IPsecme
Internet-Draft
Intended status: Standards Track
Expires: 19 August 2026

D. Migault
Ericsson
M. Hatami
S. C辿spedes
W. Atwood
Concordia University
D. Liu
Ericsson
T. Guggemos
LMU
C. Bormann
Universitaet Bremen TZI
D. Schinazi
Google LLC

15 February 2026

ESP Header Compression with Diet-ESP
draft-ietf-ipsecme-diet-esp-10

Abstract

This document specifies Diet-ESP, a compression mechanism for control information in IPsec/ESP communications. The compression uses Static Context Header Compression rules.

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 19 August 2026.

Copyright Notice

Table of Contents

1. Requirements Notation

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

2. Introduction

The Encapsulating Security Payload (ESP) [RFC4303] protocol is part of the IPsec [RFC4301] suite of protocols and can provide confidentiality, data origin authentication, integrity, anti-replay, and traffic flow confidentiality. The set of services ESP provides depends on the Security Association (SA) parameters negotiated between devices.

An ESP packet is composed of the ESP Header, the ESP Payload Data, the ESP Trailer, and the Integrity Check Value (ICV). ESP has two modes of operation: Transport and Tunnel. In Transport mode, the ESP Payload Data consists of the payload of the original IP packet; the ESP Header is inserted after the original IP packet header. In Tunnel mode, commonly used for VPNs, the ESP Header is placed after an outer IP header and before the inner IP packet headers of the original datagram. This ensures both the original IP headers and payload are protected. Consequently, the ESP Data Payload field contains either the payload from the original IP packet or the fully-encapsulated IP packet, in transport mode or tunnel mode, respectively.

The ESP Trailer, placed at the end of the ESP Payload Data, includes fields such as Padding and Pad Length to ensure proper alignment, and Next Header to indicate the protocol following the ESP header. The ICV, calculated over the ESP Header, ESP Data Payload, and ESP Trailer, is appended after the ESP Trailer to ensure packet integrity. For a simplified overview of ESP, readers are referred to Minimal ESP [RFC9333].

In Figure 1, the Payload Data, ESP Padding, Pad Length, and Next Header compose the Cleartext ESP Packet (CTE) to be protected by encryption. The Authentication Coverage extends from SPI through Padding, while the Encryption Coverage applies to the Payload Data and optional Padding, Pad Length, and Next Header fields.

While ESP is effective in securing traffic, compression can reduce packet sizes, enhancing performance in networks with limited bandwidth. In such environments, reducing the size of transmitted packets is essential to improve efficiency. This document defines Diet-ESP, a protocol that includes compression/decompression (C/D) of the various structures processed by ESP. The C/D uses Static Context Header Compression and Fragmentation (SCHC) rules [RFC8724]. The structure of a standard uncompressed ESP packet is shown in Figure 1.

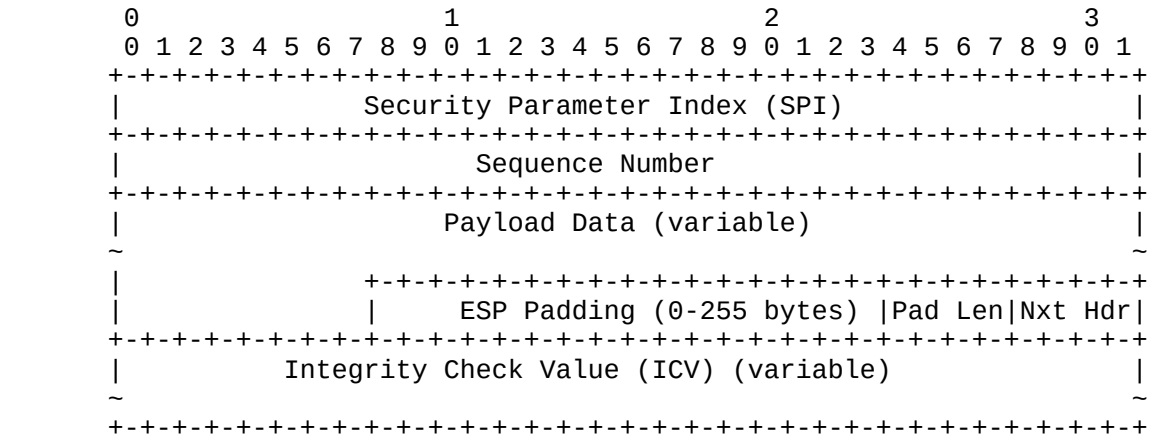

Figure 1: Top-Level Format of a standard ESP Packet. The ESP Data Payload, ESP Padding, Pad Length, and Next Header compose the Clear Text ESP Packet (CTE) to be protected by encryption.

2.1. The Three Compressors Described in this Specification

The document outlines the three compressors utilized in Diet-ESP, which are detailed as follows:

1. Inner IP Compression (IIPC): The original packet to be protected by ESP, namely, the Inner IP packet (IIP), is split into a Header subject to compression and a Payload (see Section 4 for more details). The process in the IIPC pertains to the compression and decompression of fields of the Header of the IIP. For outbound packets, after IIPC, ESP incorporates the resulting SCHC Packet and the (unaltered) Payload of the IIP into the ESP Data Payload of a Clear Text ESP packet (CTE) (refer to Figure 1). In

the case of inbound packets, decompression occurs after the compressed Header is retrieved from the ESP Payload Data within the CTE.

2. Clear Text ESP Compression (CTEC): This process pertains to the compression and decompression of the segment of the ESP packet that is covered by encryption. This encompasses the ESP Data Payload and the ESP Trailer, which includes the Padding, Pad Length, and Next Header fields, as illustrated in Figure 1. For the CTEC stage, only these three ESP Trailer fields are eligible for compression. For outbound packets, ESP subsequently encrypts the compressed CTE packet. For inbound packets, decompression takes place following the decryption process of the ESP.

3. Encrypted ESP Compression (EEC): This process pertains to the compression and decompression of the Encrypted ESP packet (EE), which consists of the ESP Header, the encrypted payload, and the Integrity Check Value (ICV). Since neither the encrypted payload nor the ICV can be compressed, only the ESP Header, specifically the SPI and SN fields, is subject to compression.

Figure 2 depicts the incorporation of Diet-ESP compressors within the IPsec framework.


Endpoint
Inner IP packet
(Outbound)
Endpoint
Inner IP packet
(Inbound)
IPsec
SA lookup
ESP
Security Association
- Attributes for Rule Derivation
Derivation of the IIPC Rule,
CTEC Rule, and EEC Rule
IIPC:
|C(Header)|Payload|
Formation of
Clear Text ESP Packet
CTEC:
|C(Header)|
Payload|C(ET)|
Encryption
Formation of
Encrypted ESP Packet
EEC:
|C(EH)|E{C(Header)|
Payload|C(ET)}|ICV|
IIPC:
DC(C(Header))|Payload|
Extraction of
Data Payload
CTEC:
|C(Header)|
Payload|DC(C(ET))|
Decryption
Parsing
Encrypted ESP
EEC:
DC(C(EH))|E{C(Header)|
Payload|C(ET)}|ICV|
SA lookup
Outbound Traffic
Inbound Traffic

Figure 2: SCHC Integration into the IPsec Stack. Packets are described for IPsec in tunnel mode. C(.) and DC(.) designate compression/ decompression for the fields inside, and E{.} designates encryption. IIP refers to the Inner IP packet, EH refers to the ESP Header, and ET refers to the ESP Trailer. IIPC, CTEC and EEC respectively designate the Inner IP Compressor, the Clear Text ESP Compressor, and the Encrypted ESP Compressor.

IPsec requires that both endpoints agree on a shared context known as the Security Association (SA). This SA is established via IKEv2 and encompasses all Attributes for Rule Derivation (AfRD) (refer to Section 4.2) essential for formulating the Rules for each compressor defined in Section 2.1, specifically the Inner IP packet Compressor (IIPC), the Clear Text ESP Compressor (CTEC), and the Encrypted ESP Compressor (EEC).

When an Inner IP packet (IIP) is received, IPsec identifies the SA linked to that packet. Upon the ESP determining the IIPC Rule from the AfRD contained within the SA, the IIPC pre-processes the the IIP and separates it into Header and Payload. Only the Header is sent for compression. The compressed Header is composed of RuleID, Compressed Residue, and an optional Padding field. The original Payload of the IIP is then appended after the compressed Header, resulting in an IIPC packet with format |C(Header)|Payload| where C(.) indicates compression. Subsequently, ESP constructs the Clear Text ESP packet (CTE). The CTEC Rule, which is derived from the AfRD of the SA, allows for the compression of the CTE, resulting in a CTEC packet with format |C(Header)|Payload|C(ET)|, where ET represents the ESP Trailer. Then, ESP encrypts the ESP Data Payload, computes the Integrity Check Value (ICV), and forms the Encrypted ESP packet (EE) formatted as |EH|E{C(Header)|Payload|C(ET)}|ICV|, where EH represents the ESP Header and E{.} indicates encryption. The EEC Rule that has been derived from the AfRD of the SA, is then utilized to compress the EE, resulting in an EEC packet with format |C(EH)|E{C(Header)|Payload|C(ET)}|ICV|. The resulting compressed ESP packet is integrated into an IP packet and transmitted as outbound traffic.

For inbound traffic, the endpoint extracts the Security Parameter Index (SPI) from the compressed EE, along with any other selectors from the packet, to conduct a lookup for the SA. As outlined in Section 9, since the SPI is derived from a potentially compressed ESP Header, there may be instances where the endpoint must explore multiple options, potentially leading to several lookups or, in the worst-case scenario, multiple signature verifications (see Section 9 for a more detailed discussion).

Once the SA is retrieved, the ESP accesses the AfRD to ascertain the EEC Rule and proceeds to decompress the EE. The ESP verifies the signature prior to decryption. Following this, the decompressor applies the CTEC Rule derived from the AfRD of the SA, allowing for the subsequent decompression. Finally, ESP extracts the Data Payload from the CTE packet, retrieves the IIPC Rule and decompresses the Header.

Note that implementations MAY differ from the architectural description but it is assumed that the output will be the same.

2.2. The Scope of SCHC in this Specification

SCHC [RFC8724] offers a mechanism for header compression as well as an optional fragmentation feature. SCHC facilitates the compression and decompression of headers by utilizing a common context that may encompass multiple Rules. Each Rule is designed to correspond with specific values or ranges of values within the header fields. When a Rule is successfully matched, the corresponding header fields are substituted with the Rule ID and the Compression Residue. The Compression Residue for the packet header is the concatenation of the non-empty residues for each field of the header. The Compression Residue is directly followed by the packet payload and an optional padding to ensure byte alignment.

This document utilizes SCHC as a practical means to illustrate the capability to compress and decompress a structured payload. It is important to note that any elements of SCHC that pertain to aspects other than compression or decompression, such as fragmentation, fall outside the purview of this document. The reference to SCHC herein is solely for descriptive purposes related to compression and decompression, and it is not anticipated that the general SCHC framework will be integrated into the ESP implementation. The structured payloads addressed in this specification pertain to internal structures managed by ESP for the processing of an IP packet. Consequently, the compression and decompression processes outlined in this document represent supplementary steps for the ESP stack in handling the ESP packet.

2.3. Diet-ESP Rules and Context

In IPsec, the process of encryption or decryption between IPsec peers necessitates a common context known as a Security Association (SA). More broadly, the SA encompasses all essential parameters required by the ESP to handle both inbound and outbound packets. SAs are unidirectional. Furthermore, IPsec can link both outbound and inbound IP packets to the SA through Traffic Selectors (TS) or Security Parameter Index (SPI). This capability allows IPsec to

uniquely associate outbound and inbound packets with a specific context (SA), which contains all pertinent information for IPsec processing.

This document adopts a comparable methodology for compression and decompression, ensuring that the SA includes all necessary parameters to create the Rules applicable for compressing or decompressing each structured payload. The Rule associated with each structured payload is generated based on specific parameters referred to in this document as Attributes for Rule Derivation (AfRD) (see Section 4.2 for a more detailed description). These AfRDs are negotiated through IKEv2 [RFC7296], and in such cases, they are likely already included in the SA. Any additional missing AfRDs are negotiated via [I-D.ietf-ipsecme-ikev2-diet-esp-extension].

3. Terminology

ESP Header Compression: A method to reduce the size of ESP headers and trailer using predefined compression rules and contexts to improve efficiency.

ESP Trailer: A set of fields added at the end of the ESP payload, including Padding, Pad Length, and Next Header, used to ensure alignment and indicate the next protocol.

Inner IP C/D (IIPC): Process that compresses/decompresses the inner IP packet headers.

Clear Text ESP C/D (CTEC): Process that compresses/decompresses all fields that will later be encrypted by ESP, which include the ESP Data Payload and ESP Trailer.

Encrypted ESP C/D (EEC): Process that compresses/decompresses ESP fields not encrypted by ESP.

Security Parameter Index (SPI): As defined in [RFC4301], Section 4.1.

Sequence Number (SN): As defined in [RFC4303], Section 2.2.

Static Context Header Compression (SCHC): A framework for header compression designed for LPWANs, as defined in [RFC8724].

Static Context Header Compression Rules (SCHC Rules): As defined in [RFC8724]

RuleID: A unique identifier for each Rule part of the Diet-ESP context.

SCHC Parameters: A set of predefined values used for SCHC compression and decompression, ensuring byte alignment and proper packet formatting based on the SCHC profile.

Traffic Selector (TS): A set of parameters (e.g., IP address range, port range, and protocol) used to define which traffic should be protected by a specific Security Association (SA).

It is assumed that the reader is familiar with other SCHC terminology defined in [RFC8376], [RFC8724], and eventually [I-D.ietf-schc-architecture].

4. Diet-ESP Integration into the IPsec Stack

4.1. SCHC Parameters for Diet-ESP

The SCHC Packet [RFC8724] is always in the form:

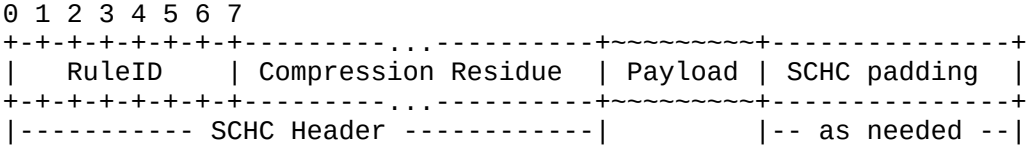

Figure 3: SCHC Packet

The RuleID is a unique identifier for each SCHC Rule. It is included in packets to ensure the receiver applies the correct decompression rule, maintaining consistency in packet processing. Note that the Rule ID does not need to be explicitly agreed upon and can be defined independently by each party. Furthermore, [RFC8724] indicates that the way the RuleID is sent is left open to the profile specification. The RuleID in Diet-ESP is expressed as 1 byte but it can be elided as there is a unique Rule determined for the compressors. The 1 byte may be used in future implementations to support multiple flows over the same SA.

SCHC padding in SCHC serves the purpose of aligning data to a designated boundary, which is typically byte-aligned or aligned to 8 bits. This document presumes that this field is not utilized in CET and EEC. This document outlines a simpler form of padding for byte-alignment, as detailed in section Section 6.1. Such alignment is essential to ensure that encryption is applied to data that is byte-aligned. The rationale for employing a padding method other than SCHC Padding is to accommodate the length of the compressed ESP Payload Data.

Another variable required for the C/D in Diet-ESP is the Maximum Packet Size (MAX_PACKET_SIZE) determined by the specific IPsec ESP configuration and the underlying transport, but it is typically aligned with the network's MTU. The size constraints are optimized based on the available link capacity and negotiated parameters between endpoints.

4.2. Attributes for Rules Derivation

The list of attributes for Rules Derivation (AfRD) is shown in Table 1. These attributes are used to express the various compressions that operate at the IIPC, CTEC, and EEC.

As outlined in Section 4, this specification does not detail the process by which the AfRD are established between peers. Instead, such negotiations are addressed in [I-D.ietf-ipsecme-ikev2-diet-esp-extension]. However, the AfRD can be classified into two distinct categories. The first category encompasses AfRD that are negotiated through a specific IKEv2 extension tailored for the negotiation of AfRD linked to a particular profile, the Diet-ESP profile in this context. The AfRD referenced in Table 1 in this category are: the DSCP Action dscp_action, the ECN Action ecn_action, the Flow Label Action flow_label_action, the ESP alignment alignment, the ESP SPI Least Significant Bits (LSB) esp_spi_lsb, and the ESP Sequence Number LSB esp_sn_lsb.

The second category pertains to AfRD that are negotiated through IKEv2 exchanges or extensions that are not specifically designed for compression purposes. This category includes AfRD associated with TS, as identified in Table 1, which are the TS IP Version ts_ip_version, the TS IP Source Start ts_ip_src_start, the TS IP Source End ts_ip_src_end, the TS IP Destination Start ts_ip_dst_start, the TS IP Destination End ts_ip_dst_end, the TS Protocol ts_proto, the TS Port Source Start ts_port_src_start, the TS Port Source End ts_port_src_end, the TS Port Destination Start ts_port_dst_start, and the TS Port Destination End ts_port_dst_end. These AfRD are derived from the Traffic Selectors established through TSi/TSr payloads during the IKEv2 CREATE_CHILD_SA exchange, as described in [RFC7296], Section 3.13. The AfRD IPsec Mode designated as ipsec_mode in Table 1 is determined by the presence or absence of the USE_TRANSPORT_MODE Notify Payload during the CREATE_CHILD_SA exchange, as detailed in [RFC7296], Section 1.3.1. The AfRD Tunnel IP designated as tunnel_ip in Table 1 is obtained from the IP address of the IKE messages exchanged during the CREATE_CHILD_SA process, as noted in [RFC7296], Section 1.1.3. The AfRDs designated as ESP Encryption Algorithm esp_encr and ESP Security Parameter Index (SPI) esp_spi in Table 1 are established through the SAi2/SAr2 payloads during the CREATE_CHILD_SA exchange, while the AfRD designated as ESP

Sequence Number esp_sn in Table 1 is initialized upon the creation of the Child SA and incremented for each subsequent ESP message. The DSCP values identified as dscp_list in Table 1 MAY be established through the DSCP Notify Payload [I-D.mglt-ipsecme-dscp-np].

The ability to derive the IIP Compressor Rules for the internal IP packet from the agreed Traffic Selectors is indicated by the variable iipc_profile.

Variable	Possible Values	Reference	Compressor
iipc_profile	"iipc_diet-esp", "iipc_not_compressed"	ThisRFC	N/A
dscp_action	"not_compressed", "lower", "sa"	ThisRFC	IIPC
ecn_action	"not_compressed", "lower"	ThisRFC	IIPC
flow_label_action	"not_compressed", "lower", "generated", "zero"	ThisRFC	IIPC
ts_ip_version	"IPv4-only", "IPv6-only"	RFC7296	IIPC
ts_ip_src_start	IPv4 or IPv6 address	RFC7296	IIPC
ts_ip_src_end	IPv4 or IPv6 address	RFC7296	IIPC
ts_ip_dst_start	IPv4 or IPv6 address	RFC7296	IIPC
ts_ip_dst_end	IPv4 or IPv6 address	RFC7296	IIPC
ts_proto	TCP, UDP, UDP-Lite, SCTP, ANY, ...	RFC7296	IIPC
ts_port_src_start	Port number	RFC7296	IIPC
ts_port_src_end	Port number	RFC7296	IIPC
ts_port_dst_start	Port number	RFC7296	IIPC
ts_port_dst_end	Port number	RFC7296	IIPC
dscp_list	list of DSCP numbers	RFCYYYY	IIPC

alignment	"8 bit", "16 bit", "32 bit", "64 bit"	ThisRFC	CTEC
esp_trailer	"Mandatory", "Optional"	ThisRFC	CTEC
ipsec_mode	"Tunnel", "Transport"	RFC4301	CTEC
tunnel_ip	IPv4 or IPv6 address	RFC4301	CTEC
esp_encr	ESP Encryption Algorithm	RFC4301	CTEC
esp_spi	ESP SPI	RFC4301	EEC
esp_spi_lsb	0-32	ThisRFC	EEC
esp_sn	ESP Sequence Number	RFC4301	EEC
esp_sn_lsb	0-32	ThisRFC	EEC

Table 1: Attributes for Rule Derivation (AfRD) to generate IIPC, CTEC and EEC Rules in Diet-ESP

Any variable starting with "ts_" is associated with the Traffic Selectors (TSi/TSr) of the SA. The notation is introduced by this specification but the definitions of the variables are in [RFC4301] and [RFC7296].

The Traffic Selectors may result in a quite complex expression, and this specification restricts that complexity. This specification restricts the resulting TSi/TSr to a single type of IP address (IPv4 or IPv6), a single protocol (e.g., UDP, TCP, or ANY), a single port range for source and destination. This specification presumes that the Traffic Selectors can be articulated as a result of CREATE_CHILD_SA with only one Traffic Selector [RFC7296], Section 3.13.1 in both TSi and TSr payloads (as described in [RFC7296], Section 3.13). The TS Type MUST be either TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE.

Let the resulting Traffic Selectors TSi/TSr be expressed via the Traffic Selector structure defined in [RFC7296], Section 3.13.1. We designate the local TS the TS - either TSi or TSr - sent by the local peer. Conversely we designate as remote TS the TS - either TSi or TSr - sent by the remote peer.

The details of each parameter are the following:

iipc_profile: designates the behavior of the IIPC layer. When set to "iipc_not_compressed" IIPC is not performed. This specification describes IIPC that corresponds to the "iipc_diet-esp" profile.

flow_label_action: indicates the Flow Label Action, that is how the Flow Label field of the inner IPv6 packet or the Identification field of the inner IPv4 packet is compressed / decompressed - See Section 5 for more information. In a nutshell, "not_compressed" indicates that Flow Label (resp. Identification) is not compressed and the field is sent as-is using value-sent. "lower" (using compute-*) indicates the value is read from the outer IP header - eventually with some adaptations when inner IP packet and outer IP packets have different versions. This field is computed from the outer IP header using compute-* (MO: ignore, CDA: compute-*)."generated" indicates the value is re-generated at the decompressor side (CDA: compute-*). In that case, the decompressed value may take a different value compared to its original value. "zero" indicates the field is removed by matching against a target value of 0 (MO: equal, CDA: not-sent). See Section 5 for further details on the applied CDA logic.

dscp_action: indicates the DSCP Action, that is how the DSCP values of the inner IP packet are compressed / decompressed - See Section 5 for more information. In a nutshell, "not_compressed" indicates that DSCP are not compressed. "lower" indicates the value is read from the outer IP header - eventually with some adaptations when inner IP packet and outer IP packets have different versions (compute-*). "sa" indicates that compression is performed according to the pre-negotioated list of DSCP values (dscp_list) agreed by the SA, using SCHC’s MO and index (CDA). See Section 5 for rule examples.

ecn_action: indicates ECN Action, that is how the ECN values of the inner IP packet are compressed / decompressed - See Section 5 for more information. In a nutshell, "not_compressed" indicates that DSCP are not compressed. "lower" indicates the value is read from the outer IP header using compute-* (eventually with some adaptations when inner IP packet and outer IP packets have different versions).

ts_ip_version: designates the TS IP version. Its value is set to "IPv4-only" when only IPv4 IP addresses are considered and to "IPv6-only" when only IPv6 addresses are considered. Practically, when IKEv2 is used, it means that the agreed TSi or TSr results only in a mutually exclusive combination of TS_IPV4_ADDR_RANGE or

TS_IPV6_ADDR_RANGE payloads. If TS Type of the resulting TSi/TSr is set to TS_IPV4_ADDR_RANGE, ts_ip_version takes the value "IPv4-only". Respectively, if TS Type is set to TS_IPV6_ADDR_RANGE, ts_ip_version is set to "IPv6-only".

ts_ip_src_start: designates the TS IP Source Start, that is the starting value range of source IP addresses of the inner packet and has the same meaning as the Starting Address field of the local TS.

ts_ip_src_end: designates TS IP Source End, that is the high end value range of source IP addresses of the inner packet and has the same meaning as the Ending Address field of the local TS.

ts_ip_dst_start: designates the TS IP Destination Start, that is the starting value range of destination IP addresses of the inner packet and has the same meaning as the Starting Address field of the remote TS.

ts_ip_dst_end: designates the TS IP Destination End, that is the high end value range of destination IP addresses of the inner packet and has the same meaning as the Ending Address field of the remote TS.

ts_proto: designates the TS Protocol, that is the Protocol ID of the resulting TSi/TSr. This profile considers the specific protocol values "TCP", "UDP", "UDP-Lite", "SCTP", and "ANY". The representation of "ANY" is given in [RFC4301], Section 4.4.4.2.

ts_port_src_start: designates the TS Port Source Start, that is the the starting value of the source port range of the inner packet and has the same meaning as the Start Port field of the local TS.

ts_port_src_end: designates the TS Port Source End, that is the high end value range of the source port range of the inner packet and has the same meaning as the End Port field of the local TS.

ts_port_dst_start: designates TS Port Destination Start, that is the starting value of the destination port range of the inner packet and has the same meaning as the Start Port field of the remote TS.

ts_port_dst_end: designates TS Port Destination End, that is the high end value range of the destination port range of the inner packet and has the same meaning as the End Port field of the remote TS.

In Diet-ESP compression, fields such as IP addresses and port numbers are split into a stable prefix and a variable suffix. The stable part is captured in the rule as the Target Value (TV), expressed with prefix_bits(start, end). The Matching Operator MO = MSB(x) verifies that the most significant x bits of the Field Value (FV) match this prefix. When the match succeeds, the Compression/Decompression Action CDA = LSB transmits only the remaining low-order bits (FL - x), where FL is the full field length in bits.

Formally, the operation of MO = MSB(x) is defined as: (FV & (((1 << x) - 1) << (FL - x))) == (TV & (((1 << x) - 1) << (FL - x)))

where FL is the total field length in bits. The parameter x is computed as:

x = FL - prefix_bits_length

dscp_list: designates the list of DSCP values associated to the inner traffic - see for example [I-D.mglt-ipsecme-dscp-np]. These are not Traffic Selectors, but the compression mandates that the packets take one of these listed DSCP values.

alignment: designates the ESP alignment as defined by [RFC4303].

esp_trailer: When configured to "Mandatory," it signifies that the implementation requires the ESP Trailer to include a Next Header and a Pad Len field, as outlined in [RFC4303]. This requirement primarily aims to ensure compatibility with current hardware implementations of ESP, as detailed in [RFC4303]. Conversely, if set to "Optional," it indicates that the implementation is capable of supporting the compression of the ESP Trailer.

ipsec_mode: designates the IPsec Mode defined in [RFC4301]. In this document, the possible values are "tunnel" for the Tunnel mode and "transport" for the Transport mode.

tunnel_ip: designates the Tunnel IP address of the tunnel defined in [RFC4301]. This field is only applicable when the Tunnel mode is used. That IP address can be an IPv4 or IPv6 address.

esp_encr: designates the ESP Encryption Algorithm - also designated as Transform 1 in [RFC7296], Section 3.3.2. The algorithm is needed to determine whether the ESP Payload Data needs to be aligned to some predefined block size and if the ESP Pad Length and Padding fields can be compressed. For the purpose of compression it is RECOMMENDED to use algorithms that already compressed their IV [RFC8750].

esp_spi: designates the Security Parameter Index defined in [RFC4301].

esp_spi_lsb: designates the LSB to be considered for the compressed SPI. A value of 32 for esp_spi_lsb will leave the SPI unchanged.

esp_sn: designates the ESP Sequence Number (SN) field defined in [RFC4301].

esp_sn_lsb: designates the LSB to be considered for the compressed SN. It works similarly to ESP SPI LSB (see esp_spi_lsb).

5. Examples of Diet-ESP SCHC Rule Tables

The tables in this section provide examples of SCHC compression rules for Diet-ESP in IPsec. The Compression/Decompression Actions (CDAs), derived from the AfRD discussed in Section 4.2, specify how various IPv6, UDP/TCP, and ESP header fields are compressed and decompressed,

Table 2, Table 3, and Table 4 collectively represent an example configuration for Diet-ESP compression, where each compressor stage applies its own dedicated rule as per [RFC8724] Section 7.2. In Table 2, the field flow_label_action is set to lower, which results in MO = ignore and CDA = compute-*, indicating the value is derived from the outer header. The dscp_action is set to *not_compressed*, corresponding to MO = ignore and CDA = value-sent, meaning the DSCP field is transmitted in full. IPv6 address and port fields are compressed using the MSB/LSB strategy, where the MO = MSB(x) checks whether the most significant x bits of the Field Value (FV) equal the rule’s Target Value (TV), and transmits only the variable suffix (FL - x bits).

The ESP-related fields, including padding and alignment, are compressed using a separate rule in Table 3, while the ESP SPI and Sequence Number are compressed independently using the rule in Table 4. Each rule includes only the fields visible and relevant to its corresponding compression stage.

5.1. Diet-ESP with Inner IP and UDP/TCP Header Compression

5.1.1. Inner IP Compression (IIPC) Rule

This rule defines the compression behavior for the IIPC, which operates before ESP encapsulation and encryption. The example handles all fields from the inner IPv6 and transport layer headers (e.g., UDP, TCP, SCTP) that are visible at this stage.

Version, Next Header, and Payload Length are elided or derived based on known values or outer headers.

Traffic Class fields (DSCP and ECN) are partially elided or transmitted depending on their CDA.

Flow Label is computed from the outer header (flow_label_action = lower).

Source/Destination Addresses and Ports are compressed using MSB/LSB strategy based on Traffic Selectors.

Checksum and UDP Length are elided and computed at the decompressor.

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
Version	3	1	Bi	ts_ipversion	equal	not-sent	0
DSCP	6	1	Dw	-	ignore	value-sent	6
ECN	2	1	Dw	-	ignore	value-sent	2
Flow Label	20	1	Dw	-	ignore	compute-*	0
Payload Length	16	1	Bi	-	ignore	compute-*	0
Next Header	8	1	Bi	ts_proto	equal	not-sent	0
Hop Limit	8	1	Dw	-	ignore	compute-*	0
Source Address	128	1	Bi	prefix_bits (ts_ip_src_start, ts_ip_src_end)	MSB(x)	LSB	FL-x
Destination Address	128	1	Bi	prefix_bits (ts_ip_dst_start, ts_ip_dst_end)	MSB(x)	LSB	FL-x
Source Port (UDP/TCP)	16	1	Bi	prefix_bits (ts_port_src_start, ts_port_src_end)	MSB(x)	LSB	FL-x
Destination Port (UDP/ TCP)	16	1	Bi	prefix_bits (ts_port_dst_start, ts_port_dst_end)	MSB(x)	LSB	FL-x
Checksum	16	1	Bi	-	ignore	compute-*	0
UDP Length	16	1	Bi	-	ignore	compute-*	0

Table 2: IIPC Rule: Compression of Inner IP and Transport Headers

5.1.2. Clear Text ESP Compression (CTEC) Rule

This rule is used by the CTEC compressor, which processes ESP-specific fields just before encryption. These fields primarily deal with formatting and alignment.

ESP Padding is used to ensure that the ESP payload aligns to the encryption algorithm’s block size or protocol requirements. It is not sent and is instead calculated (CDA = compute-*).

Byte Alignment ensures that the SCHC packet respects the alignment required by ESP and IPv6 extension header parsing (e.g., 64-bit alignment).

These fields are handled separately because they are relevant only at the Clear Text ESP Packet construction stage and are not part of the inner packet or ESP Header fields.

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
ESP Padding	-	-	-	-	ignore	compute-*	0
Byte Alignment	8	1	Bi	-	ignore	compute-*	0

Table 3: CTEC Rule: ESP Trailer and Alignment Compression

5.1.3. Encrypted ESP Compression (EEC) Rule

This rule focuses on compressing the ESP Header fields that are still visible post-encryption. These fields are typically stable over a session and can be compressed without compromising lookup reliability or replay protection.

SPI (Security Parameter Index) and Sequence Number (SN) are compressed using the Least Significant Bits (LSB) approach.

MSB(8) signifies that only 8 bits of variation need to be sent (leaving 24 fixed bits).

Field	FL	FP	DI	TV	MO	CDA	Sent [bits]
SPI	32	1	Dw	-	MSB(x)	LSB	FL - x
SN	32	1	Dw	-	MSB(x)	LSB	FL - x

Table 4: EEC Rule: ESP SPI and Sequence Number Compression

5.2. SA-Based DSCP Compression with Zeroed Flow Label

Table 5, Table 6, and Table 7 illustrate rule examples where the DSCP field is compressed using a pre-negotiated Security Association (SA) list. Here, dscp_action = sa results in MO = match-mapping and CDA = index, meaning the field value is matched from a list and only the index is sent. The ECN field is computed from the outer header (ecn_action = lower), so it is not sent. The Flow Label is removed from transmission by setting a target value of 0 (flow_label_action = zero), which results in MO = equal and CDA = not-sent.

Field	MO	CDA	Sent [bits]
DSCP	match-mapping	index	3
ECN	ignore	compute-*	0
Flow Label	equal (TV=0)	not-sent	0

Table 5: Example of Diet-ESP IIPC Rule table where the DSCP field is compressed using a pre-negotiated Security Association (SA) list

Field	MO	CDA	Sent [bits]
ESP Padding	ignore	compute-*	0
Byte Alignment	ignore	compute-*	0

Table 6: Example of Diet-ESP EEC Rule table where the DSCP field is compressed using a pre-negotiated Security Association (SA) list

Field	MO	CDA	Sent [bits]
SPI	MSB(x)	LSB	FL - x
SN	MSB(x)	LSB	FL - x

Table 7: Example of Diet-ESP EEC Rule table where the DSCP field is compressed using a pre-negotiated Security Association (SA) list

5.3. Full Header Transmission Without Compression

Table 8, Table 9, and Table 10 show a configuration disabling all compression mechanisms. Every field is transmitted in full using MO = ignore and CDA = value-sent, which is the result of setting each cda attribute to not_compressed in the AfRD.

Field	MO	CDA	Sent [bits]
DSCP	ignore	value-sent	6
ECN	ignore	value-sent	2
Flow Label	ignore	value-sent	20
Payload Length	ignore	value-sent	16
Hop Limit	ignore	value-sent	8

Table 8: Example of Diet-ESP IIPC Rule table with all compression mechanisms disabled

Field	MO	CDA	Sent [bits]
ESP Padding	ignore	compute-*	0
Byte Alignment	ignore	compute-*	0

Table 9: Example of Diet-ESP IIPC Rule table with all compression mechanisms disabled

Field	MO	CDA	Sent [bits]
SPI	ignore	value-sent	32
SN	ignore	value-sent	32

Table 10: Example of Diet-ESP EEC Rule table with all compression mechanisms disabled

5.4. Mixed Compression Strategy

Table 11 to Table 13 demonstrate a mixed strategy. The Flow Label is generated at the decompressor side and not transmitted, after following the attribute flow_label_action = generated, which results in MO = ignore and CDA = compute-*. The ECN field is derived from the outer header, while the DSCP field is transmitted as-is (dscp_action = not_compressed).

Field	MO	CDA	Sent [bits]
DSCP	ignore	value-sent	6
ECN	ignore	compute-*	0
Flow Label	ignore	compute-*	0

Table 11: Example of Diet-ESP IIPC Rule table with a mixed strategy

Field	MO	CDA	Sent [bits]
ESP Padding	ignore	compute-*	0
Byte Alignment	ignore	compute-*	0

Table 12: Example of Diet-ESP IIPC Rule table with a mixed strategy

Field	MO	CDA	Sent [bits]
SPI	MSB(x)	LSB	x
SN	MSB(x)	LSB	x

Table 13: Example of Diet-ESP EEC Rule table with a mixed strategy

6. Diet-ESP for IPsec in Tunnel Mode

6.1. Inner IP Compression (IIPC)

When the iipc_profile attribute is set to "iipc_not_compressed", the inner IP Packet (IIP) is not compressed. When iipc_profile is set to "iipc_diet-esp", IIPC proceeds with the compression. The Header fields subject to compression depend on the encapsulation mode. When ipsec_mode is set to "tunnel", all Header fields of the IIP structure are subject to compression. ts_ip_version determines how the IPv6 header (resp. the IPv4 header) is compressed - see Section 6.1.2 (resp. Section 6.1.3). Conversely, when ipsec_mode is set to "transport", the compression applies only to Header fields other than the IPv4 or IPv6 header fields. This means that the IPv4 and IPv6 headers remain uncompressed, while other Header fields within the IIP structure are subject to compression.

Note that the SCHC packet illustrated in Figure 3 appends the padding at the end of the SCHC Packet. This approach presents notable challenges, including handling a Payload that lacks byte alignment. Furthermore, the absence of a specified Payload length would necessitate the inclusion of the padding length within the padding itself, which would require a particular padding construction akin to that utilized by the ESP Padding, thereby posing difficulties for hardware implementations.

To address these issues, IIPC uses a pre-processing phase where the IIP is divided into two segments: the Header and the Payload and only the Header is considered as the candidate structure for SCHC compression. The division between Header and Payload can only occur because all the Header fields are of a fixed size. By construction, the compression process applies the defined rule to the Header, resulting in a SCHC Packet (refer to Figure 4) that features a Rule ID, a Compression Residue, a Padding field, and an empty SCHC Payload field.

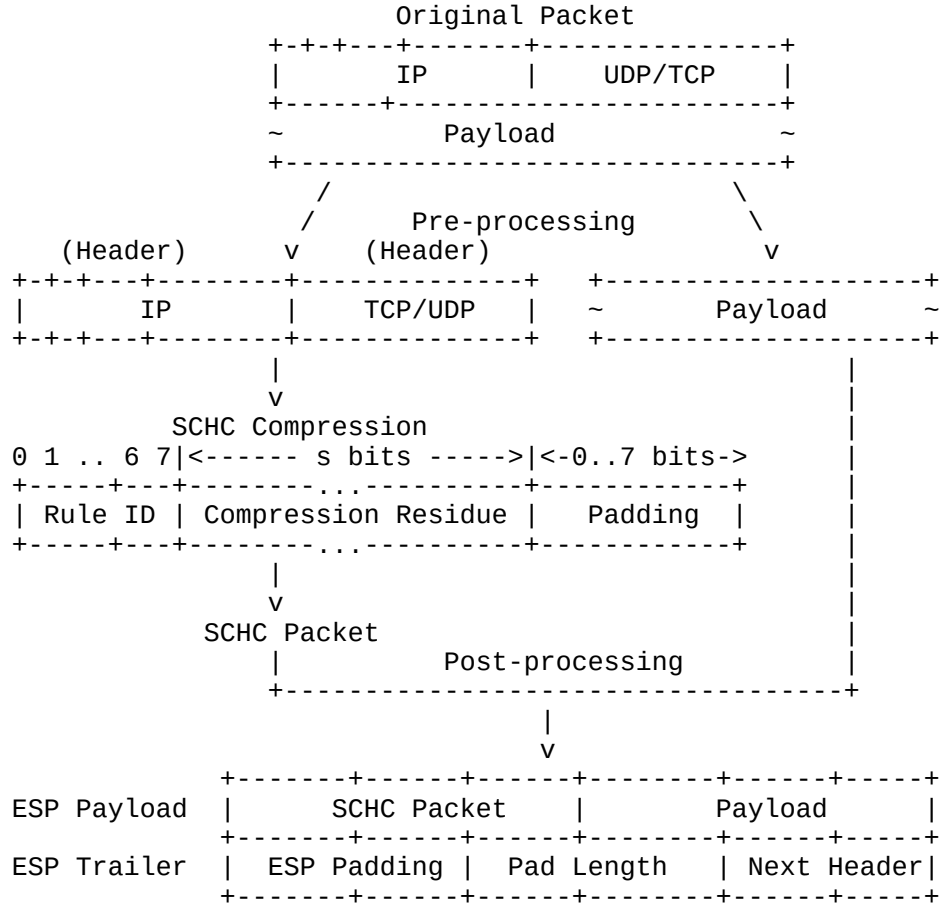

Figure 4: CTEC packet construction

Ultimately, a post-processing phase merges the SCHC Packet with the Payload. Note that the resulting SCHC packet is byte-aligned.

6.1.1. Compression of the Inner IP payload

This section describes the compression of the inner IP payload. The compression described herein only affects UDP, UDP-Lite, TCP or SCTP segments. The type of segment is specified in the IP header.

For UDP, UDP-Lite, TCP and SCTP segments, source ports, destination ports, and checksums are compressed. For source port (resp. destination port) only the least significant bits are sent. FL is set to 16 bits, TV is set to prefix_bits(ts_port_src_start, ts_port_src_end) (resp. ts_port_dst_start, ts_port_dst_end), MO is

set to "MSB" and CDA to "LSB". The checksum is elided, FL is set to 16 bits, TV is not set, MO is set to "ignore" and CDA is set to "checksum". This may result in decompressing a zero-checksum UDP packet with a valid checksum, but this has no impact as a valid checksum is universally accepted.

For UDP or UDP-Lite the length field is elided. FL is set to 16, TV is not set, MO is set to "ignore".

6.1.2. Compression of the Inner IPv6 header

The version field is elided, FL is set to 3, TV is set to ts_ip_version, MO is set to "equal" and CDA is set to "not-sent".

Traffic Class is composed of the 6 bit DSCP and 2 bit ECN. The compression of DSCP and ECN are defined independently.

DSCP values are compressed according to the dscp_action value:

* If dscp_action is set to "not_compressed", the DSCP values are included in the inner IP header. FL is set to 6 bits, TV is not set, MO is set to "ignore", CDA is set to "sent-value".

* If dscp_action is set to "lower", the DSCP field is elided and its value is copied from the Tunnel IP header. FL is set to 6 bits, TV is not set, MO is set to "ignore", CDA is set to "lower".

* If dscp_action is set to "sa", DSCP is compressed according to the DSCP values of the SA. If dscp_list contains a single element, the DSCP is elided, FL is set to 6 bits, TV is set to dscp_list[0], MO is set to "equal" and CDA is set to "not-sent". If dscp_list contains more than one DSCP value, FL is set to 6 bits, TV is set to dscp_list, MO is set to "match-mapping" and the CDA is set to "mapping-sent". For ECN, FL is set to 2 bits, TV is not set, MO is set to ignore and CDA is set to "value-sent".

ECN values are compressed according to the ecn_action value:

* If ecn_action is set to "not_compressed", the ECN field is included in the inner IP header. FL is set to 2 bits, TV is not set, MO is set to "ignore", CDA is set to "sent-value".

* If ecn_action is set to "lower", the ECN value is elided and the ECN value is copied in the outer IP header. FL is set to 2 bits, TV is not set, MO is set to "ignore", CDA is set to "lower".

Flow label is compressed according to the flow_label_action value:

* If flow_label_action is set to "not_compressed", the Flow label is included in the IPv6 header. FL is set to 20 bits, TV is not set, MO is set to "ignore", and CDA is set to "sent-value".

* If flow_label_action is set to "lower", the Flow Label is elided and read from the outer IP header (using compute-*(See Section 5)). FL is set to 20 bits, TV is not set, MO is set to "ignore", and CDA is set to "lower". If the outer IP header is an IPv4 header, only the 16 LSB of the Flow Label are inserted into the IPv4 header. At the decompression, the 4 MSB of the Flow Label are set to 0.

* If flow_label_action is set to "generated", the Flow Label is elided and the Flow Label is then re-generated (using compute-*) at the decompression (See Section 5). The resulting Flow Label differs from the initial value. FL is set to 20, TV is not set, MO is set to "ignore" and CDA is set to "generated".

* If flow_label_action is set to "zero", the Flow Label is elided and set to 0 at decompression. A 0 value indicates no flow label is present. Fl is set to 20 bits, TV is set to 0, MO is set to "equal" and CDA is set to "not-sent".

Payload Length is elided and determined from the Tunnel IP header Payload Length as well as the decompressed Payload. FL is set to 16 bits, TV is not set, MO is set to "ignore", CDA is set to "lower".

Next Header is compressed according to ts_proto:

* If ts_proto is the single value 0, Next Header is not compressed. FL is set to 8 bits, TV is not set, MO is set to "ignore", CDA is set to "sent-value".

* If ts_proto is a single non zero value, Next Header is compressed. FL is set to 8 bits, TV is set to ts_proto, MO is set to "equal" and CDA is set to "not-sent".

The IPv6 Hop Limit is read from the Tunnel IP header Hop Limit. FL is set to 8 bits, TV is not set, MO is set to "ignore" and CDA is set to "lower."

The source and destination IPv6 addresses are compressed using MSB. In both cases, FL is set to 128, TV is respectively set to prefix_bits(ts_ip_src_start, ts_ip_src_end) or prefix_bits(ts_ip_dst_start, ts_ip_dst_end), the MO is set to "MSB," and the CDA is set to "LSB."

6.1.3. Compression of the Inner IPv4 header

The fields Version, DSCP, ECN, Source Address and Destination Address are compressed as described for IPv6 in Section 6.1.2. The field Total Length (16 bits) is compressed similarly to the IPv6 field Payload Length. The field Identification (16 bits) is compressed similarly to the IPv6 field Flow Label. If the tunnel IP header is an IPv6 header, the Identification is placed as the LSB of the IPv6 header and the 4 remaining MSB are set to 0. The field Time to Live is compressed similarly to the IPv6 Hop Limit field. The Protocol field is compressed similarly to the last IPv6 Next Header field.

The Internet Header Length (IHL) is not compressed, FL is set to 4 bits, TV is not set, MO is set to ignore and CDA is set to "value-sent".

The IPv4 header checksum is elided. FL is set to 16, TV is omitted, MO is set to "ignore," and CDA is set to "checksum."

6.2. Clear Text ESP Compression (CTEC)

Once the Inner IP Packet has undergone compression as outlined in Section 6.1, the resulting compressed packet comprises a specific number of bytes. To construct the Clear Text ESP Packet, it is necessary to ascertain the ESP Payload Data, the Next Header, the ESP Pad Length, and the ESP Padding fields. (Refer to Figure 4, after post-processing.)

When esp_trailer is set to "Mandatory", the Next Header and the ESP Pad Length fields are present. Such requirement is usually expected to remain compatible with hardware implementations of ESP. The ESP Pad Length value is determined to meet the required alignment. When alignment is set to "8 bit", Pad Length is set to 0 and the Padding field is empty.

Conversely, when esp_trailer is set to "Optional", the Next Header, Pad Length, and Padding are generated as follows:

In transport mode, the IP header of the inner packet remains not compressed during the IIPC phase, and the ESP Payload Data is derived from the inner packet. Conversely, in tunnel mode, the ESP Payload Data encompasses the entirety of the packet generated by the IIPC.

In transport mode, the Next Header field is obtained from either the inner IP header or the Security Association, as specified in Section 6.1.3 or Section 6.1.2. In tunnel mode, the Next Header is elided, as it is determined by ts_ip_version. FL is set to 8 bit, TV is set to IPv4 or IPv6 depending on ts_ip_version, MO is set to "equal" and CDA is set to "not-sent".

The ESP Pad Length and ESP Padding fields are omitted only when ESP alignment has been selected to "8 bit" and esp_encr does not necessitate a specific block size alignment, or if that block size is one byte. This is represented by setting FL to (Pad Length + 1) * 8 bits, leaving TV unset, configuring MO to "ignore," and designating CDA as padding. The ESP Padding and ESP Pad Length may vary from their decompressed counterparts. However, since the IPsec process removes the padding, these variations do not affect packet processing. When esp_encr requires a specific block size, the ESP Pad Length and ESP Padding fields remain not compressed.

6.3. Encrypted ESP Compression (EEC)

Once the Clear Text Packet has undergone compression as outlined in Section 6.2, the resulting CTEC is encrypted. The header fields once the encrypted ESP packet is formed are the SPI and SN. To facilitate the processes of compression and decompression, this specification requires that the compressed ESP Header be byte-aligned. This requirement is satisfied by ensuring that the sum of esp_spi_lsb and esp_sn_lsb MUST be a multiple of 8.

SPI is compressed to its LSB. FL is set to 32 bits, TV is not set, MO is set to "MSB(4 - esp_spi_lsb)" and CDA is set to "LSB".

SN is compressed to its LSB, similarly to the SPI. FL is set to 32 bits, TV is not set, MO is set to "MSB(4 - esp_sn_lsb)" and CDA is set to "LSB".

7. Diet-ESP Compression for IPsec in Transport Mode

The transport mode mostly differs from the Tunnel mode in that the IP header of the packet is not encrypted. As a result, the IP Payload is compressed as described in Section 6.1.1. The IP header is not compressed. The byte alignment of the Compressed Payload is performed as described in Section 6.1. The Clear Text ESP Compression is performed as described in Section 6.2 except for the Next Header Field, which is compressed as described in Section 6.1.2.

8. IANA Considerations

We request the IANA to create a new registry for the IIPC Profile

IIPC Profile value	Reference
"iipc_not_compressed"	ThisRFC
"iipc_diet-esp"	ThisRFC

We request IANA to create the following registries for the "diet-esp" IIPC Profile.

Flow Label Action Value	Reference
"not_compressed"	ThisRFC
"generated"	ThisRFC
"lower"	ThisRFC
"zero"	ThisRFC

DSCP Action Value	Reference
"not_compressed"	ThisRFC
"lower"	ThisRFC
"sa"	ThisRFC

ECN Action Value	Reference
"not_compressed"	ThisRFC
"lower"	ThisRFC

Alignment	Reference
"8 bit"	ThisRFC
"16 bit"	ThisRFC
"32 bit"	ThisRFC
"64 bit"	ThisRFC

IPsec mode Value	Reference
"tunnel"	ThisRFC
"transport"	ThisRFC

9. Security Considerations

The security considerations encompass those outlined in ESP [RFC4303] as well as those pertaining to SCHC [RFC8724].

When employing ESP [RFC4303] in Tunnel Mode, the complete inner IP packet is safeguarded against man-in-the-middle attacks through cryptographic means, rendering it virtually impossible for an attacker to alter any fields associated with either the inner IP header or the inner IP payload. This level of protection is achieved by configuring the Flow Label CDA Value to "not_compressed", the DSCP CDA Value to either "not_compressed" or "sa", and the ECN CDA Value to "not_compressed".

However, this protection is compromised if the Flow Label CDA Value, DSCP CDA Value, or ECN CDA Values are set to "lower." In such cases, the values from the inner packet for the respective fields will be derived from the outer IP header, leaving these fields unprotected. It is important to note that even the Authentication Header [RFC4302] does not provide protection for these fields. When associated with a CDA value of "lower," the level of protection is akin to that provided in Transport mode. This vulnerability could be exploited by an attacker within an untrusted domain, potentially disrupting load balancing strategies that rely on the Flow Label [RFC6438][RFC6437]. More broadly, when the Flow Label CDA Value is set to "lower," the relevant Flow Label Security Considerations from [RFC6437] apply. Additionally, an attacker may manipulate the DSCP values to diminish the priority of specific packets, resulting in packets from the same flow having varying priorities, traversing different paths, and introducing additional latency to applications, thereby facilitating DDoS attacks. Consequently, these fields remain unprotected and are susceptible to modification during transmission, which may also be regarded as an acceptable risk.

When the Flow Label CDA Value is designated as "generated" or "zero," an attacker is unable to exploit the Flow Label field in any manner. The inner packet received is anticipated to possess a Flow Label distinct from that of the original encapsulated packet. However, the Flow Label is assigned by the receiving gateway. When the value is set to "zero," it is known, whereas when it is designated as "generated," it must be produced in accordance with the specifications outlined in [RFC6437].

The DSCP CDA Value is assigned as "sa" when DSCP values are linked to Security Associations (SAs), but it should not be utilized when all DSCP values are encompassed within a single SA. In such instances, "not_compressed" is recommended.

The encryption algorithm must adhere to the guidelines provided in [RFC8221] to guarantee contemporary cryptographic protection.

The least significant bits (LSB) of the ESP Security Parameter Index (SPI) determine the number of bits allocated to the SPI. An acceptable value for LSB must ensure that the peer possesses a sufficient number of SPIs, which is contingent upon the implementation of the SA lookup employed. If a peer relies solely on the SPI fields for SA lookup, then the number of LSB to consider must be sufficiently large to include the number of SPIs. A peer may compress its SPIs differently, in which case a incoming packet may have its SPI compressed to X bits while another packet may have its SPI compressed to Y bits. The operator must be cognizant that if multiple LSB values are permissible for each type of SA lookup, then multiple SA lookups and signature verifications may be required. It is advisable for a peer to ascertain the LSB associated with an incoming packet in a deterministic manner.

The ESP SN LSB must be established in a manner that allows the receiving peer to clearly ascertain the sequence number of the IPsec packet. If this requirement is not met, it will lead to an invalid signature verification, resulting in the rejection of the packet. Furthermore, the LSB should have the capacity to accommodate the maximum number of packets that may be in transit simultaneously. This approach will guarantee that the last packet received is correctly linked to the corresponding sequence number.

The ESP extension for IPv6 (and similarly for IPv4) requires a 64-bit (or 32-bit) alignment. Choosing alternative alignment values may result in a packet that fails to comply with this alignment requirement, potentially leading to rejection. The necessity for such alignment in IPv6 extensions arises from the fact that the length field in an IPv6 header extension is defined in terms of 64-bit words, making proper alignment essential for accurate packet parsing. Parsing of ESP does not present complications, as it is compatible with IPv6; the ESP extension is processed exclusively by the terminal IPsec peers and not by intermediary nodes. Furthermore, the ESP extension lacks a dedicated length field. Instead, its length is determined by the IPv6 header Length field, which is measured in bytes, along with the starting position of the ESP header extension. Consequently, it remains entirely feasible to parse an ESP extension that is not aligned to 64 bits. The same principle is applicable to IPv4.

10. Acknowledgements

We extend our gratitude to Laurent Toutain, Ana Carolina Minaburo, Pascal Thubert, and Alexandre Pelov for their guidance on SCHC and their valuable insights concerning the implementation of OpenSCHC [OpenSCHC]. Additionally, we express our appreciation to Robert Moskowitz for his inspiration in coining the term "Diet-ESP," derived from Diet-HIP, as well as to Samita Chakrabart, Tero Kivinen, Michael Richardson, and Valery Smyslov for their enduring support. The authors also wish to acknowledge the assistance provided by Mitacs through the Mitacs Accelerate program.

11. References

11.1. Normative References

[I-D.ietf-ipsecme-ikev2-diet-esp-extension]
Migault, D., Hatami, M., Liu, D., Preda, S., Atwood, J. W., Cespedes, S., Guggemos, T., and D. Schinazi, "Internet Key Exchange version 2 (IKEv2) extension for Header Compression Profile (HCP)", Work in Progress, Internet-Draft, draft-ietf-ipsecme-ikev2-diet-esp-extension-06, 21 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-ikev2-diet-esp-extension-06>.

[I-D.mglt-ipsecme-dscp-np]
Migault, D., Halpern, J. M., Preda, S., Liu, D., and U. Parkholm, "Differentiated Services Field Codepoints Internet Key Exchange version 2 Notification", Work in Progress, Internet-Draft, draft-mglt-ipsecme-dscp-np-04, 8 October 2025, <https://datatracker.ietf.org/doc/html/draft-mglt-ipsecme-dscp-np-04>.

[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>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <https://www.rfc-editor.org/info/rfc6437>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[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>.

[RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <https://www.rfc-editor.org/info/rfc8221>.

[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>.

[RFC8750] Migault, D., Guggemos, T., and Y. Nir, "Implicit Initialization Vector (IV) for Counter-Based Ciphers in Encapsulating Security Payload (ESP)", RFC 8750, DOI 10.17487/RFC8750, March 2020, <https://www.rfc-editor.org/info/rfc8750>.

11.2. Informative References

[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-05, 17 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-schc-architecture-05>.

[OpenSCHC] "OpenSCHC a Python open-source implementation of SCHC (Static Context Header Compression)", n.d., <https://github.com/openschc>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[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>.

[RFC9333] Migault, D. and T. Guggemos, "Minimal IP Encapsulating Security Payload (ESP)", RFC 9333, DOI 10.17487/RFC9333, January 2023, <https://www.rfc-editor.org/info/rfc9333>.

Appendix A. Examples of Diet-ESP

This appendix provides the details of the SCHC rules defined for Diet-ESP compression, alongside an explanation and illustrative examples for both Tunnel and Transport modes.

A.1. Tunnel Mode

This section provides a structured example of how Diet-ESP operates in Tunnel Mode. The example includes Attributes for Rule Derivation (AfRD), SCHC rules, the Inner IP packet (IIP), the compression process, and the decompression process.

A.1.1. Json Representation in Tunnel Mode

In Tunnel Mode, the full inner IP packet is compressed before ESP encapsulation, with each compression stage applying its own dedicated SCHC rule. The "iipc_diet-esp" profile activates the IIPC rule to compress the inner IPv6 and transport headers. The CTEC rule compresses ESP trailer-related fields such as padding and alignment. The EEC rule handles compression of the ESP SPI and Sequence Number, which are part of the outer ESP header and visible post-encryption.

The separation into three rules ensures modularity and clarity:

IIPC Rule: compresses source/destination IP addresses using MSB/LSB, compresses ports using LSB, and elides predictable fields like Next Header and Length.

CTEC Rule: elides padding and enforces alignment constraints.

EEC Rule: compresses ESP SPI and Sequence Number using LSB with an MSB match.

```
{
  "iipc_rule": {
    "RuleID": 1,
    "fields": [
      {
        "FID": "ts_ip_src_start",
        "FL": 128,
        "TV": "2001:db8::1234",
        "MO": "MSB",
        "CDA": "LSB"
      },
      {
        "FID": "ts_ip_dst_start",
        "FL": 128,
        "TV": "2001:db8::5678",
        "MO": "MSB",
        "CDA": "LSB"
      },
      {
        "FID": "IPv6.NextHeader",
        "FL": 8,
        "TV": 17,
        "MO": "equal",
        "CDA": "not-sent"
      },
      {
        "FID": "ts_port_src_start",
        "FL": 16,
        "TV": 5001,
        "MO": "MSB",
        "CDA": "LSB"
      },
      {
        "FID": "ts_port_dst_start",
        "FL": 16,
        "TV": 4500,
        "MO": "MSB",
        "CDA": "LSB"
      },
      {
        "FID": "UDP.Length",
        "FL": 16,
        "MO": "ignore",
        "CDA": "compute-length"
      },
```

```
        {
          "FID": "UDP.Checksum",
          "FL": 16,
          "MO": "ignore",
          "CDA": "checksum"
        }
      ]
    },

    "ctec_rule": {
      "RuleID": 2,
      "fields": [
        {
          "FID": "ESP.Padding",
          "FL": 8,
          "MO": "ignore",
          "CDA": "compute-padding"
        },
        {
          "FID": "alignment",
          "FL": 8,
          "TV": "64 bit",
          "MO": "equal",
          "CDA": "not-sent"
        }
      ]
    },

    "eec_rule": {
      "RuleID": 3,
      "fields": [
        {
          "FID": "esp_spi",
          "FL": 32,
          "MO": "MSB(24)",
          "CDA": "LSB"
        },
        {
          "FID": "esp_sn",
          "FL": 32,
          "MO": "MSB(24)",
          "CDA": "LSB"
        }
      ]
    }
  }
```

A.1.2. Attributes for Rule Derivation (AfRD)

The SCHC rules for Tunnel Mode are derived from the following AfRD:

* IPsec Mode: Tunnel
* Traffic Selector IP Version: IPv6-only
* Traffic Selector Source Address: 2001:db8::1234
* Traffic Selector Destination Address: 2001:db8::5678
* DSCP Action: Lower
* ECN Action: Lower
* ESP SPI Compression: LSB
* ESP SN Compression: LSB

A.1.3. Inner IP Packet (IIP)

The original packet (IIP), before compression consists of:

* IPv6 Source Address: 2001:db8::1234
* IPv6 Destination Address: 2001:db8::5678
* UDP Source Port: 5001
* UDP Destination Port: 4500
* UDP Length: 16 bytes
* Payload Data

A.1.4. Diet-ESP Compression

The rules for the IIPC, CTEC, and EEC layers are defined as IIPC to compress IPv6 headers and UDP headers, CTEC to compress ESP Trailer fields, and EEC to compress ESP SPI and Sequence Number. Each compressor applies the rule selected by the SA as follows:

1. IIPC: UDP Header Compression
    * UDP ports are compressed using the LSB technique.
    * UDP Length is removed (computed at decompression).

* UDP Checksum is omitted.

2. IIPC: IPv6 Header Compression

* Source and Destination Addresses are compressed using MSB.

* Next Header field is omitted.

3. CTEC: ESP Trailer Compression

* Pad Length and Padding are omitted.

* Next Header is omitted.

4. EEC: ESP Header Compression

* SPI and SN are compressed using LSB.

* Compressed Packet Output

The final compressed packet consists of an EEC packet with a compressed ESP header, a compressed IIP Header, and the payload.

A.1.5. Diet-ESP Decompression

The decompression process reverses these steps:

1. EEC: ESP Header Decompression

* SPI and SN are reconstructed using the LSB values.

2. CTEC: ESP Trailer Decompression (Optional)

3. IIPC: IPv6 Header Decompression

* ESP Next Header and Padding fields are restored.

* IPv6 Source and Destination Addresses are restored.

4. IIPC: UDP Header Decompression

* UDP ports are restored using the decompressed LSB values.

* UDP Length and Checksum are recalculated.

A.2. Transport Mode

This section follows the same structure as Tunnel Mode but applies to Transport Mode, where the IP header remains not compressed.

A.2.1. Json Representation in Transport Mode

In Transport Mode, the inner IP header is not compressed, and only the UDP header fields (within the inner payload) and the ESP headers are subject to compression. Following the correct SCHC rule structure, the compression logic is split across three rules:

The IIPC rule compresses UDP Source and Destination Ports using LSB and elides the Length and Checksum fields using compute-* and checksum CDAs.

The CTEC rule compresses ESP trailer fields such as Padding and Next Header, and ensures byte alignment.

The EEC rule compresses the ESP SPI and Sequence Number using the LSB approach.

Since the inner IP header is retained in Transport Mode, the IP Source and Destination Addresses are not compressed but instead transmitted in full.

```
{
  "iipc_rule": {
    "RuleID": 1,
    "fields": [
      {
        "FID": "ts_port_src_start",
        "FL": 16,
        "TV": 1234,
        "MO": "MSB",
        "CDA": "LSB"
      },
      {
        "FID": "ts_port_dst_start",
        "FL": 16,
        "TV": 5678,
        "MO": "MSB",
        "CDA": "LSB"
      },
      {
        "FID": "UDP.Length",
        "FL": 16,
        "MO": "ignore",
```

```
          "CDA": "compute-length"
        },
        {
          "FID": "UDP.Checksum",
          "FL": 16,
          "MO": "ignore",
          "CDA": "checksum"
        }
      ]
    },

    "ctec_rule": {
      "RuleID": 2,
      "fields": [
        {
          "FID": "ESP.Padding",
          "FL": 8,
          "MO": "ignore",
          "CDA": "compute-padding"
        },
        {
          "FID": "ESP.NextHeader",
          "FL": 8,
          "TV": 17,
          "MO": "equal",
          "CDA": "not-sent"
        },
        {
          "FID": "alignment",
          "FL": 8,
          "TV": "64 bit",
          "MO": "equal",
          "CDA": "not-sent"
        }
      ]
    },

    "eec_rule": {
      "RuleID": 3,
      "fields": [
        {
          "FID": "esp_spi",
          "FL": 32,
          "MO": "MSB(24)",
          "CDA": "LSB"
        },
        {
          "FID": "esp_sn",
```

```
          "FL": 32,
          "MO": "MSB(24)",
          "CDA": "LSB"
        }
      ]
    }
  }
```

A.2.2. Attributes for Rule Derivation (AfRD)

The SCHC rules for Transport Mode are derived from the following AfRD:

* IPsec Mode: Transport
* Traffic Selector IP Version: IPv6-only
* Traffic Selector Source Address: 2001:db8::1001
* Traffic Selector Destination Address: 2001:db8::2002
* DSCP CDA: Lower
* ECN CDA: Lower
* ESP SPI Compression: LSB
* ESP SN Compression: LSB

A.2.3. Inner IP Packet (IIP)

The original packet before compression consists of:

* IPv6 Source Address: 2001:db8::1001
* IPv6 Destination Address: 2001:db8::2002
* UDP Source Port: 1234
* UDP Destination Port: 5678
* UDP Length: 18 bytes
* ESP Payload Data

A.2.4. Diet-ESP Compression

1. IIPC: UDP Header Compression
    * UDP ports are compressed using the LSB technique.
    * UDP Length is removed (computed at decompression).
    * UDP Checksum is omitted.
2. CTEC: ESP Trailer Compression
    * Pad Length and Padding are omitted.
    * Next Header is omitted.
3. EEC: ESP Header Compression
    * SPI and SN are compressed using LSB.
4. Compressed Packet Output

   The final compressed packet consists of the compressed ESP header, IIPC compressed data, and payload.

A.2.5. Diet-ESP Decompression

The decompression process mirrors the compression steps, restoring SPI, SN, UDP headers, ESP Next Header, and Padding fields.

A.2.6. GitHub Repository: Diet-ESP SCHC Implementation

The source code for the implementation of the Diet-ESP profile, including the compression and decompression logic using the SCHC rules, is available on GitHub. Access the code at the following link:

GitHub Repository: Diet-ESP SCHC Implementation (https://github.com/mglt/pyesp/tree/master/examples/draft-diet-esp.py)

This repository contains the rule definitions, examples, and source code for implementing and testing the Diet-ESP profile. Refer to the README file for setup instructions and usage guidelines.

Authors' Addresses

Daniel Migault
Ericsson
Email: daniel.migault@ericsson.com

Maryam Hatami
Concordia University
Email: maryam.hatami@mail.concordia.ca

Sandra C迪spedes
Concordia University
Email: sandra.cespedes@concordia.ca

J. William Atwood
Concordia University
Email: william.atwood@concordia.ca

Daiying Liu
Ericsson
Email: harold.liu@ericsson.com

Tobias Guggemos
LMU
Email: guggemos@nm.ifi.lmu.de

Carsten Bormann
Universitaet Bremen TZI
Email: cabo@tzi.org

David Schinazi
Google LLC
Email: dschinazi.ietf@gmail.com