICMP type, code and the compressed payload as residue. It can be noted that this field is identified as "variable" in the rule which will introduce a size before the IPv6 compressed header.

Field	FL	FP	DI	Target Value	Matching Operator	CDA	Sent bits
IPv6 Headers description							
ICMPv6 Type	8	1	Dw	128	equal	not-sent	
ICMPv6 Code	8	1	Dw	[0, 1, 2, 3, 4, 5, 6]	match-mapping	mapping-sent	
ICMPv6 Checksum	16	1	Dw		ignore	compute-*	
ICMPv6 Payload	var	1	Dw		rev-rule-match	rev-compress-sent	(compressed IPv6 header*8) + 4 or +12 (for variable length)

Table 5: Example of compression rule for a ICMP error to a device

LT: do we add packet too big, for instance if a fragmentation rule cannot handle a size larger than 1280?

5. YANG identities and tree

Figure 6 shows the augmentation of the Data Model defined in [RFC9363]

This YANG module extends Field ID identities to includes fields contained in ICMPv6 header. Note that the ICMPv6 payload is parsed to the specific field "fid-icmpv6-payload"

It also defines two new Most identities:

- * mo-rev-rule-match: The value contained in the Field Value matches a rule. The direction used for matching is the opposite of the incoming message: UP becomes DOWN and DOWN becomes UP. This MO can be used to test if the Payload contained in the ICMPv6 message matches a rule. This means that the original packet, at the origine of the ICMPv6 message, may have been generated from the SCHC decompression.

- * `mo-rule-match`: The value contained in the Target Value matches a rule. The direction is the one of the incoming message. This MO is not used for ICMPv6 messages, but since it can be used in other situations, it has been included in the Data Model.

The Field Value may be compressed by a rule. The result SHOULD be included in the SCHC message as a variable length residue. It contains the Rule ID used by the compression, the residue, the payload and some padding bits since the variable length init is in bytes.

- * `cda-rev-compress-sent`: The direction used for compression is the opposite of the incoming message: UP becomes DOWN and DOWN becomes UP.
- * `cda-compress-sent`: The direction used for compression is the same as for the incoming message.

module: ietf-schc-oam

```
augment /schc:schc/schc:rule/schc:nature/schc:compression:
  +--rw proxy-behavior?          schc-oam:proxy-type
  +--rw proxy-behavior-value* [index]
     +--rw index                uint16
     +--rw value?              binary
```

Figure 6: YANG tree

6. YANG Module

```
module ietf-schc-oam {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-schc-oam";
  prefix schc-oam;

  import ietf-schc {
    prefix schc;
  }

  organization
    "IETF IPv6 over Low Power Wide-Area Networks (lpwan) working group";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/lpwan/about/>
    WG List: <mailto:p-wan@ietf.org>
    Editor: Laurent Toutain
           <mailto:laurent.toutain@imt-atlantique.fr>
    Editor: Ana Minaburo
           <mailto:ana@ackl.io>";
```

description

"

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

This module extends the ietf-schc module to include the compound-ack behavior for Ack On Error as defined in RFC YYYY. It introduces a new leaf for Ack on Error defining the format of the SCHC Ack and add the possibility to send several bitmaps in a single answer.";

```
revision 2023-06-26 {
```

```
  description
```

```
    "Initial version for RFC YYYY ";
```

```
  reference
```

```
    "RFC YYYY: OAM";
```

```
}
```

```
identity fid-icmpv6-base-type {
```

```
  base schc:fid-base-type;
```

```
  description
```

```
    "Field IP base type for ICMPv6 headers described in RFC 4443";
```

```
  reference
```

```
    "RFC 4443  Internet Control Message Protocol (ICMPv6)  
              for the Internet Protocol Version 6 (IPv6) Specification";
```

```
}
```

```
// ICMPv6 Fields
```

```
identity fid-icmpv6-type {
  base schc:fid-icmpv6-base-type;
  description
    "ICMPv6 type field";
}

identity fid-icmpv6-code {
  base schc:fid-icmpv6-base-type;
  description
    "ICMPv6 code field";
}

identity fid-icmpv6-checksum {
  base schc:fid-icmpv6-base-type;
  description
    "ICMPv6 checksum field";
}

identity fid-icmpv6-mtu {
  base schc:fid-icmpv6-base-type;
  description
    "ICMPv6 MTU (see draft OAM)";
}

identity fid-icmpv6-pointer {
  base schc:fid-icmpv6-base-type;
  description
    "ICMPv6 field (see draft OAM)";
}

identity fid-icmpv6-identifier {
  base schc:fid-icmpv6-base-type;
  description
    "ICMPv6 identifier field";
}

identity fid-icmpv6-sequence {
  base schc:fid-icmpv6-base-type;
  description
    "ICMPv6 sequence number field";
}

identity fid-icmpv6-payload {
  base schc:fid-icmpv6-base-type;
  description
    "payload in the ICMPv6 message";
}
```

```
// MO and CDA

identity mo-rule-match {
  base schc:mo-base-type;
  description
    "Macthing operator return true, if the TV matches a rule
    keeping UP and DOWN direction." ;
}

identity mo-rev-rule-match {
  base schc:mo-base-type;
  description
    "Macthing operator return true, if the TV matches a rule
    reversing UP and DOWN direction." ;
}

identity cda-compress-sent {
  base schc:mo-base-type;
  description
    "Send a compressed version of TV keeping UP and
    DOWN direction." ;
}

identity cda-rev-compress-sent {
  base schc:mo-base-type;
  description
    "Send a compressed version of TV reversing UP and
    DOWN direction." ;
}

// Proxy actions

identity proxy-schc-message{
  description
    "Define how the message is proxied after compression";
}

identity proxy-none {
  base proxy-schc-message;
  description
    "The message is not proxied and sent to L2,
    default behavior of RFC 8724";
}

identity proxy-pingv6 {
  base proxy-schc-message;
  description
```

```
    "The message is processed by an ping6 proxy";
}

typedef proxy-type {
  type identityref {
    base proxy-schc-message;
  }
  description
    "type used in rules";
}

// SCHC rule

augment "/schc:schc/schc:rule/schc:nature/schc:compression" {
  leaf proxy-behavior {
    type schc-oam:proxy-type;
    default "schc-oam:proxy-none";
    description
      "Entity proxying the SCHC message.";
  }
  list proxy-behavior-value {
    key "index";
    uses schc:tv-struct;
    description
      "Parameters associated to the proxy action.";
  }
  description
    "added to SCHC rules";
}

}
```

Figure 7: YANG module

7. Security considerations

flood the return path with ICMP error messages.

8. IANA Considerations

TODO

9. References

9.1. 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>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <<https://www.rfc-editor.org/info/rfc4884>>.
- [RFC6291] Andersson, L., van Helvoort, H., Bonica, R., Romascanu, D., and S. Mansfield, "Guidelines for the Use of the "OAM" Acronym in the IETF", BCP 161, RFC 6291, DOI 10.17487/RFC6291, June 2011, <<https://www.rfc-editor.org/info/rfc6291>>.
- [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>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [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>>.
- [RFC9363] Minaburo, A. and L. Toutain, "A YANG Data Model for Static Context Header Compression (SCHC)", RFC 9363, DOI 10.17487/RFC9363, March 2023, <<https://www.rfc-editor.org/info/rfc9363>>.

9.2. Informative References

[RFC8376] Farrell, S., Ed., "Low-Power Wide Area Network (LPWAN) Overview", RFC 8376, DOI 10.17487/RFC8376, May 2018, <<https://www.rfc-editor.org/info/rfc8376>>.

Authors' Addresses

Dominique Barthel
Orange SA
28 chemin du Vieux Chene
BP 98
38243 Meylan Cedex
France
Email: dominique.barthel@orange.com

Laurent Toutain
IMT Atlantique
2 rue de la Chataigneraie
CS 17607
35576 Cesson-Sevigne Cedex
France
Email: laurent.toutain@imt-atlantique.fr

Arunprabhu Kandasamy
Acklio
1137A avenue des Champs Blancs
35510 Cesson-Sevigne Cedex
France
Email: arun@ackl.io

Diego Dujovne
Universidad Diego Portales
Vergara 432
Santiago
Chile
Email: diego.dujovne@mail.udp.cl

Juan Carlos Zuniga
Cisco
Montreal QC
Canada
Email: juzuniga@cisco.com

SCHC Working Group
Internet-Draft
Intended status: Informational
Expires: 20 April 2026

A. Pelov
IMT Atlantique
P. Thubert

A. Minaburo
Consultant
17 October 2025

Static Context Header Compression (SCHC) Architecture
draft-ietf-schc-architecture-05

Abstract

This document defines the SCHC architecture.

Status of This Memo

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

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

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

This Internet-Draft will expire on 20 April 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1.	Introduction	2
2.	Requirements Language	3
3.	Terminology	3
4.	Building Blocks	4
4.1.	SCHC Stratum (plural: strata)	4
4.2.	Discriminator	5
4.3.	SCHC Control End-Point	6
4.3.1.	SCHC Control Header	7
4.4.	SCHC Data End-Point	8
4.4.1.	SCHC Data Header	9
4.5.	SCHC Profiles	10
4.6.	SCHC Operation	10
4.6.1.	SCHC Rules	11
4.6.2.	SoR identification	11
4.7.	SCHC Management	11
4.7.1.	SCHC Instance Manager	12
4.7.2.	SCHC Data Model	12
5.	SCHC Architecture	14
6.	The Static Context Header Compression	17
6.1.	SCHC over Network Technologies	18
6.1.1.	SCHC over PPP	19
6.1.2.	SCHC over Ethernet	20
6.1.3.	SCHC over IPv6	20
6.1.4.	SCHC over UDP	21
7.	SCHC Endpoints for LPWAN Networks	21
7.1.	SCHC Device Lifecycle	22
7.1.1.	Device Development	22
7.1.2.	Rules Publication	23
7.1.3.	SCHC Device Deployment	23
7.1.4.	SCHC Device Maintenance	23
7.1.5.	SCHC Device Decommissioning	24
8.	Security Considerations	24
9.	IANA Consideration	24
10.	Acknowledgements	24
11.	References	24
11.1.	Normative References	24
11.2.	Informative References	25
	Authors' Addresses	27

1. Introduction

The IETF LPWAN WG defined the necessary operations to enable IPv6 over selected Low-Power Wide Area Networking (LPWAN) radio technologies. [rfc8376] presents an overview of those technologies.

The Static Context Header Compression (SCHC) [rfc8724] technology is the core product of the IETF LPWAN working group and was the basis to form the SCHC Working Group. [rfc8724] defines a generic framework for header compression and fragmentation, based on a static context that is pre-installed on the SCHC endpoints.

This document details the constitutive elements of a SCHC-based solution, and how the solution can be deployed. It provides a general architecture for a SCHC deployment, positioning the required specifications, describing the possible deployment types, and indicating models whereby the rules can be distributed and installed to enable reliable and scalable operations.

2. Requirements Language

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.

3. Terminology

- * C/D. Compression and Decompression.
- * Context. All the information related to the Rules for a SCHC Header operation, which can be either Non-Compression, C/D, F/R, or CORECONF_Management.
- * FID. Field Identifiers, describing the name of the field in a protocol header.
- * F/R. Fragmentation and Reassembly.
- * Rule. A description of the header fields to performs compression/decompression, fragmentation/reassembly, SCHC end-points and CORECONF_Management.
- * SCHC Gateway (end-point). The SCHC end-point located upstream, e.g., in a Network Core Software.
- * SCHC Device (end-point). The SCHC end-point located downstream, e.g., in a constrained physical device.
- * SCHC Header. A SCHC Header contains the information necessary for a SCHC operation, e.g., a Rule ID and a residue. The SCHC Control Header transports SCHC's control information whereas the SCHC Data Header transports user payload that was processed by SCHC.

- * SCHC end-point. An entity (e.g., Device, Application and Network Gateway) involved in the SCHC process. Each SCHC end-point will have its Set of Rules (SoR), based on the profile, the protocols, the device, the behaviour and a Set of Variables (SoV).
- * SCHC Instance. The session between SCHC end-points in two or more peer nodes operating SCHC to communicate using a common SoR and a matching SoV. There are at least two SCHC Instances involved per SCHC stratum, one SCHC Control Instance that manages the SCHC Control Header to discriminate the SCHC Data Instance(s) and one or more SCHC Data Instance(s) that handle(s) the SCHC Data Header, for, e.g., compression and decompression operations.
- * SCHC Instance Manager. Provides the management of SCHC end-points, the SoR of each end-point and the dialog between hosts to keep the SCHC synchronization, and the establishment of SCHC Instances with peer nodes.
- * SoR (Set of rules). Group of Rules used in a SCHC end-point. The set of rules contains Rules for different nature as compression, no compression, fragmentation, SCHC end-points and CORECONF management.
- * SoV (Set of Variables). External information that needs to be known to identify the correct protocol, the SCHC Instance id, and the flow when there is one.
- * SCHC Stratum. A SCHC Stratum is the SCHC analogous to a classical layer in the IP architecture, but its operation may cover multiple IP layers or only a subset of a layer.

4. Building Blocks

This section specifies the principal blocks defined for building and using the SCHC architecture in any network topology and protocol.

4.1. SCHC Stratum (plural: strata)

A SCHC Stratum is the SCHC analogous to a classical layer in the IP architecture, but its operation may cover multiple IP layers or only a subset of a layer, e.g., IP only, IP+UDP, CoAP, or OSCORE [rfc8824]. The term stratum is thus used to avoid confusion with traditional layers. Also, SCHC Strata are not stacked, though they can be nested.

The SCHC Stratum data in a datagram is composed of a SCHC Control Header (which may be compressed to the point that it is fully implicit and thus elided), a SCHC Data Header (that is used to

uncompress a section of the SCHC datagram), and possibly some remaining payload that is unaffected by the SCHC Stratum. The SCHC Stratum operation requires at least 2 end-points, one for the SCHC Control Header and one or more for the SCHC Data Header.

A SCHC datagram may contain additional stratum data, to be handled by sequential (nested) SCHC Strata, where the inner (nested) Stratum operates within the decompressed/reassembled payload of the outer (nesting) Stratum.

A SCHC Stratum is instantiated in participating nodes as a pair of SCHC end-points, and matching SCHC end-points in communicating nodes are associated to form a SCHC end-point. A SCHC end-point may be Point to point (P2P), or Point to Multipoint. A P2MP SCHC end-point is unidirectional, meaning that all the SCHC datagrams are generated by the same node. A P2P SCHC end-point may be unidirectional or bidirectional, symmetrical (between peers) or asymmetrical (between a device and an application).

A SCHC end-point operates datagram fragmentation and/or data compression and decompression, and maintains the state and timers associated with the Stratum operation over the consecutive datagrams or fragments.

The SCHC end-points that handle the compression for nested Strata might differ for the same datagram, meaning that the payload of a given Stratum might be compressed/uncompressed by a different entity, possibly in a different node. It results that the degree of compression (the number of Strata) for a given datagram may vary as the datagram progresses through the layers inside a node and then through the network.

4.2. Discriminator

The key to determine how to decompress a SCHC Control Header in a stratum is called a Discriminator.

The Discriminator is typically extrinsic to the stratum data.

It may be found in the datagram context, e.g., the ID of the interface, VLAN, SSID, or PPP SCHC Instance on which the datagram is received.

It may also be received in the datagram, natively or uncompressed from a nesting stratum, e.g.:

- * A source and destination MAC or IP addresses of the datagram carrying SCHC datagrams

- * A source and destination port number of the transport layer carrying SCHC datagrams
- * A next header field
- * An MPLS label
- * A TLS Association
- * Any other kind of connection id.

The Discriminator enables to determine the SCHC end-point that is used to decompress the SCHC Control Header, called a SCHC Control end-point.

Once uncompressed, the SCHC Control Header enables to determine the SCHC end-point, called a SCHC Data end-point, that is used to restore the datagram data that is compressed in the stratum.

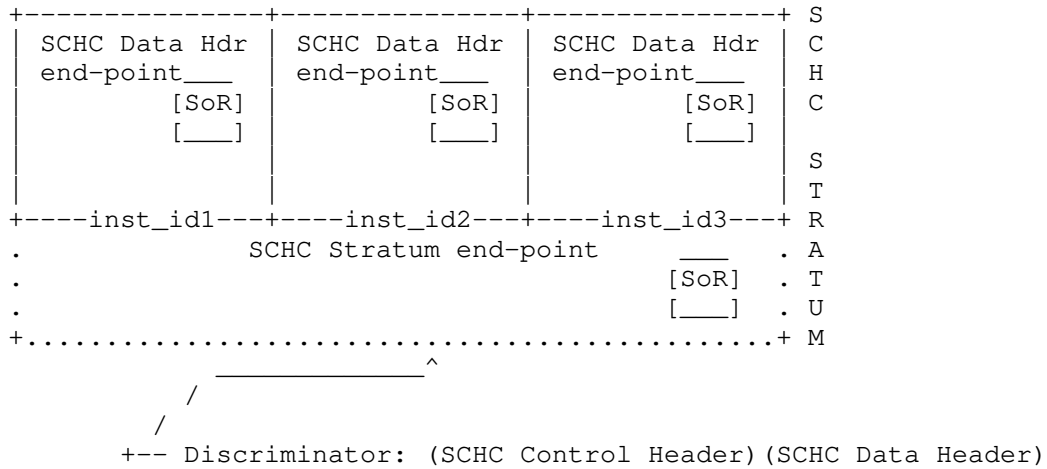
4.3. SCHC Control End-Point

The SCHC Control end-point manages the SCHC Control Headers and provides the information and the selection of a SCHC Data end-point.

The rules for that end-point might be such that all the fields in the SCHC Control Header are well-known, in which case the header is fully elided in the stratum data and recreated from the rules.

The rules might also leverage intrinsic data that is found in-line in the stratum data, in which case the first bits of the stratum data are effectively residue to the compression of the SCHC Control Header. Finally, the rules may leverage extrinsic data as the Discriminator does.

Figure 1 illustrates the case where a given stratum may compress multiple protocols SCHC Instances, each corresponding to a different SCHC Data end-point.



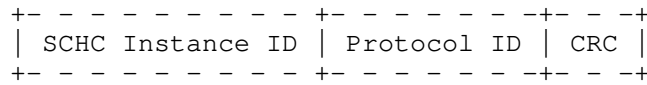
Each SCHC Data end-point uses its own Set of Rules, but share the same SCHC Control Header.

Figure 1: SCHC end-points for a stratum

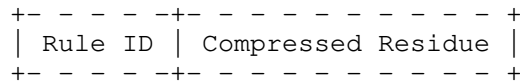
4.3.1. SCHC Control Header

The SCHC Control Header carries information that is required for the SCHC strata operation. For example, it selects the correct end-point and checks the validity of the datagram. There IS NOT always a RuleID if there is only one Rule for the SCHC Control Header, whose length is 0. The SCHC Control Header format is not fixed, and the SoR MUST have one or more Rules describing the formats. SCHC Control Header contains different fields. For the end-point, if the SCHC Control Header identifies, e.g., the next protocol in the stack, the format of the SCHC Control Header may be represented as shown in Figure 2.

Non-compressed SCHC Control Header Format:



SCHC Control Header Compressed:



Rule uses to compressed the SCHC Control Header:

RuleID

FID	FL	POS	DI	TV	MO	CDA
SCHC.sesid	10	1	Bi	0x00	MSB(7)	LSB
SCHC.proto	8	1	Bi	value	equal	not-sent
SCHC.CRC	8	1	Bi		ignore	value-sent

Figure 2: Example of SCHC Control Header Format and the corresponding Rule

In this example the Rule defines:

- * A SCHC InstanceID is 10 bits length and it is used to identify the SoR used for this end-point of SCHC.
- * A Protocol ID in 1-byte length giving the value send in the layer below the SCHC datagram to identify the uncompressed protocol stack.
- * And A CRC. The CRC field is 8 bits length and covers the SCHC Control Header and the SCHC datagram from error. When it is elided by the compression, the layer-4 checksum MUST be replaced by another validation sequence.

4.4. SCHC Data End-Point

A SCHC Data end-point handles this node's side of a SCHC Data instance? It is characterized by a particular SoR common with the corresponding distant end-point. The [rfc8724] defines a protocol operation between a pair of peers. In a SCHC strata, several SCHC end-points may contain different SoR.

When the SCHC Device is a highly constrained unit, there is typically only one end-point for that Device, and all the traffic from and to the device is exchanged with the same Network Gateway. All the

traffic can thus be implicitly associated with the single end-point that the device supports, and the Device does not need to manipulate the concept. For that reason, SCHC avoids to signal explicitly the end-point identification in its data datagrams.

The Network Gateway, on the other hand, maintains multiple end-points, one per SCHC Device. The end-point is derived from the lower layer, typically the source of an incoming SCHC datagram as a discriminator in the Figure 1. The end-point is used in particular to select the set of rules that apply to the SCHC Device, and the current state of their exchange, e.g., timers and previous fragments.

4.4.1. SCHC Data Header

As defined in section 5.1 of [rfc8724], a SCHC datagram (or packet) is composed of the compressed header called the SCHC Data Header followed by the uncompressed remainder payload from the original datagram (or packet). The SCHC Data Header, contains the data generated by the SCHC operation. It is composed of a RuleID followed by the content described in the Rule. The content may be a C/D datagram, a F/R datagram, a CORECONF_Management or a Non Compressed datagram.

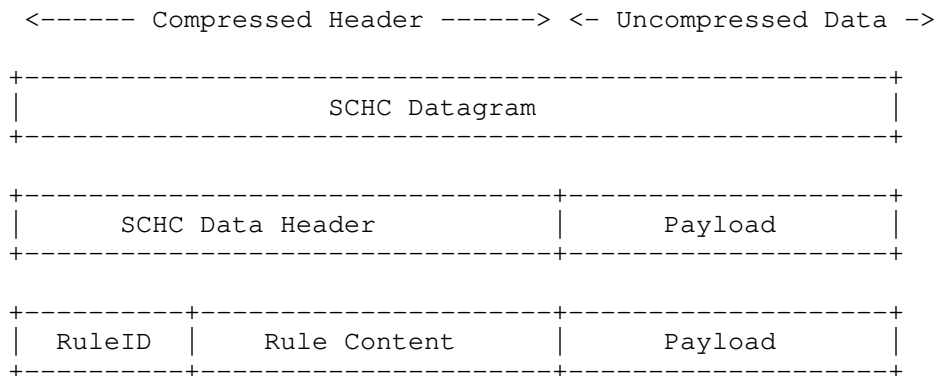
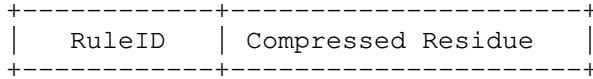


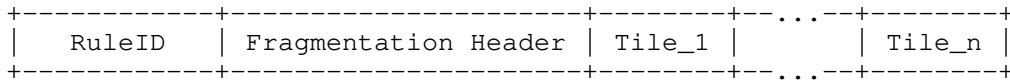
Figure 3: SCHC Datagram

Figure 4 shows the compressed header format that is composed of the RuledID and a Compressed Residue, which is the output of compressing a datagram header with a Rule.

C/D Compressed SCHC Data Header:



F/R Compressed SCHC Data Header:



CORECONF_Management SCHC Data Header:

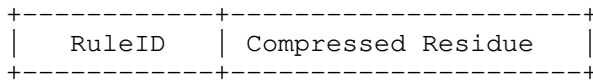


Figure 4: SCHC Data Header

4.5. SCHC Profiles

A SCHC profile is the specification to adapt the use of SCHC with the necessities of the technology to which it is applied. In the case of star topologies and because LPWAN technologies [rfc8376] have strict yet distinct constraints, e.g., in terms of maximum frame size, throughput, and directionality, also a SCHC end-point and the fragmentation model with the parameters' values for its use.

Appendix D. "SCHC Parameters" of [rfc8724] lists the information that an LPWAN technology-specific document must provide to profile SCHC fragmentation for that technology.

As an example, [rfc9011] provides the SCHC fragmentation profile for LoRaWAN networks.

4.6. SCHC Operation

The SCHC operation requires a shared sense of which SCHC Device is Uplink (Dev to App) and which is Downlink (App to Dev), see [rfc8376]. In a star deployment, the hub is always considered Uplink and the spokes are Downlink. The expectation is that the hub and spoke derive knowledge of their role from the network configuration and SCHC does not need to signal which is hub thus Uplink vs. which is spoke thus Downlink. In other words, the link direction is determined from extrinsic properties, and is not advertised in the protocol.

Nevertheless, SCHC is very generic and its applicability is not limited to star-oriented deployments and/or to use cases where applications are very static and the state provisioned in advance. In particular, a peer-to-peer (P2P) SCHC end-point (see Section 4.4) may be set up between peers of equivalent capabilities, and the link direction cannot be inferred, either from the network topology nor from the device capability.

In that case, by convention, the device that initiates the connection that sustains the SCHC end-point is considered as being Downlink, i.e. it plays the role of the Dev in [rfc8724].

This convention can be reversed, e.g., by configuration, but for proper SCHC operation, it is required that the method used ensures that both ends are aware of their role, and then again this determination is based on extrinsic properties.

4.6.1. SCHC Rules

SCHC Rules are a description of the header protocols fields, into a list of Field Descriptors. The [rfc8724] gives the format of the Rule description for C/D, F/R and non-compression. In the same manner the SCHC Control Header and SCHC CORECONF_Management will use the [rfc8724] field descriptors to compress the format information.

Each type of Rule is identified with a RuleID. There are different types of Rules: C/D, F/R, SCHC Control Header, CORECONF_Management and No Compression. Notice that each Rule type used an independent range of RuleID to identify its rules.

A Rule does not describe how the compressor parses a datagram header. Rules only describe the behavior for each header field.

SCHC Action. ToDo

4.6.2. SoR identification

ToDo

4.7. SCHC Management

RFC9363 writes that only the management can be done by the two entities of the end-point, and other SoR cannot be manipulated.

Management rules are explicitly define in the SoR, see Figure 5. They are compression Rules for CORECONF messages to get or modify the SoR of the end-point. The management can be limited with the [I-D.ietf-schc-access-control] access definition.

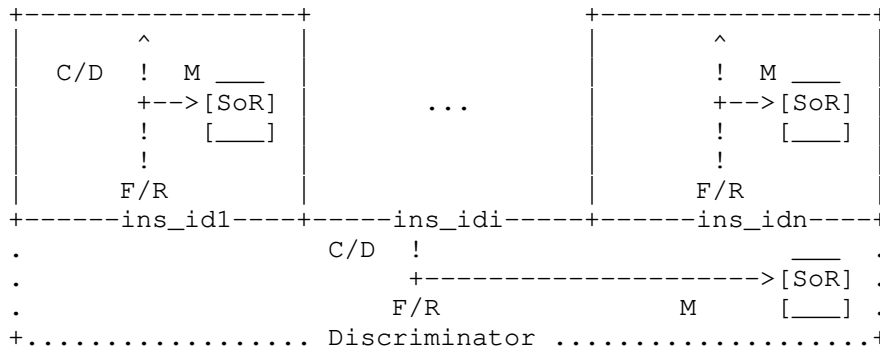


Figure 5: Inband Management

4.7.1. SCHC Instance Manager

The SCHC Instance Manager provides the management of SCHC end-points, the SoR of each end-point and the dialog between hosts to keep the SCHC synchronization, and the establishment of SCHC Instances with peer nodes. Changes that involve the SoR must be transactional, in a way that ensures that the compression and decompression of a datagram is done with the same SoR on every end-points.

The management of the SCHC end-points includes the capability for the end-points to modify the common SoR, by:

- * modifyng rules values (such as TV, MO or CDA) in existing rules,
- * adding or
- * removing rules.

The rule management uses the CORECONF interface based on CoAP. The management traffic is carried as SCHC compressed datagrams tagged to some specific rule IDs.

4.7.2. SCHC Data Model

A SCHC end-point, summarized in the Figure 6, implies C/D and/or F/R and CORECONF_Management and SCHC end-points Rules present in both end and that both ends are provisioned with the same SoR.



Figure 6: Summarized SCHC elements

A common rule representation that expresses the SCHC rules in an interoperable fashion is needed to be able to provision end-points from different vendors to that effect, [rfc9363] defines a rule representation using the YANG [rfc7950] formalism.

[rfc9363] defines a YANG data model to represent the rules. This enables the use of several protocols for rule management, such as NETCONF[RFC6241], RESTCONF[RFC8040], and CORECONF[I-D.ietf-core-comi]. NETCONF uses SSH, RESTCONF uses HTTPS, and CORECONF uses CoAP(s) as their respective transport layer protocols. The data is represented in XML under NETCONF, in JSON[RFC8259] under RESTCONF and in CBOR[RFC8949] under CORECONF.

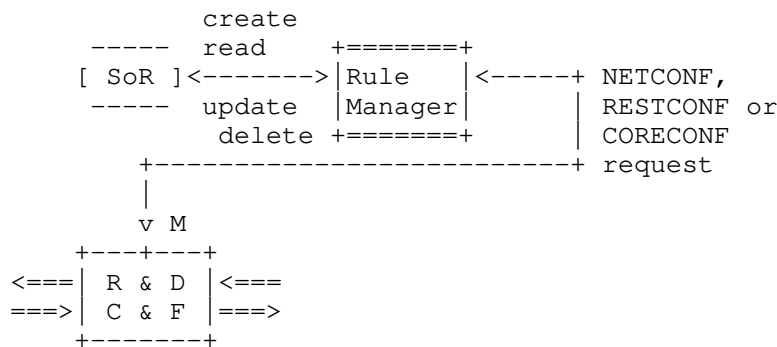


Figure 7: Summarized SCHC elements

The Rule Manager (RM) is in charge of handling data derived from the YANG Data Model and apply changes to the context and SoR of each SCHC end-point Figure 7.

The RM is an Application using the Internet to exchange information, therefore:

- * for the network-level SCHC, the communication does not require routing. Each of the end-points having an RM and both RMs can be viewed on the same link, therefore wellknown Link Local addresses can be used to identify the Device and the core RM. L2 security MAY be deemed as sufficient, if it provides the necessary level of protection.
- * for application-level SCHC, routing is involved and global IP addresses SHOULD be used. End-to-end encryption is RECOMMENDED.

Management messages can also be carried in the negotiation protocol, for instance, the [I-D.ietf-schc-over-ppp] proposes a solution. The RM traffic may be itself compressed by SCHC: if CORECONF protocol is used, [rfc8824] can be applied.

5. SCHC Architecture

As described in [rfc8824], SCHC combining several SCHC end-points. The [rfc8724] states that a SCHC end-point needs the rules to process C/D and F/R before the SCHC Instance starts and that the SoR of the end-point control layer cannot be modified. However, the rules may be updated in certain end-points to improve the performance of C/D, F/R, or CORECONF_Management. The [I-D.ietf-schc-access-control] defines the possible modifications and who can modify, update, create and delete Rules or part of them in the end-points' SoR.

As represented in Figure 8, the compression of the IP and UDP headers may be operated by a network SCHC end-point whereas the end-to-end compression of the application payload happens between the Device and the application. The compression of the application payload may be split in two end-points to deal with the encrypted portion of the application PDU. Fragmentation applies before LPWAN transmission layer.

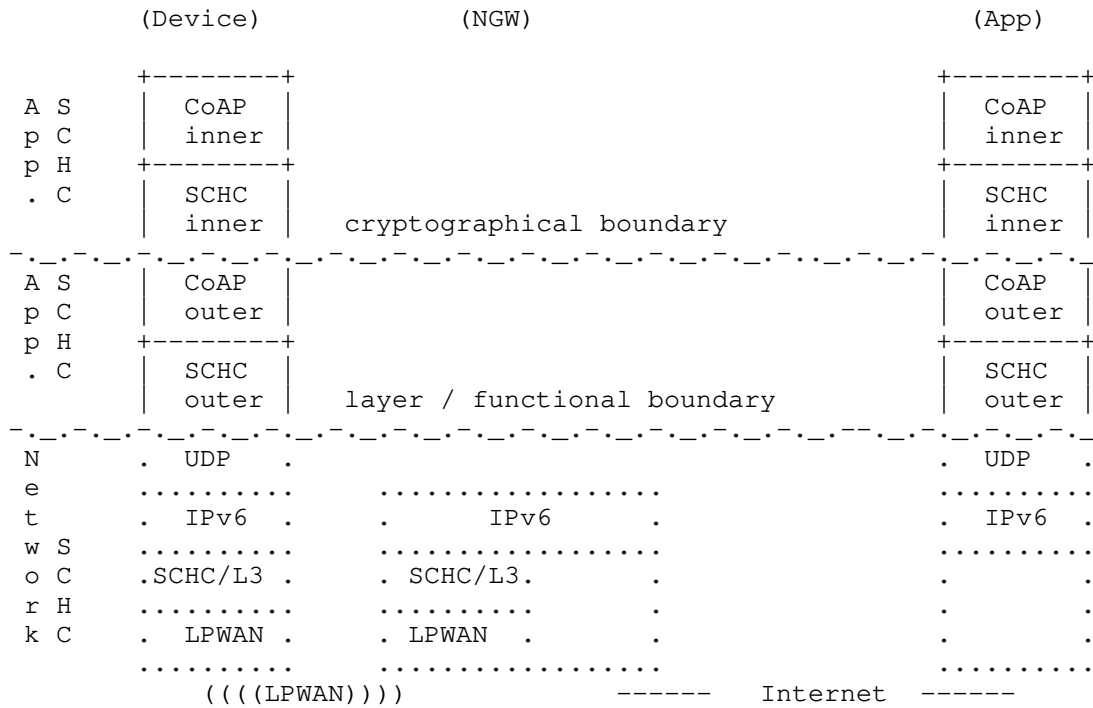
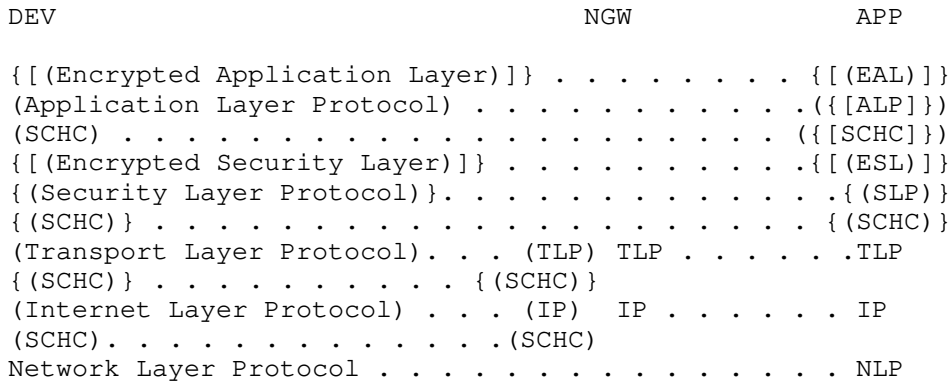


Figure 8: Different SCHC end-points in a global system

This document defines a generic architecture for SCHC that can be used at any of these levels. The goal of the architectural document is to orchestrate the different protocols and data model defined by the LPWAN and SCHC working groups to design an operational and interoperable framework for allowing IP application over constrained networks.

The Figure 9 shows the protocol stack and the corresponding SCHC stratas enabling the compression of the different protocol headers. The SCHC Control Header eases the introduction of intermediary host in the end-to-end communication transparently. All the SCHC Control Headers are compressed and in some cases are elided, for example for LPWAN networks. The layers using encryption does not have a SCHC Control Header in the middle because they are the same entity. Figure 10 shows an example of an IP/UDP/CoAP in an LPWAN network.

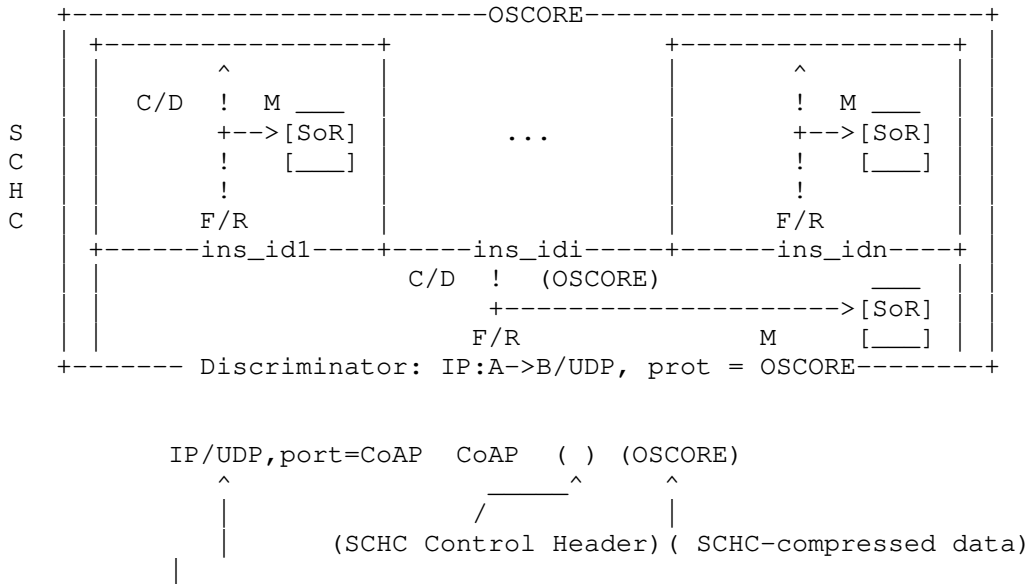


Where: {} Optional; [] Encrypted; () Compressed.

Figure 9: SCHC Architecture

In Figure 9, each line represents a layer or a stratum, parentheses surround a compressed header, and if it is optional, it has curly brackets. All the SCHC strata are compressed. Square brackets represent the encrypted data; if the encryption is optional, curly brackets precede the square brackets.

Figure 10 represents the stack of SCHC end-points that operate over 3 strata, one for OSCORE, one for CoAP, and one for IP and UDP.



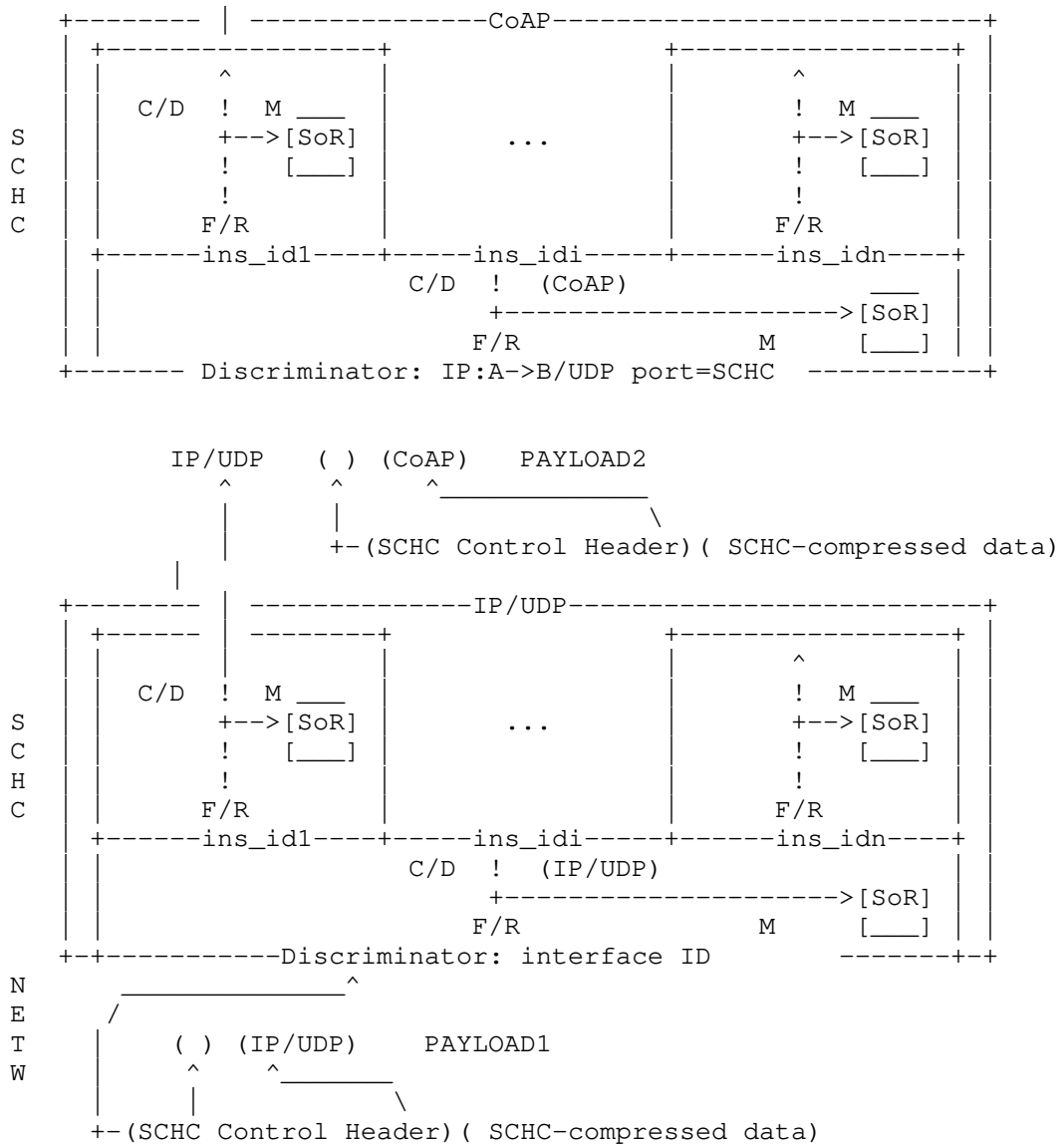


Figure 10: SCHC Strata Example

6. The Static Context Header Compression

SCHC [rfc8724] specifies an extreme compression capability based on a description that must match on the compressor and decompressor side. This description comprises a set of Compression/Decompression (C/D) rules.

The SCHC Parser analyzes incoming datagrams and creates a list of fields that it matches against the compression rules. The rule that matches is used to compress the datagram, and the rule identifier (RuleID) is transmitted together with the compression residue to the decompressor. Based on the RuleID and the residue, the decompressor can rebuild the original datagram and forward it in its uncompressed form over the Internet. When no Rule matches the header, the No Compression Rule is used. When several Rules match the header the implementation must choose one. How it is done or based on which parameters is out of the scope of this document. SCHC compresses datagrams and there is no notion of flows.

[rfc8724] also provides a Fragmentation/Reassembly (F/R) capability to cope with the maximum and/or variable frame size of a Link, which is extremely constrained in the case of an LPWAN network.

If a SCHC-compressed datagram is too large to be sent in a single Link-Layer PDU, the SCHC fragmentation can be applied on the compressed datagram. The process of SCHC fragmentation is similar to that of compression; the fragmentation rules that are programmed for this Device are checked to find the most appropriate one, regarding the SCHC datagram size, the link error rate, and the reliability level required by the application.

The ruleID allows to determine if it is a compression or fragmentation rule or any other type of Rule.

6.1. SCHC over Network Technologies

SCHC can be used in multiple environments and multiple protocols. It was designed by default to work on native MAC frames with LPWAN technologies such as LoRaWAN[rfc9011], IEEE std 802.15.4 [I-D.ietf-6lo-schc-15dot4], and SigFox[rfc9442].

To operate SCHC over Ethernet, IPv6, and UDP, the definition of, respectively, an Ethertype, an IP Protocol Number, and a UDP Port Number are necessary, more in [I-D.ietf-intarea-schc-protocol-numbers]. In either case, there's a need for a SCHC Control Header that is sufficient to identify the SCHC peers (endpoints) and their role (device vs. app), as well as the SCHC Instance between those peers that the datagram pertains to.

In either of the above cases, the expectation is that the SCHC Control Header is transferred in a compressed form. This implies that the rules to uncompress the header are well known and separate from the rules that are used to uncompress the SCHC Data Header. The expectation is that for each stratum, the format of the SCHC Control Header and the compression rules are well known, with enough information to identify the SCHC Instance at that stratum, but there is no expectation that they are the same across strata.

6.1.1. SCHC over PPP

The LPWAN architecture (Figure 15) generalizes the model to any kind of peers. In the case of more capable devices, a SCHC Device may maintain more than one end-point with the same peer, or a set of different peers. Since SCHC does not signal the end-point in its datagrams, the information must be derived from a lower layer point to point information. For end-point, the SCHC end-point control can be associated one-to-one with a tunnel, a TLS SCHC Instance, or a TCP or a PPP connection.

For end-point, [I-D.ietf-schc-over-ppp] describes a type of deployment where the C/D and/or F/R operations are performed between peers of equal capabilities over a PPP [rfc2516] connection. SCHC over PPP illustrates that with SCHC, the protocols that are compressed can be discovered dynamically and the rules can be fetched on-demand using CORECONF messages Rules, ensuring that the peers use the exact same set of rules.

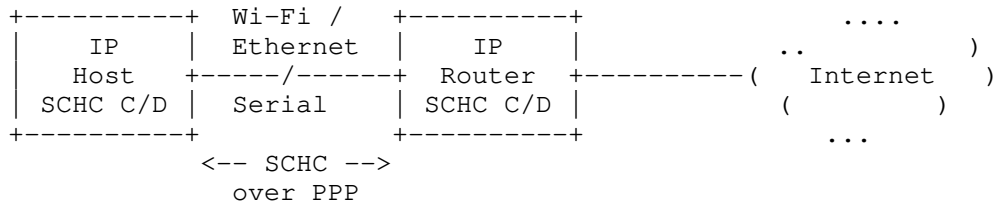


Figure 11: PPP-based SCHC Deployment

In that case, the SCHC end-point is derived from the PPP connection. This means that there can be only one end-point per PPP connection, and that all the flow and only the flow of that end-point is exchanged within the PPP connection. As discussed in Section 7, the Uplink direction is from the node that initiated the PPP connection to the node that accepted it.

6.1.2. SCHC over Ethernet

Before the SCHC compression takes place, the SCHC Control Header showed in the Figure 12, is virtually inserted before the real protocol header and data that are compressed in the SCHC Instance, e.g. a IPv6 in this figure.

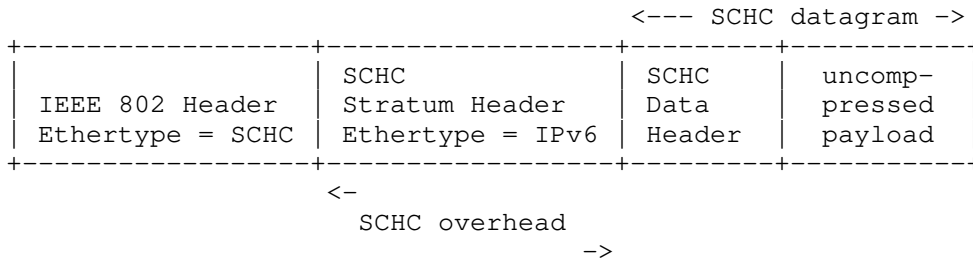


Figure 12: SCHC over Ethernet

6.1.3. SCHC over IPv6

In the case of IPv6, the expectation is that the Upper Layer Protocol (ULP) checksum can be elided in the SCHC compression of the ULP, because the SCHC Control Header may have its own checksum that protects both the SCHC Control Header and the whole ULP, header and payload.

The SCHC Control Header between IPv6 and the ULP is not needed because of the Next Header field on the IPv6 header format.

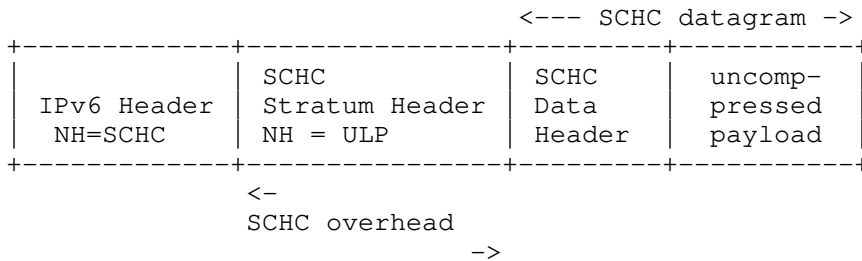


Figure 13: SCHC over IPv6

In the air, both the SCHC Control Header and the ULP are compressed. The SCHC Instance endpoints are typically identified by the source and destination IP addresses. If the roles are well-known, then the endpoint information can be elided and deduced from the IP header. If there is only one SCHC Instance, it can be elided as well, otherwise a rule and residue are needed to extract the SCHC Instance ID.

6.1.4. SCHC over UDP

When SCHC operates over the Internet, middleboxes may block datagrams with a next header that is SCHC. To avoid that issue, it would be desirable to prepend a UDP header before the SCHC Control Header as shown in figure Figure 14.

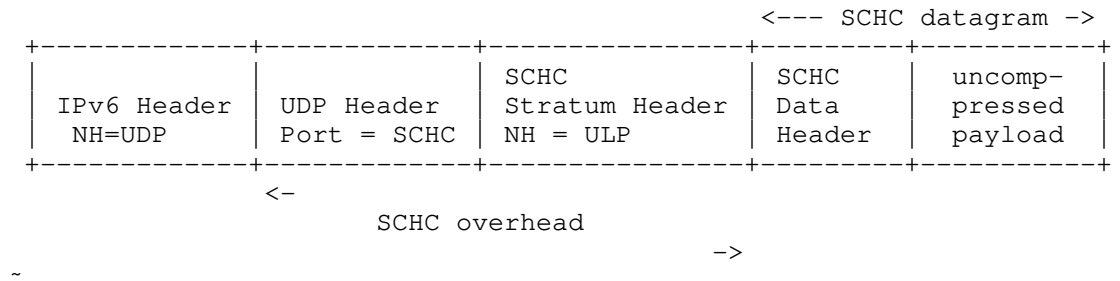


Figure 14: SCHC over UDP

In that case, the destination port can indicate SCHC as in an header chain, and the source port can indicate the SCHC Instance in which case it can be elided in the compressed form of the SCHC Control Header. The UDP checksum protects both the SCHC Control Header and the whole ULP, so the SCHC and the ULP checksums can both be elided. In other words, in the SCHC over UDP case, the SCHC Control Header can be fully elided, but the datagram must carry the overhead of a full UDP header.

7. SCHC Endpoints for LPWAN Networks

Section 3 of [rfc8724] depicts a typical network architecture for an LPWAN network, simplified from that shown in [rfc8376] and reproduced in Figure 15.

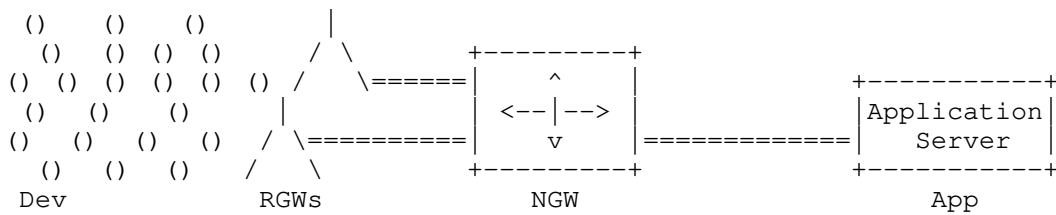


Figure 15: Typical LPWAN Network Architecture

Typically, an LPWAN network topology is star-oriented, which means that all datagrams between the same source-destination pair follow the same path from/to a central point. In that model, highly constrained Devices (Dev) exchange information with LPWAN Application Servers (App) through a central Network Gateway (NGW), which can be powered and is typically a lot less constrained than the Devices. Because Devices embed built-in applications, the traffic flows to be compressed are known in advance and the location of the C/D and F/R functions (e.g., at the Dev and NGW), and the associated rules, can be pre provisioned in the system before use.

7.1. SCHC Device Lifecycle

In the context of LPWANs, the expectation is that SCHC rules are associated with a physical device that is deployed in a network. This section describes the actions taken to enable an automatic commissioning of the device in the network.

7.1.1. Device Development

The expectation for the development cycle is that message formats are documented as a data model that is used to generate rules. Several models are possible:

1. In the application model, an interface definition language and binary communication protocol such as Apache Thrift is used, and the parser code includes the SCHC operation. This model imposes that both ends are compiled with the generated structures and linked with generated code that represents the rule operation.
2. In the device model, the rules are generated separately. Only the device-side code is linked with generated code. The Rules are published separately to be used by a generic SCHC engine that operates in a middle box such as a SCHC gateway.
3. In the protocol model, both endpoint generate a datagram format that is imposed by a protocol. In that case, the protocol itself is the source to generate the Rules. Both ends of the SCHC

compression are operated in middle boxes, and special attention must be taken to ensure that they operate on the compatible SoR, basically the same major version of the same SoR.

Depending on the deployment, the tools that generate the Rules should provide knobs to optimize the SoR, e.g., more rules vs. larger residue.

7.1.2. Rules Publication

In the device model and in the protocol model, at least one of the endpoints must obtain the SoR dynamically. The expectation is that the SoR are published to a reachable repository and versioned (minor, major). Each SoR should have its own Uniform Resource Names (URN) [RFC8141] and a version.

The SoR should be authenticated to ensure that it is genuine, or obtained from a trusted app store. A corrupted SoR may be used for multiple forms of attacks, more in Section 8.

7.1.3. SCHC Device Deployment

The device and the network should mutually authenticate themselves. The autonomic approach [RFC8993] provides a model to achieve this at scale with zero touch, in networks where enough bandwidth and compute are available. In highly constrained networks, one touch is usually necessary to program keys in the devices.

The initial handshake between the SCHC endpoints should comprise a capability exchange whereby URN and the version of the SoR are obtained or compared. SCHC may not be used if both ends can not agree on an URN and a major version. Manufacturer Usage Descriptions (MUD) [RFC8520] may be used for that purpose in the device model.

Upon the handshake, both ends can agree on a SoR, their role when the rules are asymmetrical, and fetch the SoR if necessary. Optionally, a node that fetched a SoR may inform the other end that it is ready for transmission.

7.1.4. SCHC Device Maintenance

URN update without device update (bug fix) FUOTA => new URN => reprovisioning

7.1.5. SCHC Device Decommissioning

Signal from device/vendor/network admin

8. Security Considerations

SCHC is sensitive to the rules that could be abused to form arbitrary long messages or as a form of attack against the C/D and/or F/R functions, say to generate a buffer overflow and either modify the Device or crash it. It is thus critical to ensure that the rules are distributed in a fashion that is protected against tempering, e.g., encrypted and signed.

9. IANA Consideration

This document has no request to IANA

10. Acknowledgements

The authors would like to thank (in alphabetic order): Laurent Toutain

11. References

11.1. 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/rfc/rfc2119>>.
- [RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017, <<https://www.rfc-editor.org/rfc/rfc8141>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [rfc8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <<https://www.rfc-editor.org/rfc/rfc8724>>.

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

11.2. Informative References

- [I-D.ietf-6lo-schc-15dot4]
Gomez, C. and A. Minaburo, "Transmission of SCHC-compressed packets over IEEE 802.15.4 networks", Work in Progress, Internet-Draft, draft-ietf-6lo-schc-15dot4-11, 14 October 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-6lo-schc-15dot4-11>>.
- [I-D.ietf-core-comi]
Veillette, M., Van der Stok, P., Pelov, A., Bierman, A., and C. Bormann, "CoAP Management Interface (CORECONF)", Work in Progress, Internet-Draft, draft-ietf-core-comi-20, 6 May 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-20>>.
- [I-D.ietf-intarea-schc-protocol-numbers]
Moskowitz, R., Card, S. W., Wiethuechter, A., and P. Thubert, "Protocol Numbers for SCHC", Work in Progress, Internet-Draft, draft-ietf-intarea-schc-protocol-numbers-02, 8 April 2024, <<https://datatracker.ietf.org/doc/html/draft-ietf-intarea-schc-protocol-numbers-02>>.
- [I-D.ietf-schc-access-control]
Minaburo, A., Toutain, L., and I. Martinez, "SCHC Access Control", Work in Progress, Internet-Draft, draft-ietf-schc-access-control-00, 13 December 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-schc-access-control-00>>.
- [I-D.ietf-schc-over-ppp]
Thubert, P., "SCHC over PPP", Work in Progress, Internet-Draft, draft-ietf-schc-over-ppp-00, 25 July 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-schc-over-ppp-00>>.
- [rfc2516] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516, DOI 10.17487/RFC2516, February 1999, <<https://www.rfc-editor.org/rfc/rfc2516>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/rfc/rfc6241>>.
- [rfc7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/rfc/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/rfc/rfc8040>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/rfc/rfc8259>>.
- [rfc8376] Farrell, S., Ed., "Low-Power Wide Area Network (LPWAN) Overview", RFC 8376, DOI 10.17487/RFC8376, May 2018, <<https://www.rfc-editor.org/rfc/rfc8376>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/rfc/rfc8520>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/rfc/rfc8949>>.
- [RFC8993] Behringer, M., Ed., Carpenter, B., Eckert, T., Ciavaglia, L., and J. Nobre, "A Reference Model for Autonomic Networking", RFC 8993, DOI 10.17487/RFC8993, May 2021, <<https://www.rfc-editor.org/rfc/rfc8993>>.
- [rfc9011] Gimenez, O., Ed. and I. Petrov, Ed., "Static Context Header Compression and Fragmentation (SCHC) over LoRaWAN", RFC 9011, DOI 10.17487/RFC9011, April 2021, <<https://www.rfc-editor.org/rfc/rfc9011>>.
- [rfc9363] Minaburo, A. and L. Toutain, "A YANG Data Model for Static Context Header Compression (SCHC)", RFC 9363, DOI 10.17487/RFC9363, March 2023, <<https://www.rfc-editor.org/rfc/rfc9363>>.

[rfc9442] Z̄ã±iga, J., Gomez, C., Aguilar, S., Toutain, L., Cãspedes, S., Wistuba, D., and J. Boite, "Static Context Header Compression (SCHC) over Sigfox Low-Power Wide Area Network (LPWAN)", RFC 9442, DOI 10.17487/RFC9442, July 2023, <<https://www.rfc-editor.org/rfc/rfc9442>>.

Authors' Addresses

Alexander Pelov
IMT Atlantique
rue de la Chataigneraie
35576 Cesson-Sevigne Cedex
France
Email: alexander.pelov@imt-atlantique.fr

Pascal Thubert
06330 Roquefort les Pins
France
Email: pascal.thubert@gmail.com

Ana Minaburo
Consultant
35510 Cesson-Sevigne Cedex
France
Email: anaminaburo@gmail.com

SCHC Working Group
Internet-Draft
Obsoletes: 8824 (if approved)
Intended status: Standards Track
Expires: 25 April 2024

M. Tiloca
RISE AB
L. Toutain
IMT Atlantique
I. Martinez
Nokia Bell Labs
A. Minaburo
Consultant
23 October 2023

Static Context Header Compression (SCHC) for the Constrained Application
Protocol (CoAP)
draft-tiloca-schc-8824-update-02

Abstract

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

Discussion Venues

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

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 25 April 2024.

Copyright Notice

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

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

Table of Contents

1.	Introduction	3
1.1.	Terminology	6
2.	SCHC Applicability to CoAP	6
3.	CoAP Headers Compressed with SCHC	8
3.1.	Differences between CoAP and UDP/IP Compression	9
4.	Compression of CoAP Header Fields	10
4.1.	CoAP Version Field	10
4.2.	CoAP Type Field	10
4.3.	CoAP Code Field	10
4.4.	CoAP Message ID Field	11
4.5.	CoAP Token Field	11
5.	Compression of CoAP Options	11
5.1.	CoAP Option Content-Format and Accept Fields	12
5.2.	CoAP Option Max-Age, Uri-Host, and Uri-Port Fields	12
5.3.	CoAP Option Uri-Path and Uri-Query Fields	12
5.3.1.	Variable Number of Path or Query Elements	14
5.4.	CoAP Option Size1, Size2, Proxy-URI, and Proxy-Scheme Fields	14
5.5.	CoAP Option ETag, If-Match, If-None-Match, Location-Path, and Location-Query Fields	14

5.6.	CoAP Option Hop-Limit Field	15
5.7.	CoAP Option Echo Field	15
5.8.	CoAP Option Request-Tag Field	15
5.9.	CoAP Option EDHOC Field	16
6.	Compression of CoAP Extensions	16
6.1.	Block	16
6.2.	Observe	16
6.3.	No-Response	17
6.4.	OSCORE	17
7.	Compression of the CoAP Payload Marker	23
8.	Examples of CoAP Header Compression	23
8.1.	Mandatory Header with CON Message	23
8.2.	OSCORE Compression	25
8.3.	Example OSCORE Compression	29
9.	CoAP Header Compression with Proxies	41
9.1.	Without End-to-End Security	41
9.2.	With End-to-End Security	42
10.	Examples of CoAP Header Compression with Proxies	43
10.1.	Without End-to-End Security	46
10.2.	With End-to-End Security	53
11.	Security Considerations	68
12.	IANA Considerations	70
13.	References	70
13.1.	Normative References	70
13.2.	Informative References	71
Appendix A.	YANG Data Model	72
Acknowledgments	72
Authors' Addresses	72

1. Introduction

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

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

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

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

A Rule describes the packets' entire header with an ordered list of Field Descriptors (see Section 7 of [RFC8724]). Thereby, each description contains the Field ID (FID), Field Length (FL), and Field Position (FP), as well as a Direction Indicator (DI) (upstream, downstream, and bidirectional) and some associated Target Values (TVs). The DI is used for compression to give the best TV to the FID when these values differ in their transmission direction. Therefore, a field may be described several times in the same Rule.

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

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

- * send the field value (value-sent),
- * send nothing (not-sent),
- * send some Least Significant Bits (LSBs) of the field, or
- * send an index (mapping-sent).

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

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

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

In particular, this document replaces and obsoletes [RFC8824] as follows.

- * It provides clarifications and amendments to the original specification text, based on collected feedback and reported errata.
- * It clarifies the SCHC compression for the CoAP options Size1, Size2, Proxy-Uri, and Proxy-Scheme (see Section 5.4).
- * It defines the SCHC compression for the CoAP option Hop-Limit (see Section 5.6).
- * It defines the SCHC compression for the recently defined CoAP options Echo (see Section 5.7), Request-Tag (see Section 5.8), EDHOC (see Section 5.9), as well as Q-Block1 and Q-Block2 (see Section 6.1).
- * It updates the SCHC compression processing for the CoAP option OSCORE (see Section 6.4), in the light of recent developments related to the security protocol OSCORE as defined in [I-D.ietf-core-oscore-key-update] and [I-D.ietf-core-oscore-groupcomm].
- * It clarifies how the SCHC compression handles the CoAP payload marker (see Section 7).
- * It defines the SCHC compression of CoAP headers in the presence of CoAP proxies (see Section 9), for which examples are provided (see Section 10).

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Readers are expected to be familiar with the terms and concepts related to the SCHC framework [RFC8724], the web-transfer protocol CoAP [RFC7252], and the security protocols OSCORE [RFC8613] and Group OSCORE [I-D.ietf-core-oscore-groupcomm].

2. SCHC Applicability to CoAP

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

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

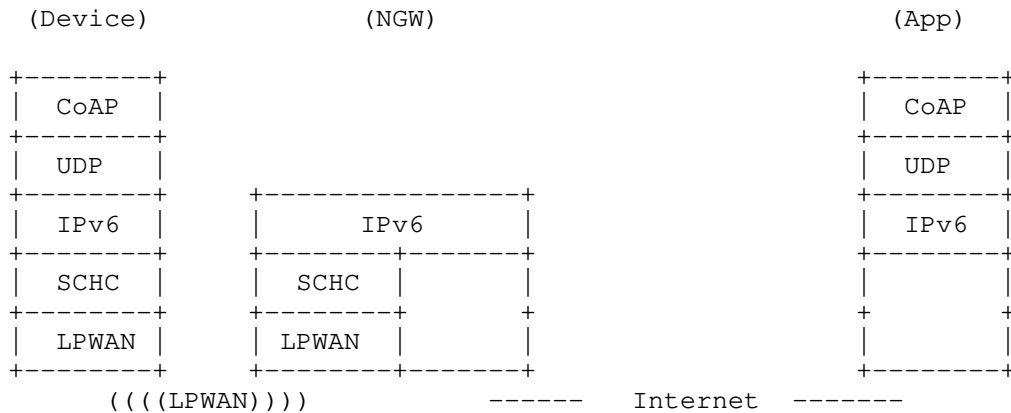


Figure 1: Compression/Decompression at the LPWAN Boundary

Figure 1 shows the use of SCHC header compression above Layer 2 in the Device and the NGW. The SCHC layer receives non-encrypted packets and can apply compression Rules to all the headers in the stack. On the other end, the NGW receives the SCHC packet and reconstructs the headers using the Rule and the Compression Residue.

After the decompression, the NGW forwards the IPv6 packet toward the destination. The same process applies in the other direction when a non-encrypted packet arrives at the NGW. Thanks to the IP forwarding based on the IPv6 prefix, the NGW identifies the Device and compresses headers using the Device's Rules.

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

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

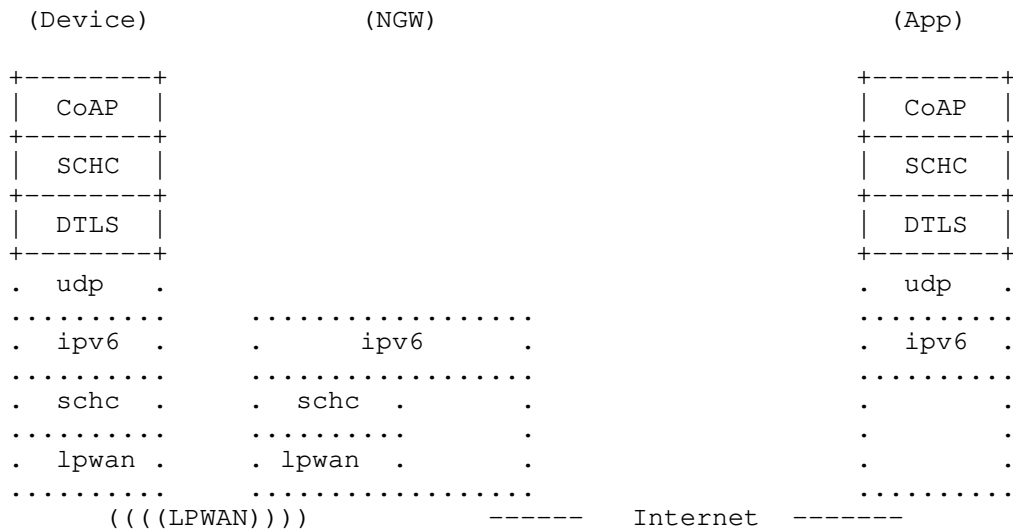


Figure 2: Standalone CoAP End-to-End Compression/Decompression

The third example depicted in Figure 3 shows the use of Object Security for Constrained RESTful Environments (OSCORE) [RFC8613]. In this case, SCHC needs two Rules to compress the CoAP header. A first Rule focuses on the Inner header. The result of this first compression is encrypted using the OSCORE mechanism. Then, a second Rule compresses the Outer header, including the CoAP Option OSCORE.

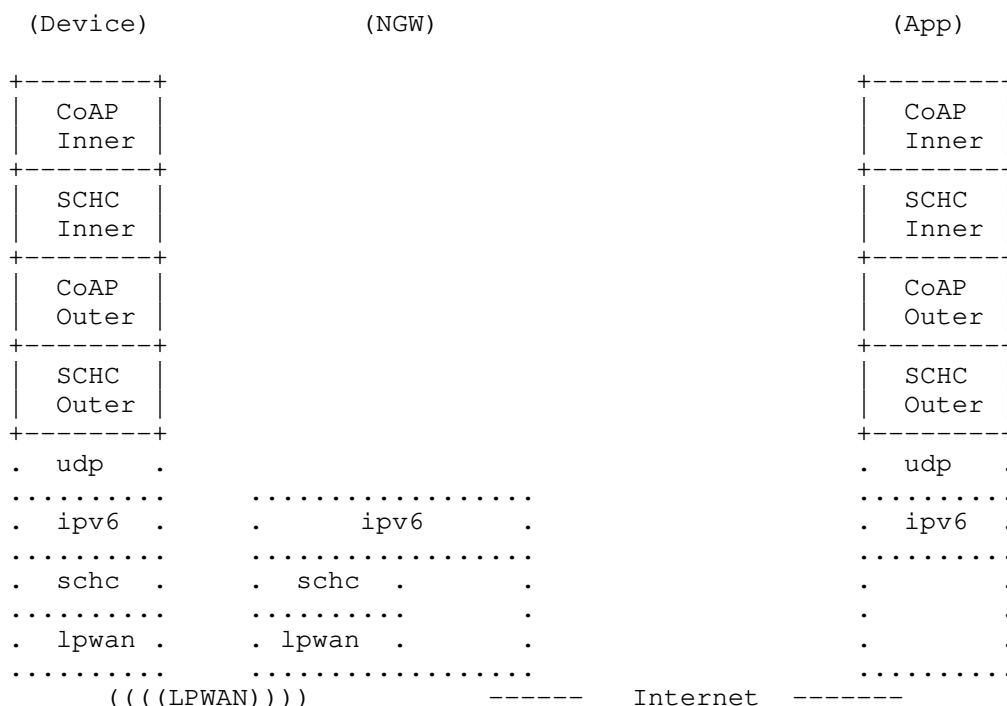


Figure 3: OSCORE Compression/Decompression

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

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

3. CoAP Headers Compressed with SCHC

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

As for other header compression protocols, when the compressor does not find a correct Rule to compress the header, the packet MUST be sent uncompressed using the RuleID dedicated to this purpose, and where the Compression Residue is the complete header of the packet. See Section 6 of [RFC8724].

3.1. Differences between CoAP and UDP/IP Compression

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

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

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

- * Even when a field is "symmetric" (i.e., found in both directions), the values carried in each direction are different. The compression may use a "match-mapping" MO to limit the range of expected values in a particular direction and reduce the Compression Residue's size. Through the DI, a Field Descriptor in the Rules splits the possible field value into two parts, one for each direction. For instance, if a client sends only Confirmable (CON) requests [RFC7252], the Type can be elided by compression, and the reply from the server may use one single bit to carry either the Acknowledgement (ACK) or Reset (RST) type. The field Code has the same behavior: the 0.0X code format value in a request and the Y.ZZ code format in a response.
- * In SCHC, the Rule defines the different header fields' length, so SCHC does not need to send it. In IPv6 and UDP headers, the fields have a fixed size, known by definition. On the other hand, some CoAP header fields have variable lengths, and the Rule description specifies it. For example, the size of the Token field may vary from 0 to 8 bytes, and the CoAP options rely on the Type-Length-Value encoding format to specify the size of the actual option value in bytes.

When doing SCHC compression of a variable-length field, Section 7.4.2 of [RFC8724] makes it possible to define a function for the Field Length in the Field Descriptor, in order to determine the length before compression. If the Field Length is unknown, the Rule will set it as a variable, and SCHC will send the compressed field's length in the Compression Residue.

- * A field can appear several times in the CoAP headers. This is typically the case for elements of a URI (path or queries). The SCHC specification [RFC8724] allows a FID to appear several times in the Rule and uses the Field Position (FP) to identify the correct instance, thereby removing the MO's ambiguity.
- * Field Lengths defined in CoAP can be too large when it comes to LPWAN traffic constraints. For instance, this is particularly true for the Message ID field and the Token field. SCHC uses different MOs to perform the compression (see Section 7.4 of [RFC8724]). In this case, SCHC can apply the Most Significant Bits (MSBs) MO to reduce the information carried on LPWANs.

4. Compression of CoAP Header Fields

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

4.1. CoAP Version Field

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

4.2. CoAP Type Field

CoAP [RFC7252] has four types of messages: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK), and Reset (RST).

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

4.3. CoAP Code Field

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

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

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

4.4. CoAP Message ID Field

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

4.5. CoAP Token Field

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

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

For the Token Value, SCHC MUST NOT send it as variable-size data in the Compression Residue, to avoid ambiguity with the Token Length. Therefore, SCHC MUST use the Token Length value to define the size of the Compression Residue. SCHC designates a specific function, "tk1", that the Rule MUST use to complete the Field Descriptor. During the decompression, this function returns the value contained in the Token Length field.

5. Compression of CoAP Options

CoAP defines options placed after the mandatory header and the Token field, ordered by option number (see Section 3 of [RFC7252]). Each option instance in a message uses the format Delta-Type (D-T), Length (L), Value (V). The SCHC Rule builds the description of each option as follows:

- * in the FID: an identifier of the option with option number built from the D-T;
- * in the FL: the option length, consistent with what is specified in Sections 7.4.1 and 7.4.2 of [RFC8724]; and

* in the TV: the option value.

When the Option Length has a well-known size, the Rule may keep the length value. Therefore, SCHC compression does not send it. Otherwise, SCHC compression carries the length of the Compression Residue, in addition to the Compression Residue value. Note that the length coding differs between CoAP options and SCHC variable size Compression Residue.

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

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

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

5.1. CoAP Option Content-Format and Accept Fields

If the client expects a single specific value, SCHC can elide these fields, by specifying the value in the TV of a Rule description with an "equal" MO and a "not-sent" CDA. Otherwise, if the client expects several possible values, a "match-mapping" MO SHOULD be used to limit the Compression Residue's size. If not, SCHC has to send the option value in the Compression Residue (with fixed or variable length).

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

SCHC compresses these three fields in the same way. When the values of these options are known, SCHC can elide these fields. If the option uses well-known values, SCHC can use a "match-mapping" MO. Otherwise, these options' values will be sent in the Compression Residue, i.e., the SCHC Rule description does not set the TV, while setting the MO to "ignore" and the CDA to "value-sent".

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

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

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

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

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

Table 1: Complex Path Example

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

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

For instance, for a CORECONF path /c/X6?k=eth0, the Rule description can be as shown in Table 2. That is, SCHC compresses the first part of the Uri-Path with a "not-sent" CDA. Also, SCHC will send the second element of the Uri-Path with the length (i.e., 0x2 "X6") followed by the query option with the length (i.e., 0x4 "eth0").

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

Table 2: CORECONF URI compression

5.3.1. Variable Number of Path or Query Elements

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

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

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

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

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

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

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

5.6. CoAP Option Hop-Limit Field

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

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

5.7. CoAP Option Echo Field

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

When a packet uses the Echo Option, SCHC compression SHOULD send its content in the Compression Residue. That is, in the SCHC Rule, the TV is not set, while the MO is set to "ignore" and the CDA is set to "value-sent". An exception applies in case the server generates the values to use for the Echo Option by means of a persistent counter (see Appendix A of [RFC9175]). In such a case, a Rule MAY use the MSB MO and the LSB CDA. This would be effectively applicable until the persistent counter at the server becomes greater than the maximum threshold value that produces an MSB-matching.

5.8. CoAP Option Request-Tag Field

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

protected, this also protects against interchange of fragments between different block-wise request operations.

When a packet uses the Request-Tag Option, SCHC compression MAY send its content in the Compression Residue. That is, in the SCHC Rule, the TV is not set, while the MO is set to "ignore" and the CDA is set to "value-sent". Alternatively, if a pre-defined set of Request-Tag values used by the client is known, a Rule MAY use a "match-mapping" MO when there are different options for the same FID.

5.9. CoAP Option EDHOC Field

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

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

6. Compression of CoAP Extensions

6.1. Block

When a packet uses a Block1 or Block2 Option [RFC7959] or a Q-Block1 or Q-Block2 Option [RFC9177], SCHC compression MUST send its content in the Compression Residue. In the SCHC Rule, the TV is not set, while the MO is set to "ignore" and the CDA is set to "value-sent". The Block1, Block2, Q-Block1, and Q-Block2 options allow fragmentation at the CoAP level that is compatible with SCHC fragmentation. Both fragmentation mechanisms are complementary, and the node may use them for the same packet as needed.

6.2. Observe

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

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

6.3. No-Response

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

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

6.4. OSCORE

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

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

- * As defined in Section 4.1 of [I-D.ietf-core-oscore-key-update], the eight least significant bit, when set, indicates that the OSCORE Option includes a second byte of flags. The seventh least significant bit is currently unassigned.
- * As defined in Section 5 of [I-D.ietf-core-oscore-groupcomm], the sixth least significant bit, when set, indicates that the message including the OSCORE option is protected with the group mode of Group OSCORE (see Section 8 of [I-D.ietf-core-oscore-groupcomm]). When not set, the bit indicates that the message is protected either with OSCORE, or with the pairwise mode of Group OSCORE (see Section 9 of [I-D.ietf-core-oscore-groupcomm]), while the specific OSCORE Security Context used to protect the message determines which of the two cases applies.

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

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

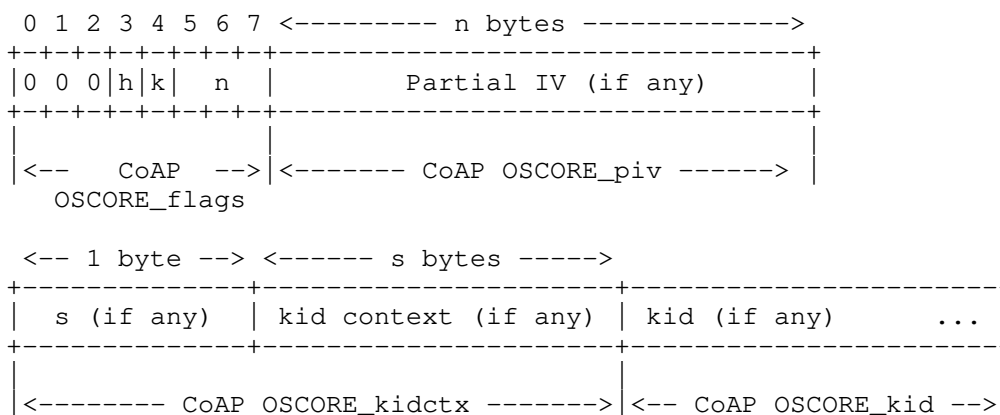


Figure 4: OSCORE Option

Figure 5 shows the extended OSCORE Option value encoding, with the second byte of flags also present. As defined in Section 4.1 of [I-D.ietf-core-oscore-key-update], the least significant bit d of this byte, when set, indicates that two additional fields are included in the option, following the kid context field (if any).

These two fields, namely x and nonce, are used when running the key update protocol KUDOS defined in [I-D.ietf-core-oscore-key-update], with x specifying the length of the nonce field in bytes as well as the specific behavior to adopt during the KUDOS execution.

If the seventh least significant bit z of the x field is set, it indicates that two additional fields are included in the option, following the x and nonce fields. These two fields, namely y and old_nonce, are also used when running the key update protocol KUDOS, with y specifying the length of the old_nonce field in bytes.

Figure 5 provides the breakdown of the x field, where its four least significant bits form the sub-field m, which specifies the size of nonce in bytes, minus 1. Also, the figure provides the breakdown of the y field, where its four least significant bits form the sub-field w, which specifies the size of old_nonce in bytes, minus 1.

To better perform OSCORE SCHC compression, the Rule description needs to identify the OSCORE Option and the fields it contains.

Conceptually, it discerns up to eight distinct pieces of information within the OSCORE Option: the flag bits, the piv, the kid context prepended by its size, the x byte, the nonce, the y byte, the old_nonce, and the kid. The SCHC Rule splits the OSCORE Option into eight corresponding Field Descriptors in order to compress those pieces of information:

- * CoAP OSCORE_flags
- * CoAP OSCORE_piv
- * CoAP OSCORE_kidctx
- * CoAP OSCORE_x
- * CoAP OSCORE_nonce
- * CoAP OSCORE_y
- * CoAP OSCORE_oldnonce
- * CoAP OSCORE_kid

Figure 4 shows the original format of the OSCORE Option with the four fields OSCORE_flags, OSCORE_piv, OSCORE_kidctx, and OSCORE_kid superimposed on it. Also, Figure 5 shows the extended format of the OSCORE option with all the eight fields superimposed on it.

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

In addition, the following applies.

- * For the x field, if both endpoints know the value, then the corresponding Field Descriptor in the SCHC Rule describes the TV set to that value, with the MO set to "equal" and the CDA set to "not-sent". This models: the case where the x field is not present, and thus TV is set to b''; and the case where the two endpoints run KUDOS with a pre-agreed size of the nonce field as per the m sub-field of the x field, as well as with a pre-agreed combination of its modes of operation, as per the bits b and p of the x field.

Otherwise, if the value changes over time, then the corresponding Field Descriptor in the SCHC Rule does not set the TV, while it sets the MO to "ignore" and the CDA to "value-sent". The Rule may also use a "match-mapping" MO to compress this field, in case the two endpoints pre-agree on a set of alternative ways to run KUDOS, with respect to the size of the nonce field and the combination of the KUDOS modes of operation to use.

- * For the y field, if both endpoints know the value, then the corresponding Field Descriptor in the SCHC Rule describes the TV set to that value, with the MO set to "equal" and the CDA set to "not-sent". This models: the case where the y field is not present, and thus TV is set to b''; and the case where the two endpoints run KUDOS with a pre-agreed size of the old_nonce field as per the w sub-field of the y field.

Otherwise, if the value changes over time, then the corresponding Field Descriptor in the SCHC Rule does not set the TV, while it sets the MO to "ignore" and the CDA to "value-sent". The Rule may also use a "match-mapping" MO to compress this field, in case the two endpoints pre-agree on a set of sizes of the old_nonce field.

- * For the nonce field, if it is not present (i.e., the x field is not present in the first place), then the corresponding Field Descriptor in the SCHC Rule describes the TV set to b'', with the MO set to "equal" and the CDA set to "not-sent".

Otherwise, if the nonce field is present, then the corresponding Field Descriptor in the SCHC Rule has the TV not set, while the MO is set to "ignore" and the CDA is set to "value-sent". In such a case, for the value of the nonce field, SCHC MUST NOT send it as variable-length data in the Compression Residue, to avoid ambiguity with the length of the nonce field encoded in the x field. Therefore, SCHC MUST use the m sub-field of the x field to define the size of the Compression Residue. SCHC designates a specific function, "osc.x.m", that the Rule MUST use to complete the Field Descriptor. During the decompression, this function returns the length of the nonce field in bytes, as the value of the m sub-field of the x field, plus 1.

- * For the old_nonce field, if it is not present (i.e., the y field is not present in the first place), then the corresponding Field Descriptor in the SCHC Rule describes the TV set to b'', with the MO set to "equal" and the CDA set to "not-sent".

Otherwise, if the old_nonce field is present, then the corresponding Field Descriptor in the SCHC Rule has the TV not set, while the MO is set to "ignore" and the CDA is set to "value-

sent". In such a case, for the value of the old_nonce field, SCHC MUST NOT send it as variable-length data in the Compression Residue, to avoid ambiguity with the length of the old_nonce field encoded in the y field. Therefore, SCHC MUST use the w sub-field of the y field to define the size of the Compression Residue. SCHC designates a specific function, "osc.y.w", that the Rule MUST use to complete the Field Descriptor. During the decompression, this function returns the length of the old_nonce field in bytes, as the value of the w sub-field of the y field, plus 1.

7. Compression of the CoAP Payload Marker

The following applies with respect to the 0xFF payload marker. A SCHC compression Rule for CoAP includes all the expected CoAP options, therefore the payload marker does not have to be specified in a SCHC Rule description.

If the CoAP message to compress with SCHC is not going to be protected with OSCORE [RFC8613] and includes a payload, then the 0xFF payload marker MUST NOT be included in the compressed message, which is composed of the Compression RuleID, the Compression Residue (if any), and the CoAP payload.

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

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

8. Examples of CoAP Header Compression

8.1. Mandatory Header with CON Message

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

```
+-----+
| RuleID 1 |
+-----+
```

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

Table 3: CoAP Context to compress header without Token

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

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

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

8.2. OSCORE Compression

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

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

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

- * Class E: Encrypted options moved to the Inner Plaintext.
- * Class I: Integrity-protected options, included in the Additional Authenticated Data (AAD) when protecting the Plaintext, but otherwise left untouched in the Outer Message.
- * Class U: Unprotected options, left untouched in the Outer Message.

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

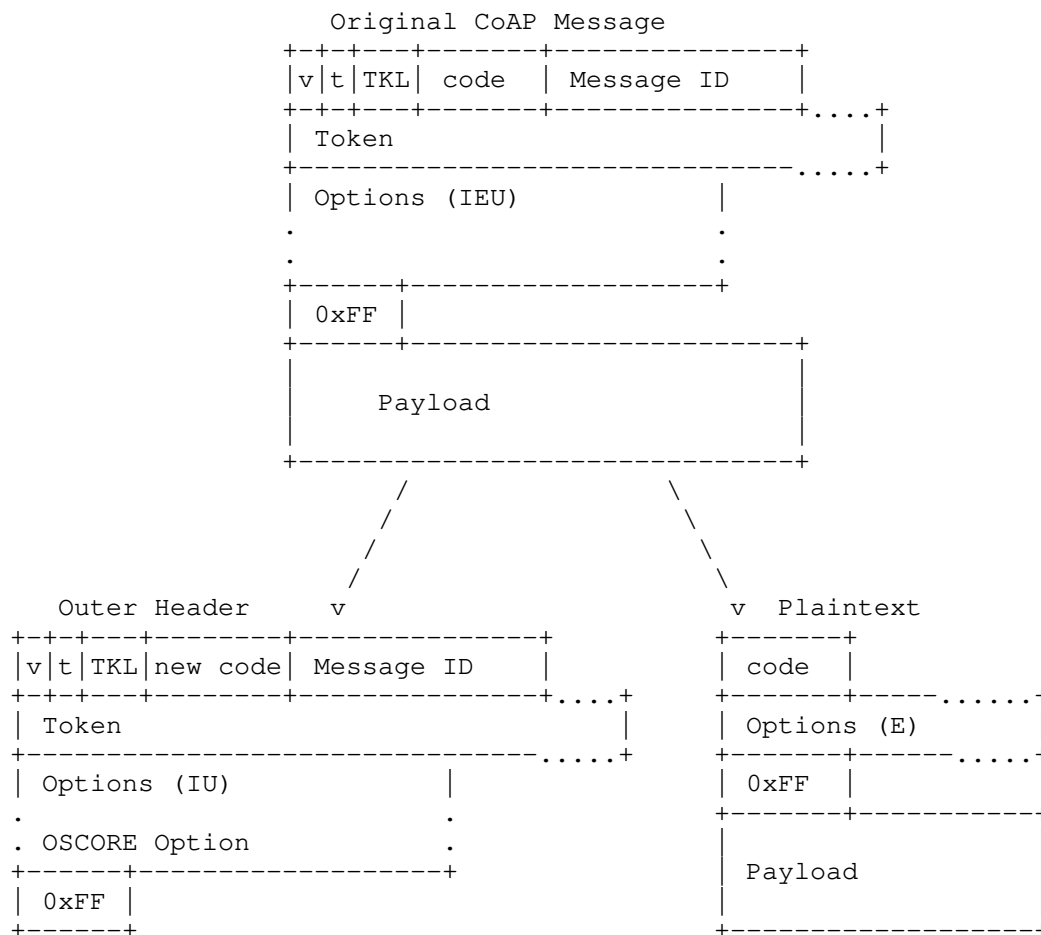


Figure 6: CoAP Packet Split into OSCORE Outer Header and Plaintext

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

In the Outer header, the original header code is hidden and replaced by a well-known value. As specified in Sections 4.1.3.5 and 4.2 of [RFC8613], the original header code is replaced with POST for requests and Changed for responses, when the message does not include the Observe Option. Otherwise, the original header code is replaced with FETCH for requests and Content for responses.

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

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

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

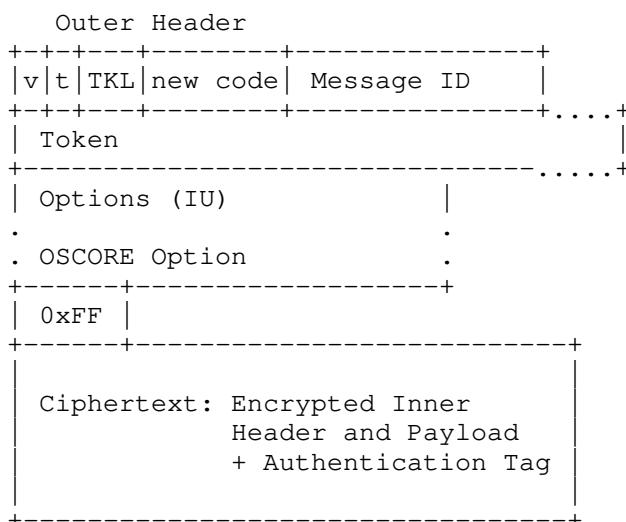


Figure 7: OSCORE Message

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

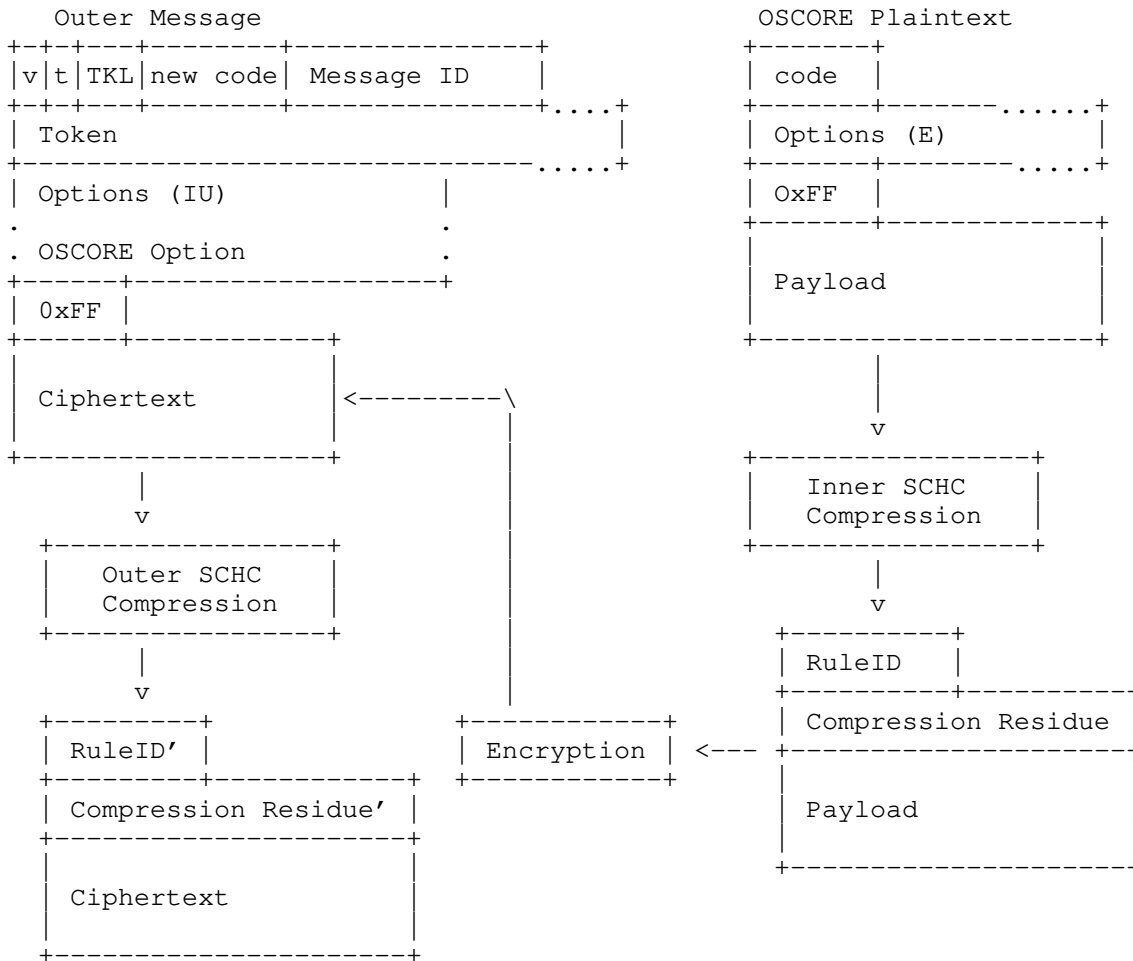


Figure 8: OSCORE Compression Diagram

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

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

- * After the Inner SCHC Compression of a CoAP message including a payload, the payload marker MUST NOT be included in the input to the AEAD Encryption, which is composed of the Inner Compression RuleID, the Inner Compression Residue (if any), and the CoAP payload.
- * The Outer SCHC Compression takes as input the OSCORE-protected message, which always includes a payload (i.e., the OSCORE Ciphertext) preceded by the payload marker.
- * After the Outer SCHC Compression, the payload marker MUST NOT be included in the final compressed message, which is composed of the Outer Compression RuleID, the Outer Compression Residue (if any), and the OSCORE Ciphertext.

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

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

8.3. Example OSCORE Compression

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

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

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

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

0x0001 = mid
0x82 = token

Options:

0xbb74656d7065726174757265
Option 11: Uri_Path
Value = temperature

Original message length: 17 bytes
```

Figure 9: CoAP GET Request

Its corresponding response is the Content response in Figure 10.

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

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

0x0001 = mid
0x82 = token

0xFF  Payload marker

Payload:
0x32332043

Original message length: 10 bytes
```

Figure 10: CoAP Content Response

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

```

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

```

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

Table 4: Inner SCHC Rule

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

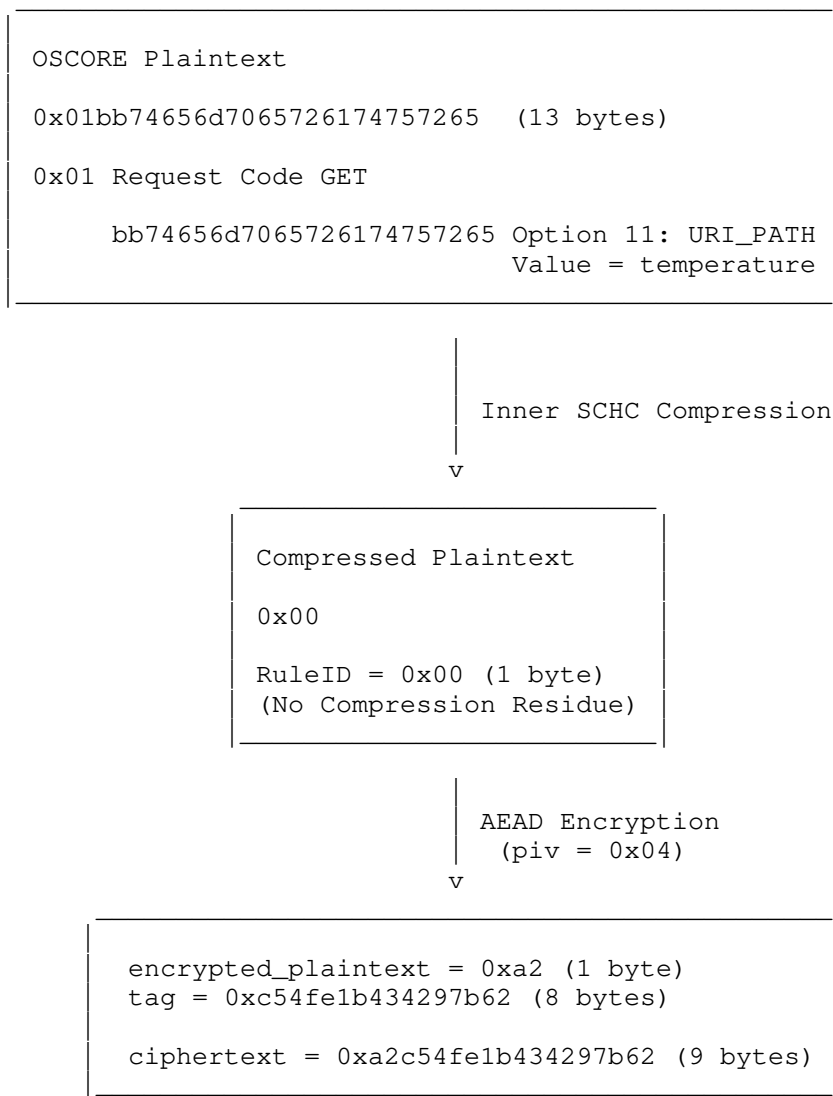


Figure 11: Plaintext Compression and Encryption for GET Request

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

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

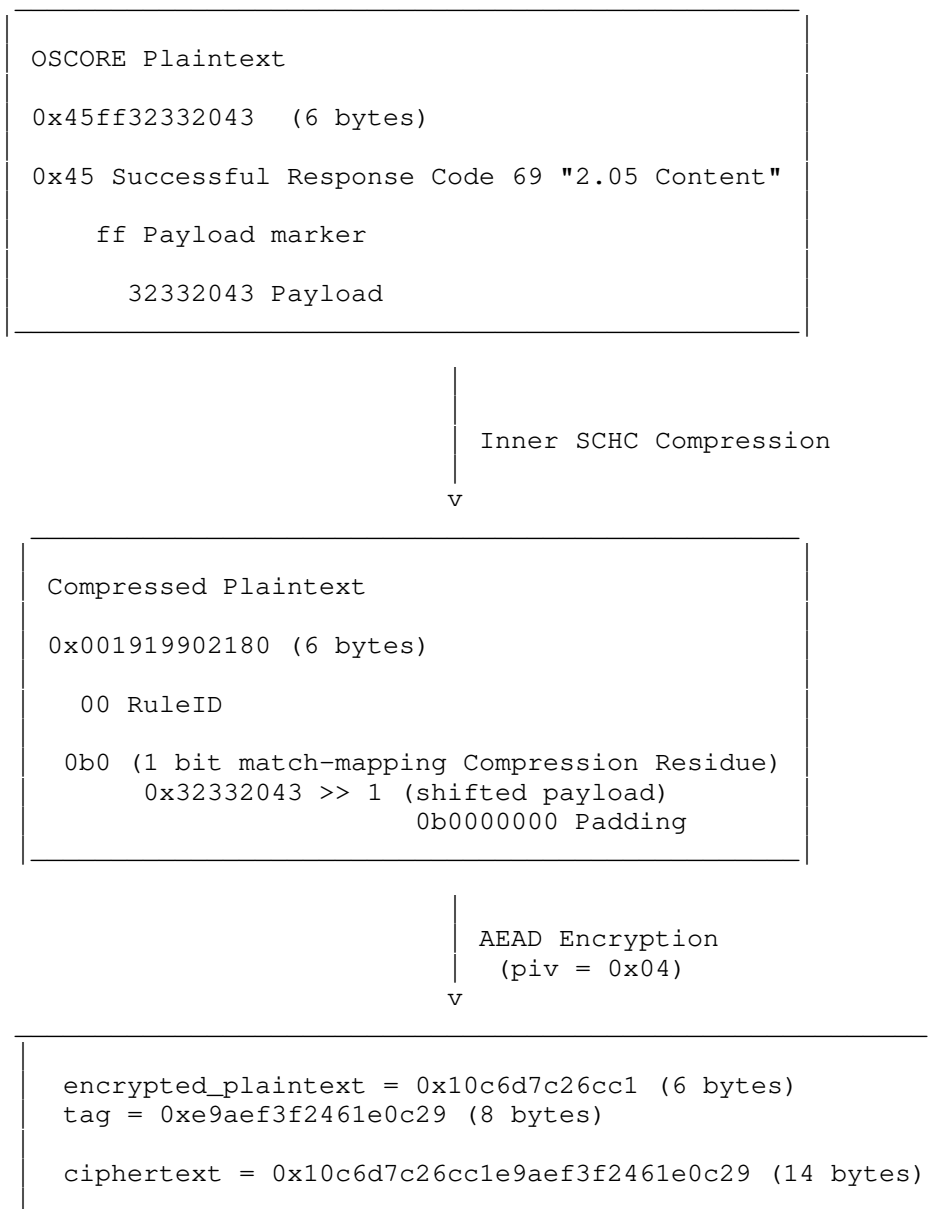


Figure 12: Plaintext Compression and Encryption for Content Response

The Outer SCHC Rule shown in Table 5 is used, also to process the OSCORE Option fields. Figure 13 and Figure 14 show a dump of the OSCORE messages generated from the example messages, also including

the Inner Compressed ciphertext in the payload. These are the messages that have to be compressed via the Outer SCHC Compression scheme.

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

```
+-----+
| RuleID 1 |
+-----+
```

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP Version	2	1	Bi	01	equal	not-sent	
CoAP Type	2	1	Up	0	equal	not-sent	
CoAP Type	2	1	Dw	2	equal	not-sent	
CoAP TKL	4	1	Bi	1	equal	not-sent	
CoAP Code	8	1	Up	2	equal	not-sent	
CoAP Code	8	1	Dw	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	var	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 (44)	LSB	KKKK

CoAP OSCORE_kidctx	var	1	Bi	b''	equal	not- sent
CoAP OSCORE_x	8	1	Bi	b''	equal	not- sent
CoAP OSCORE_nonce	osc.x.m	1	Bi	b''	equal	not- sent
CoAP OSCORE_y	8	1	Bi	b''	equal	not- sent
CoAP OSCORE_oldnonce	osc.y.w	1	Bi	b''	equal	not- sent
CoAP OSCORE_flags	var	1	Dw	b''	equal	not- sent
CoAP OSCORE_piv	var	1	Dw	b''	equal	not- sent
CoAP OSCORE_kid	var	1	Dw	b''	equal	not- sent

Table 5: Outer SCHC Rule

Protected message:

=====

0x4102000182980904636c69656e74ffa2c54fe1b434297b62
(24 bytes)

Header:

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

0x0001 = mid

0x82 = token

Options:

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

0xFF Payload marker

Payload:

0xa2c54fe1b434297b62 (9 bytes)

Figure 13: Protected and Inner SCHC Compressed GET Request

Protected message:

=====

0x614400018290ff10c6d7c26cc1e9aef3f2461e0c29

(21 bytes)

Header:

0x6144

01 Ver

10 ACK

0001 TKL

01000100 Successful Response Code 68 "2.04 Changed"

0x0001 = mid

0x82 = token

Options:

0x90 (1 byte)

Option 9: OSCORE

Value = b''

0xFF Payload marker

Payload:

0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)

Figure 14: Protected and Inner SCHC Compressed Content Response

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

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

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

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

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

Compressed message:

```
=====
0x011489458a9fc3686852f6c4 (12 bytes)
0x01 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 15: SCHC-OSCORE Compressed GET Request

Compressed message:

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

Compression Residue:

```
0b0001 010 (7 bits -> 1 byte with padding)
    mid tkn
```

Payload

```
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)
```

Figure 16: SCHC-OSCORE Compressed Content Response

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

```

+-----+
| RuleID 2 |
+-----+

```

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP Version	2	1	Bi	01	equal	not-sent	
CoAP Type	2	1	Up	0	equal	not-sent	
CoAP Type	2	1	Dw	2	equal	not-sent	
CoAP TKL	4	1	Bi	1	equal	not-sent	
CoAP Code	8	1	Up	2	equal	not-sent	
CoAP Code	8	1	Dw	[69,132]	match-mapping	mapping-sent	C
CoAP MID	16	1	Bi	0000	MSB(12)	LSB	MMMM
CoAP Token	tkl	1	Bi	0x80	MSB(5)	LSB	TTT
CoAP Uri-Path		1	Up	"temperature"	equal	not-sent	

Table 6: SCHC-CoAP Rule (No OSCORE)

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

Compressed message:

=====

0x0214

0x02 = RuleID

Compression Residue:

0b00010100 (1 byte)

Compressed message length: 2 bytes

Figure 17: CoAP GET Compressed without OSCORE

Compressed message:

=====

0x020a32332043

0x02 = RuleID

Compression Residue:

0b00001010 (1 byte)

Payload

0x32332043

Compressed message length: 6 bytes

Figure 18: CoAP Content Compressed without OSCORE

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

9. CoAP Header Compression with Proxies

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

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

9.1. Without End-to-End Security

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

In particular, SCHC is used as defined below.

- * The sender application endpoint compresses the CoAP message, by using the SCHC Rules that it shares with the next hop towards the recipient application endpoint. The resulting, compressed message is sent to the next hop towards the recipient application endpoint.
- * Each proxy decompresses the incoming compressed message, by using the SCHC Rules that it shares with the (previous hop towards the) sender application endpoint.

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

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

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

9.2. With End-to-End Security

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

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

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

In particular, SCHC is used as defined below.

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

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

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

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

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

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

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

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

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

10. Examples of CoAP Header Compression with Proxies

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

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

The following also applies in the presented examples.

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

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

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

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

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

In particular, note that:

- * On the communication leg between the Device and the proxy, the CoAP Message ID has value 0x0001 and the CoAP Token has value 0x82.
- * On the communication leg between the proxy and the Application Server, the CoAP Message ID has value 0x0004 and the CoAP Token has value 0x75.

Original message:

=====

0x41010001823b6578616d706c652e636f6d
8b74656d7065726174757265d40f636f6170

Header:

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

0x0001 = mid

0x82 = token

Options:

0x3b6578616d706c652e636f6d

Option 3: Uri-Host

Value = example.com

0x8b74656d7065726174757265

Option 11: Uri-Path

Value = temperature

0xd40f636f6170

Option 39: Proxy-Scheme

Value = coap

Original message length: 35 bytes

Figure 19: CoAP GET Request

Original message:

```
=====
0x6145000475ff32332043
```

Header:

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

0x0004 = mid

0x75 = token

0xFF Payload marker

Payload:

```
0x32332043
```

Original message length: 10 bytes

Figure 20: CoAP Content Response

10.1. Without End-to-End Security

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

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

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

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP Version	2	1	Bi	01	equal	not-sent	
CoAP Type	2	1	Up	0	equal	not-sent	
CoAP Type	2	1	Dw	[0, 2]	match-mapping	mapping-sent	T
CoAP TKL	4	1	Bi	1	equal	not-sent	
CoAP Code	8	1	Up	[1, 2, 3, 4]	match-mapping	mapping-sent	CC
CoAP Code	8	1	Dw	[65, 68, 69, 132]	match-mapping	mapping-sent	CC
CoAP MID	16	1	Bi	0x00	MSB(12)	LSB	MMMM
CoAP Token	tkl	1	Bi	0x80	MSB(5)	LSB	TTT
CoAP Uri-Host	var (B)	1	Up		ignore	value-sent	
CoAP Uri-Path	var	1	Up	"temperature"	equal	not-sent	
CoAP Proxy-Scheme	var	1	Up	"coap"	equal	not-sent	

Table 7: SCHC Rule between the Device and the Proxy

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

RuleID 1

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP Version	2	1	Bi	01	equal	not-sent	
CoAP Type	2	1	Up	0	equal	not-sent	
CoAP Type	2	1	Dw	[0, 2]	match- mapping	mapping- sent	T
CoAP TKL	4	1	Bi	1	equal	not-sent	
CoAP Code	8	1	Up	[1, 2, 3, 4]	match- mapping	mapping- sent	CC
CoAP Code	8	1	Dw	[65, 68, 69, 132]	match- mapping	mapping- sent	CC
CoAP MID	16	1	Bi	0x00	MSB(12)	LSB	MMMM
CoAP Token	tkl	1	Bi	0x70	MSB(5)	LSB	TTT
CoAP Uri-Host	var (B)	1	Up		ignore	value- sent	
CoAP Uri-Path	var	1	Up	"temperature"	equal	not-sent	

Table 8: SCHC Rule between the Proxy and the Application Server

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

Compressed message:

=====

0x00055b2bc30b6b836329731b7b68 (14 bytes)

0x00 RuleID

055b2bc30b6b836329731b7b68 compression residue
and padded payload

Compression Residue (101 bits -> 13 bytes with padding)

0b 00 0001 010 1011 | 0x6578616d706c652e636f6d
code mid tkn Uri-Host (residue size and residue)

Compressed message length: 14 bytes

Figure 21: CoAP GET Request Compressed for the Proxy

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

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

Message to forward:

```
=====
0x41010004753b6578616d706c652e636f6d
  8b74656d7065726174757265
```

Header:

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

0x0004 = mid

0x75 = token

Options:

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

```
0x8b74656d7065726174757265
```

```
Option 11: Uri-Path
```

```
Value = temperature
```

Original message length: 29 bytes

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

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

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

Compressed message to forward:

```
=====
0x0112db2bc30b6b836329731b7b68 (14 bytes)
0x01 RuleID
  12db2bc30b6b836329731b7b68 compression residue
                                and padded payload
```

Compression Residue (101 bits -> 13 bytes with padding)

```
0b 00 0100 101 1011 | 0x6578616d706c652e636f6d
   code mid tkn Uri-Host (residue size and residue)
```

Compressed message length: 14 bytes

Figure 23: CoAP GET Request Forwarded by the Proxy

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

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

Compressed message:

=====

0x01c94c8cc810c0 (7 bytes)

0x01 RuleID

c94c8cc810c0 compression residue
and padded payload

Compression Residue (10 bits -> 2 bytes with padding)

0b 1 10 0100 101
type code mid tkn

Payload

0x32332043 (4 bytes)

Compressed message length: 7 bytes

Figure 24: CoAP Content Response Compressed for the Proxy

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

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

Message to forward:

```
=====
0x6145000182ff32332043
```

Header:

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

0x0001 = mid

0x82 = token

0xFF Payload marker

Payload:

```
0x32332043
```

Original message length: 10 bytes

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

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

Compressed message:

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

Compression Residue (10 bits -> 2 bytes with padding)

```
0b 1 10 0001 010
   type code mid tkn
```

Payload

```
0x32332043 (4 bytes)
```

Compressed message length: 7 bytes

Figure 26: CoAP Content Response Forwarded by the Proxy

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

10.2. With End-to-End Security

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

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

```
+-----+
| RuleID 2 |
+-----+
```

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP Code	8	1	Up	[1, 2, 3, 4]	match-mapping	mapping-sent	CC
CoAP Code	8	1	Dw	[65, 68, 69, 132]	match-mapping	mapping-sent	CC
CoAP Uri-Path	var	1	Up	"temperature"	equal	not-sent	

Table 9: Inner SCHC Rule between the Device and the Application Server

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

```
+-----+
| RuleID 3 |
+-----+
```

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP Version	2	1	Bi	01	equal	not-sent	
CoAP Type	2	1	Up	0	equal	not-sent	
CoAP Type	2	1	Dw	[0, 2]	match-mapping	mapping-sent	T
CoAP TKL	4	1	Bi	1	equal	not-sent	
CoAP Code	8	1	Up	2	equal	not-sent	
CoAP Code	8	1	Dw	68	equal	not-sent	
CoAP MID	16	1	Bi	0x00	MSB (12)	LSB	MMMM
CoAP Token	tkl	1	Bi	0x80	MSB (5)	LSB	TTT
CoAP Uri-Host	var (B)	1	Up		ignore	value-sent	
CoAP OSCORE_flags	var	1	Up	0x09	equal	not-sent	
CoAP OSCORE_piv	var	1	Up	0x00	MSB (4)	LSB	PPPP
CoAP OSCORE_kid	var	1	Up	0x0000	MSB (12)	LSB	KKKK
CoAP OSCORE_kidctx	var	1	Bi	b''	equal	not-sent	
CoAP OSCORE_x	8	1	Bi	b''	equal	not-sent	
CoAP OSCORE_nonce	osc.x.m	1	Bi	b''	equal	not-sent	

CoAP OSCORE_y	8	1	Bi	b''	equal	not-sent	
CoAP OSCORE_oldnonce	osc.y.w	1	Bi	b''	equal	not-sent	
CoAP OSCORE_flags	var	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	
CoAP Proxy-Scheme	var	1	Up	"coap"	equal	not-sent	

Table 10: Outer SCHC Rule between the Device and the Proxy

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

```
+-----+
| RuleID 4 |
+-----+
```

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
CoAP Version	2	1	Bi	01	equal	not-sent	
CoAP Type	2	1	Up	0	equal	not-sent	
CoAP Type	2	1	Dw	[0, 2]	match- mapping	mapping- sent	T
CoAP TKL	4	1	Bi	1	equal	not-sent	
CoAP	8	1	Up	2	equal	not-sent	

Code								
CoAP Code	8	1	Dw	68	equal	not-sent		
CoAP MID	16	1	Bi	0x00	MSB (12)	LSB	MMMM	
CoAP Token	tkl	1	Bi	0x70	MSB (5)	LSB	TTT	
CoAP Uri-Host	var (B)	1	Up		ignore	value-sent		
CoAP OSCORE_flags	var	1	Up	0x09	equal	not-sent		
CoAP OSCORE_piv	var	1	Up	0x00	MSB (4)	LSB	PPPP	
CoAP OSCORE_kid	var	1	Up	0x0000	MSB (12)	LSB	KKKK	
CoAP OSCORE_kidctx	var	1	Bi	b''	equal	not-sent		
CoAP OSCORE_x	8	1	Bi	b''	equal	not-sent		
CoAP OSCORE_nonce	osc.x.m	1	Bi	b''	equal	not-sent		
CoAP OSCORE_y	8	1	Bi	b''	equal	not-sent		
CoAP OSCORE_oldnonce	osc.y.w	1	Bi	b''	equal	not-sent		
CoAP OSCORE_flags	var	1	Dw	b''	equal	not-sent		
CoAP OSCORE_piv	var	1	Dw	b''	equal	not-sent		
CoAP OSCORE_kid	var	1	Dw	b''	equal	not-sent		

Table 11: Outer SCHC Rule between the Proxy and the Application Server

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

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

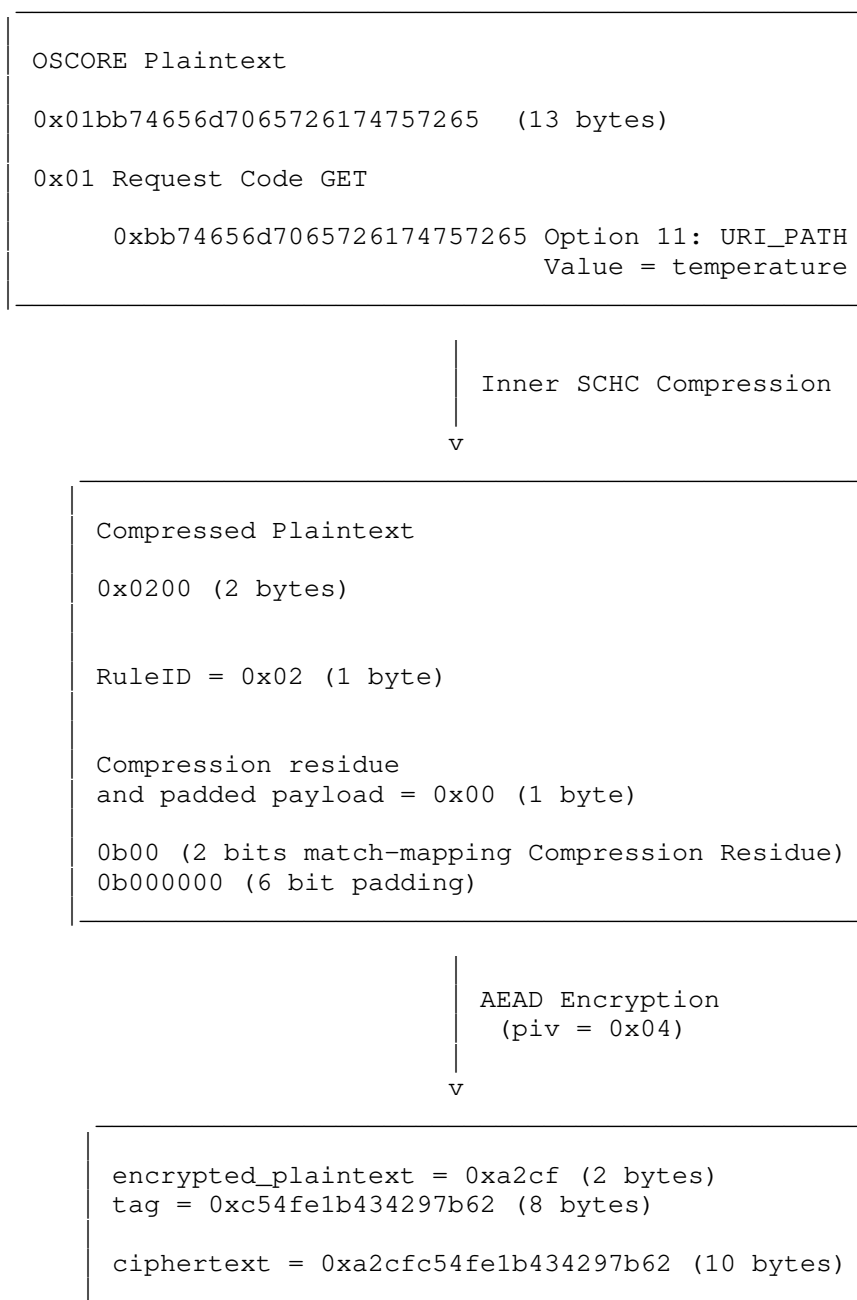


Figure 27: Plaintext Compression and Encryption for the GET Request

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

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

```
OSCORE Plaintext  
  
0x45ff32332043 (6 bytes)  
  
0x45 Successful Response Code 69 "2.05 Content"  
  
  0xff Payload marker  
  
    0x32332043 Payload
```

| Inner SCHC Compression
v

```
Compressed Plaintext  
  
0x028c8cc810c0 (6 bytes)  
  
RuleID = 0x02  
  
Compression residue  
and padded payload = 0x8c8cc810c0 (5 bytes)  
  
0b10 (2 bits match-mapping Compression Residue)  
  0x32332043 >> 2 (shifted payload)  
    0b000000 Padding
```

| AEAD Encryption
 (piv = 0x04)
v

```
encrypted_plaintext = 0x10c6d7c26cc1 (6 bytes)  
tag = 0xe9aef3f2461e0c29 (8 bytes)  
  
ciphertext = 0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)
```

Figure 28: Plaintext Compression and Encryption for the Content Response

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

Protected message:

=====

0x41020001823b6578616d706c652e636f6d
6409040005d411636f6170ffa2cfc54fe1b434297b62
(39 bytes)

Header:

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

0x0001 = mid
0x82 = token

Options:

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

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

0xd411636f6170
Option 39: Proxy-Scheme
Value = coap

0xFF Payload marker

Payload:

0xa2cfc54fe1b434297b62 (10 bytes)

Figure 29: Protected and Inner SCHC Compressed CoAP GET Request

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

Compressed message:

=====

0x03156caf0c2dae0d8ca5cc6deda8b459f8a9fc3686852f6c40 (25 bytes)

0x03 RuleID

156caf0c2dae0d8ca5cc6deda8b459f8a9fc3686852f6c40 compression
residue and
padded payload

Compression Residue (107 bits -> 14 bytes with padding)

0b 0001 010 1011 | 0x6578616d706c652e636f6d | 0b 0100 0101
mid tkn Uri-Host (residue size and residue) piv kid

Payload

0xa2cfc54felb434297b62 (10 bytes)

Compressed message length: 25 bytes

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

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

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

Protected message:

```
=====
0x41020004753b6578616d706c652e636f6d
 6409040005ffa2cfc54fe1b434297b62
(33 bytes)
```

Header:

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

```
0x0004 = mid
0x75 = token
```

Options:

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

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

0xFF Payload marker

Payload:

```
0xa2cfc54fe1b434297b62 (10 bytes)
```

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

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

Compressed message:

=====

```
0x044b6caf0c2dae0d8ca5cc6deda8b459f8a9fc3686852f6c40 (25 bytes)
0x04 RuleID
    4b6caf0c2dae0d8ca5cc6deda8b459f8a9fc3686852f6c40 compression
                                                residue and
                                                padded payload
```

Compression Residue (107 bits -> 14 bytes with padding)

```
0b 0100 101    1011 | 0x6578616d706c652e636f6d | 0b 0100 0101
    mid tkn  Uri-Host  (residue size and residue)          piv  kid
```

Payload

```
0xa2cfc54fe1b434297b62 (10 bytes)
```

Compressed message length: 25 bytes

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

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

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

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

Protected message:

=====

0x614400047590ff10c6d7c26cc1e9aef3f2461e0c29

(21 bytes)

Header:

0x6144

01 Ver

10 ACK

0001 TKL

01000100 Successful Response Code 68 "2.04 Changed"

0x0004 = mid

0x75 = token

Options:

0x90

Option 9: OSCORE

Value = b''

0xFF Payload marker

Payload:

0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)

Figure 33: Protected and Inner SCHC Compressed CoAP Content Response

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

Compressed message:

=====

```
0x04a510c6d7c26cc1e9aef3f2461e0c29 (16 bytes)
0x04 RuleID
    a510c6d7c26cc1e9aef3f2461e0c29 compression residue
                                   and padded payload
```

Compression Residue (8 bits -> 1 byte with padding)

```
0b    1 0100 101
    type mid tkn
```

Payload

```
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)
```

Compressed message length: 16 bytes

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

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

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

Protected message:

=====

0x614400018290ff10c6d7c26cc1e9aef3f2461e0c29

(21 bytes)

Header:

0x6144

01 Ver

10 ACK

0001 TKL

01000100 Successful Response Code 68 "2.04 Changed"

0x0001 = mid

0x82 = token

Options:

0x90

Option 9: OSCORE

Value = b''

0xFF Payload marker

Payload:

0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)

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

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

Compressed message:

=====

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

Compression Residue (8 bits -> 1 byte with padding)

```
0b    1 0001 010
    type mid tkn
```

Payload

```
0x10c6d7c26cc1e9aef3f2461e0c29 (14 bytes)
```

Compressed message length: 16 bytes

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

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

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

11. Security Considerations

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

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

When SCHC is used with OSCORE, the security considerations discussed in [RFC8613] continue to apply. When SCHC is used with Group OSCORE, the security considerations discussed in [I-D.ietf-core-oscore-groupcomm] apply. When SCHC is used in the presence of CoAP proxies, the security considerations discussed in Section 11.2 of [RFC7252] continue to apply.

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

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

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

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

12. IANA Considerations

This document has no actions for IANA.

13. References

13.1. Normative References

[I-D.ietf-core-oscore-edhoc]

Palombini, F., Tiloca, M., Höglund, R., Hristozov, S., and G. Selander, "Using EDHOC with CoAP and OSCORE", Work in Progress, Internet-Draft, draft-ietf-core-oscore-edhoc-09, 13 October 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-edhoc-09>>.

[I-D.ietf-core-oscore-groupcomm]

Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and J. Park, "Group Object Security for Constrained RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-20, 2 September 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-20>>.

[I-D.ietf-core-oscore-key-update]

Höglund, R. and M. Tiloca, "Key Update for OSCORE (KUDOS)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-key-update-06, 23 October 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-key-update-06>>.

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

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

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

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

- [RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <<https://www.rfc-editor.org/rfc/rfc7959>>.
- [RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <<https://www.rfc-editor.org/rfc/rfc7967>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/rfc/rfc8174>>.
- [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/rfc/rfc8613>>.
- [RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <<https://www.rfc-editor.org/rfc/rfc8724>>.
- [RFC8768] Boucadair, M., Reddy, K. T., and J. Shallow, "Constrained Application Protocol (CoAP) Hop-Limit Option", RFC 8768, DOI 10.17487/RFC8768, March 2020, <<https://www.rfc-editor.org/rfc/rfc8768>>.
- [RFC9175] Amsüss, C., Preuß, J., and G. Selander, "Constrained Application Protocol (CoAP): Echo, Request-Tag, and Token Processing", RFC 9175, DOI 10.17487/RFC9175, February 2022, <<https://www.rfc-editor.org/rfc/rfc9175>>.
- [RFC9177] Boucadair, M. and J. Shallow, "Constrained Application Protocol (CoAP) Block-Wise Transfer Options Supporting Robust Transmission", RFC 9177, DOI 10.17487/RFC9177, March 2022, <<https://www.rfc-editor.org/rfc/rfc9177>>.

13.2. Informative References

[I-D.ietf-core-groupcomm-bis]

Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-09, 10 July 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-09>>.

[I-D.ietf-lake-edhoc]

Selander, G., Mattsson, J. P., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ietf-lake-edhoc-22, 25 August 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-lake-edhoc-22>>.

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

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

Appendix A. YANG Data Model

TBD

Acknowledgments

The authors sincerely thank Christian Amsüss, Quentin Lampin, John Preuß Mattsson, Carles Gomez Montenegro, Göran Selander, Pascal Thubert, and Éric Vyncke for their comments and feedback.

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

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden
Email: marco.tiloca@ri.se

Laurent Toutain
IMT Atlantique
CS 17607, 2 rue de la Chataigneraie
35576 Cesson-Sevigne Cedex
France
Email: Laurent.Toutain@imt-atlantique.fr

Ivan Martinez
Nokia Bell Labs
12 Rue Jean Bart
91300 Massy
France
Email: ivan.martinez_bolivar@nokia-bell-labs.com

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

SCHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 24 April 2024

A. Minaburo
Consultant
L. Toutain
Institut MINES TELECOM; IMT Atlantique
I. Martinez
Nokia Bell Labs
22 October 2023

SCHC Rule Access Control
draft-toutain-schc-access-control-03

Abstract

The framework for SCHC defines an abstract view of the rules, formalized through a YANG Data Model. In its original description, rules are static and shared by two endpoints. This document defines augmentation to the existing Data Model in order to restrict the changes in the rule and, therefore, the impact of possible attacks.

Status of This Memo

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

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

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

This Internet-Draft will expire on 24 April 2024.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

- 1. Introduction 2
- 2. Conventions and Definitions 2
- 3. Terminology 2
- 4. SCHC TV/MO/CDA possible combinations 3
- 5. YANG Access Control 3
- 6. YANG Data Model 4
 - 6.1. leaf ac-modify-set-of-rules 4
 - 6.2. leaf ac-modify-compression-rule 5
 - 6.3. leaf ac-modify-field 5
 - 6.3.1. CoAP base header Access Control. 5
 - 6.3.2. CoAP Options Access Control. 6
- 7. Normative References 8
- Appendix A. YANG Data Model 10
- Appendix B. Security Considerations 13
- Appendix C. IANA Considerations 13
- Authors' Addresses 13

1. Introduction

SCHC is a compression and fragmentation mechanism defined in [RFC8724] while [RFC9363] provides a YANG Data Model for formal representation of SCHC Rules used either for compression/decompression (C/D) or fragmentation/reassembly (F/R). The inappropriate changes to SCHC Rules leads to some possible attacks. The goal of this document is to define a augmentation to the existing Data Model in order to restrict the changes in the rules and, therefore, the impact of possible attacks.

2. Conventions and Definitions

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.

3. Terminology

It is expected that the reader will be familiar with the terms and concepts associated with the SCHC framework [RFC8724], [I-D.ietf-schc-architecture], and managment request processing [I-D.ietf-core-comi], NETCONF[RFC6241], RESTCONF [RFC8040].

ToDo * Access Control. * Management request processing: The NETCONF, RESTCONF or CORECONF request is processed and passed to the end-point Rule Manager. * Rule Manager (RM). * Context. SCHC Rules

4. SCHC TV/MO/CDA possible combinations

SCHC compression behavior uses the TV, MO, and CDA to generate the correct residue. But not all the combinations of this fields descriptors are possible, and then an attack can be detected or avoided. Figure 1 shows all the combinations and those that are enabled. SCHC defines two TV values: set and not set. SCHC MO can be Equal, Ignore, MSB, or Match-mapping. And SCHC CDA can be not-sent, value-sent, mapping-sent, LSB, compute-*, DevIID, or AppIID.

TV / MO	CDA						
	not-sent	value-sent	mapping-sent	LSB	compute-*	DevIID	AppIID
set / Equal	ok	absurd	x	x	absurd	absurd	absurd
not set / Equal	x	x	x	x	absurd	absurd	absurd
set / Ignore	ok (D)	absurd	x	x	ok	ok	ok
not set / Ignore	x	ok	x	x	ok	ok	ok
set / MSB	absurd	absurd	x	ok	absurd	absurd	absurd
not set / MSB	absurd	absurd	x	ok	absurd	absurd	absurd
set / Match-mapping	x	absurd	ok	x	absurd	absurd	absurd
not set / Match-mapping	x	x	absurd	x	absurd	absurd	absurd

Figure 1: SCHC TV, MO, CDA valid combinations

5. YANG Access Control

YANG language allows to specify read-only or read write nodes. NACM [RFC8341] extends this by allowing users or groups of users to perform specific actions.

This granularity does not fit this rule model. For instance, the goal is not to allow all the field-id leaves to be modified. The objective is to allow a specific rule entry to be changed and, therefore, some of the leaves to be modified. For instance, an entry with FID containing Uri-path may have its target value modified, as in the same rule, the entry regarding the application prefix should not be changed.

The SCHC access control augments the YANG module defined in [RFC9363] to allow a remote entity to manipulate the rules. Several levels are defined.

- * in the set of rules, it authorizes or not a new rule to be added.
- * in a compression rule, it allows adding or removing field descriptions.
- * in a compression rule, it allows modifying some elements of the rule, such as the TV, MO, or/and CDA, and associated values.
- * in a fragmentation rule, it allows modifying some parameters.

6. YANG Data Model

The YANG DM proposed in Appendix A extends the SCHC YANG Data Model introduced in [RFC9363]. It adds read-only leaves containing access rights. If these leaves are not present, the information cannot be modified.

6.1. leaf ac-modify-set-of-rules

This leaf controls modifications applied to a set of rules. They are specified with the rule-access-right enumeration:

- * no-change (0): rules cannot be modified in the Set of Rules. This is the equivalent of having no access control elements in the set of rules.
- * modify-existing-element (1): an existing rule may be modified.
- * add-remove-element (2): a rule can be added or deleted from the Set of Rules, or an existing rule can be modified.

6.2. leaf ac-modify-compression-rule

This leaf allows to modify a compression element. To be active, leaf ac-modify-set-of-rules MUST be set to modify-existing-element or add-remove-element. This leaf uses the same enumeration as add-remove-element:

- * no-change (0): The rule cannot be modified.
- * modify-existing-element (1): an existing Field Description may be modified.
- * add-remove-element (2): a Field Description can be added or deleted from the Rule or an existing rule can be modified.

6.3. leaf ac-modify-field

This leaf allows to modify a Field Description in a compression rule. To be active, leaves ac-modify-set-of-rules and ac-modify-compression-rule MUST be set to modify-existing-element or add-remove-element and ac-modify-compression-rule and leaf

6.3.1. CoAP base header Access Control.

CoAP protocol uses a request/reply model with compact messages. The format of these messages starts with a fixed format of 4-byte length, followed by a variable Token format with a length between 0 and 8 bytes. While applying SCHC header compression [RFC8824], the based header is only readable and MUST not be modified. Figure 2 shows the access-control for the FID and TV in a Rules.

Access Control FID	FID	FL	FP	DI	Access Control TV	TV (default value)
Read Only	CoAP.Version	2	1	Bi	Read Only	1
Read Only	CoAP.Type	2	1	Bi	Read Only	CON, NON-C, ACK, RST.
Read Only	CoAP.TKL	4	1	Bi	Read/Write	none
Read Only	CoAP.Code	8	1	Bi	Read Only	See CoAP Code Registries
Read Only	CoAP.MessageID	16	1	Bi	Read/Write	none
Read Only	CoAP.Token	0-8	1	Bi	Read/Write	none

Figure 2: Access Control for the CoAP Header

6.3.2. CoAP Options Access Control.

The CoAP options are used by both request and responses messages. Some of them are defined as repeatable which implies that it MAY be included one or more times in a message. In this case, a SCHC Rule MAY be able to modify the FID and the TV in order to include the repetition. The only FID's that have access to be modifiable are those that have been defined as repeatable. The Figure 3 give the control access for all the CoAP Options defined in [RFC7252]; [RFC8613]; [RFC8768]; [RFC9177]; [RFC7959]; and [RFC9175].

Access Control FID	CoAP Opt. Num.	FID	FL	FP	DI	Access Control TV	TV (default value)
Read/Write	1	CoAP.Option.If-Match	0-8	var	Bi	Read/Write	none
Read Only	3	CoAP.Option.Uri-Host	1-255	var	Bi	Read/Write	Sect. 5 RFC7252
Read/Write	4	CoAP.Option.ETag	1-8	var	Bi	Read/Write	none

Read Only	5	CoAP.Option.If-None-Match	0	1	Bi	Read Only	empty
Read Only	6	CoAP.Option.Observe	0-3	var	Bi	Read Only	none
Read Only	7	CoAP.Option.Uri-Port	0-2	var	Bi	Read Only	Sect. 5 RFC7252
Read/Write	8	CoAP.Option.Location-Path	0-255	var	Bi	Read/Write	none
Read Only	9	CoAP.Option.OSCORE	0-255	var	Bi	Read Only	none
Read/Write	11	CoAP.Option.Uri-Path	0-255	var	Bi	Read/Write	none
Read Only	12	CoAP.Option.Content-Format	0-2	var	Bi	Read Only	none
Read Only	14	CoAP.Option.Max-Age	0-4	var	Bi	Read Only	60
Read/Write	15	CoAP.Option.Uri-Query	0-255	var	Bi	Read/Write	none
Read Only	16	CoAP.Option.Hop-Limit	1	1	Bi	Read Only	16
Read Only	17	CoAP.Option.Accept	0-2	var	Bi	Read Only	none
Read Only	19	CoAP.Option.Q-Block1	0-3	var	Bi	Read Only	none
Read/Write	20	CoAP.Option.Location-Query	0-255	var	Bi	Read/Write	none
Read Only	23	CoAP.Option.Block2	0-3	var	Bi	Read Only	none
Read Only	27	CoAP.Option.Block1	0-3	var	Bi	Read Only	none
Read Only	28	CoAP.Option.Size2	0-4	var	Bi	Read Only	none

Read/Write	31	CoAP.Option.Q-Block2	0-3	var	Bi	Read/Write	none
Read Only	35	CoAP.Option.Proxy-Uri	1-1034	var	Bi	Read Only	none
Read Only	39	CoAP.Option.Proxy-Scheme	1-255	var	Bi	Read Only	none
Read Only	60	CoAP.Option.Size1	0-4	var	Bi	Read Only	none
Read Only	252	CoAP.Option.Echo	1-40	var	Bi	Read Only	none
Read Only	258	CoAP.Option.No-Response	0-1	1	Up	Read Only	0
Read Only	292	CoAP.Option.Request-Tag	0-8	var	Bi	Read Only	none

Figure 3: CoAP Options access-control

7. Normative References

[I-D.ietf-core-comi]

Veillette, M., Van der Stok, P., Pelov, A., Bierman, A., and C. Bormann, "CoAP Management Interface (CORECONF)", Work in Progress, Internet-Draft, draft-ietf-core-comi-16, 4 September 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-16>>.

[I-D.ietf-schc-architecture]

Pelov, A., Thubert, P., and A. Minaburo, "Static Context Header Compression (SCHC) Architecture", Work in Progress, Internet-Draft, draft-ietf-schc-architecture-01, 6 October 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-schc-architecture-01>>.

[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>>.

- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.
- [RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <<https://www.rfc-editor.org/info/rfc7959>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [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>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/info/rfc8613>>.
- [RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <<https://www.rfc-editor.org/info/rfc8724>>.
- [RFC8768] Boucadair, M., Reddy, K. T., and J. Shallow, "Constrained Application Protocol (CoAP) Hop-Limit Option", RFC 8768, DOI 10.17487/RFC8768, March 2020, <<https://www.rfc-editor.org/info/rfc8768>>.

- [RFC8824] Minaburo, A., Toutain, L., and R. Andreasen, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", RFC 8824, DOI 10.17487/RFC8824, June 2021, <<https://www.rfc-editor.org/info/rfc8824>>.
- [RFC9175] Amsüss, C., Preuß Mattsson, J., and G. Selander, "Constrained Application Protocol (CoAP): Echo, Request-Tag, and Token Processing", RFC 9175, DOI 10.17487/RFC9175, February 2022, <<https://www.rfc-editor.org/info/rfc9175>>.
- [RFC9177] Boucadair, M. and J. Shallow, "Constrained Application Protocol (CoAP) Block-Wise Transfer Options Supporting Robust Transmission", RFC 9177, DOI 10.17487/RFC9177, March 2022, <<https://www.rfc-editor.org/info/rfc9177>>.
- [RFC9363] Minaburo, A. and L. Toutain, "A YANG Data Model for Static Context Header Compression (SCHC)", RFC 9363, DOI 10.17487/RFC9363, March 2023, <<https://www.rfc-editor.org/info/rfc9363>>.

Appendix A. YANG Data Model

```
<CODE BEGINS> file "ietf-schc-access-control@2023-02-14.yang"
module ietf-schc-access-control {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-schc-access-control";
  prefix schc-ac;

  import ietf-schc {
    prefix schc;
  }

  organization
    "IETF IPv6 over Low Power Wide-Area Networks (lpwan) working group";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/lpwan/about/>
    WG List: <mailto:lp-wan@ietf.org>
    Editor: Juan-Carlos Zuniga
      <mailto:juancarlos.zuniga@sigfox.com>";
  description
    "
    Copyright (c) 2021 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject to
```

the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

This module extends the ietf-schc module to include the compound-ack behavior for Ack On Error as defined in RFC YYYY. It introduces a new leaf for Ack on Error defining the format of the SCHC Ack and add the possibility to send several bitmaps in a single answer.";

```

revision 2023-02-14 {
  description
    "Initial version for RFC YYYY ";
  reference
    "RFC YYYY: Compound Ack";
}

typedef rule-access-right {
  type enumeration {
    enum no-changes {
      value 0;
      description
        "No change are allowed.";
    }
    enum modify-existing-element {
      value 1;
      description
        "can modify content inside an element.";
    }
    enum add-remove-element {
      value 2;
      description
        "Allows to add or remove or modify an element.";
    }
  }
}

```

```
    }

    typedef field-access-right {
      type enumeration {
        enum no-change {
          value 0;
          description
            "Reserved slot number.";
        }
        enum change-tv {
          value 1;
          description
            "Reserved slot number.";
        }
        enum change-mo-cda-tv {
          value 2;
          description
            "Reserved slot number.";
        }
      }
    }

    }

    augment "/schc:schc/schc:rule" {
      leaf ac-modify-set-of-rules {
        config false;
        type rule-access-right;
      }
    }

    augment "/schc:schc/schc:rule/schc:nature/schc:compression" {
      leaf ac-modify-compression-rule {
        config false;
        type rule-access-right;
      }
    }

    augment "/schc:schc/schc:rule/schc:nature/schc:compression/schc:entry" {
      leaf ac-modify-field {
        config false;
        type field-access-right;
      }
    }

    }

    augment "/schc:schc/schc:rule/schc:nature/schc:fragmentation" {
      leaf ac-modify-timers {
        config false;
        type boolean;
      }
    }
  }
}
```

```
    }  
  }
```

```
}  
<CODE ENDS>
```

Appendix B. Security Considerations

TBD

Appendix C. IANA Considerations

TBD

Authors' Addresses

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

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

Ivan Martinez
Nokia Bell Labs
12 Rue Jean Bart
91300 Massy
France
Email: ivan.martinez_bolivar@nokia-bell-labs.com