ipsecme
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

D. Migault
Ericsson
T. Guggemos
LMU Munich
C. Bormann
Universitaet Bremen TZI
D. Schinazi
Google LLC

March 11, 2019

ESP Header Compression and Diet-ESP
draft-mglt-ipsecme-diet-esp-07

Abstract

With the use of encrypted ESP for secure IP communication, the compression of IP payload is only possible with complex frameworks, such as RObust Header Compression (ROHC). Such frameworks are too complex for numerous use cases and especially for IoT scenarios, which makes IPsec not being used here, although it offers architectural benefits.

ESP Header Compression (EHC) defines a flexible framework to compress communications protected with IPsec/ESP. Compression and decompression is defined by EHC Rules orchestrated by EHC Strategies. The necessary state is hold within the IPsec Security Association and can be negotiated during key agreement, e.g. with IKEv2.

The document specifies the necessary parameters of the EHC Context to allow compression of ESP and the most common included protocols, such as IPv4, IPv6, UDP and TCP and the corresponding EHC Rules. It also defines the Diet-ESP EHC Strategy which compresses up to 32 bytes per packet for traditional IPv6 VPN and up to 66 bytes for IPv6 VPN sent over a single TCP or UDP session.

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 September 12, 2019.

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

IPsec/ESP [RFC4303] secures communications either using end-to-end security or by building a VPN, where the traffic is carried to a secure domain via a security gateway.

IPsec/ESP was not designed to minimize its associated networking overhead. In fact, bandwidth optimization often adds computational overhead that may negatively impact large infrastructures in which bandwidth usage is not a constraint. On the other hand, in IoT communications, sending extra bytes can significantly impact the battery life of devices and thus the life time of the device. The document describes a framework that optimizes the networking overhead associated to IPsec/ESP for these devices.

Most compression mechanisms work with dynamic compression contexts. Some mechanisms, such as ROHC, agree and dynamically change the context over a dedicated channel. Others, such as 6LowPAN, send the context together with the actual protocol information in a separate compression header. Those mechanism fail when it comes to compress encrypted payloads as appearing in ESP. This is found to be a major reason, why IPsec and in particular ESP is not widely developed in environments where bandwidth saving is a critical task, such as in IoT scenarios.

ESP Header Compression (EHC) chooses another form of context agreement, which is similar to the one defined by Static Context Header Compression (SCHC). It works with a static compression context agreed for a specific Security Association. The context itself can be negotiated during the key agreement, which allows only minimal the changes to the actual ESP implementation.

EHC itself is defined as a framework that specifically compresses ESP protected communications. EHC is highly flexible to address any use

case where compression is necessary. EHC takes advantage of the negotiation between the communication endpoint to agree on the cryptographic parameters, which in some cases already includes parameters that remain constant during the communications (like layer 4 ports, or IP addresses) and can thus be used as part of the compression context. Only additional, EHC specific parameters need to be agreed for the purpose of compression. In addition EHC Rules define how fields may be compressed and decompressed given the provided parameters. Finally, EHC defines EHC Strategy which defines how a set of EHC Rule is coordinated.

This document specifies EHC Context parameters for the most common Layer 3 and 4 protocols and the associated EHC Rules. Additionally, an EHC Strategy called Diet-ESP is defined, which compresses up to 32 bytes per packet for traditional VPN and up to 66 bytes for VPN set over a single TCP or UDP session. Its main purpose is a maximum level of compression with a minimum of additional agreement. This is achieved by defining a default usage of existing IPsec SA parameters wherever possible.

3. Terminology

This document uses the following terminology:

- EHC ESP Header Compression
- IoT Internet of Things
- IP If not stated otherwise, IP means IPv6.
- LSB Least Significant Bytes
- MSB Most Significant Bytes
- SAD IPsec Security Association Database
- SA IPsec Security Association
- SPD IPsec Security Policy Database
- TS IPsec Traffic Selector
- SPI ESP Security Parameter Index
- SN ESP Sequence Number
- PAD ESP Padding
- PL ESP Pad Length
- NH Next Header
- IV Initialization Vector
- IIV Implicit Initialization Vector
- ICV Integrity Check Value
- VPN Virtual Private Network

4. Protocol Overview

ESP Header Compression (EHC) compresses IPsec ESP packets, thus reducing the size of the packet sent on the wire, while carrying an equivalent level of information with an equivalent level of security.

EHC is able to compress any protocol encapsulated in ESP and ESP itself. Concerned fields include those of the ESP protocol, as well as other protocols in the ESP payload such as the IP header when the tunnel mode is used, but also upper layer protocols, such as the UDP or the TCP header. Non ESP fields may be compressed by ESP under certain circumstances, but EHC is not intended to provide a generic way outside of ESP to compress these protocols. Compression of the unprotected IP header and the unencrypted ESP header may be performed by mechanism such as 6LoWPAN [RFC4944], SCHC [I-D.toutain-6lpwa-ipv6-static-context-hc], ROHC [RFC5795] or 6LoWPAN-GHC [RFC7400].

EHC is based on a static compression context, EHC Rules coordinated by an EHC Strategy:

EHC Context: Stores the information of a specific header field which can be compressed by EHC. This can be specific header values such as IP addresses or L4 ports do not have to be send on the wire at all, or compression information for fields which can be partially compressed, such as sequence numbers.

EHC Rules: Defines how the information of the EHC Context is used to compress a specific field. It defines compression functions, such as "elided", "least significant byte" and others, being applied on the header field.

EHC Strategy: Is applied to efficiently coordinate EHC Context and EHC Strategy. The EHC Strategy "Diet-ESP" defined in this document utilizes the information in the IPsec SA to pre-define the EHC Context without explicitly exchanging the EHC Context.

As depicted in Figure 1, the EHC Strategy - Diet-ESP in our case - and the EHC Context are agreed upon between the two peers, e.g. during key exchange. The EHC Rules are to be implemented on the peers and do not require further agreement.

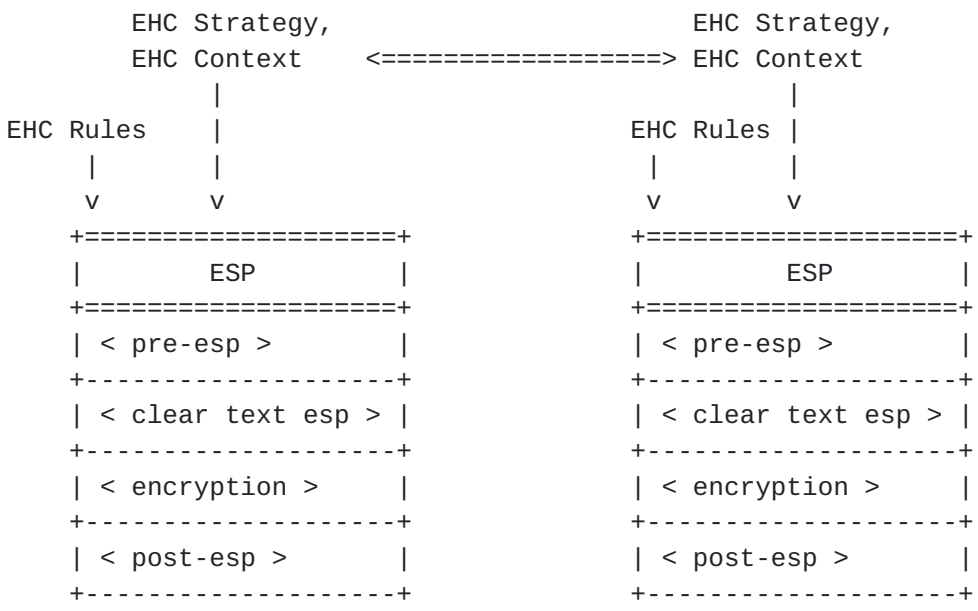

Figure 1: ESP Header Compression Overview

In Figure 1, the ESP stack is represented by various sub layers describing the packet processing inside the ESP:

pre-esp: represents treatment performed to a non ESP packet, i.e. before ESP encapsulation or decapsulation is being performed. Any compression of protocols not specific to but encrypted by ESP, such as L4 and higher protocols, is performed here.

clear text esp: designates the ESP encapsulation / decapsulation processing performed on an non encrypted ESP packet. This layer includes compression for fields which are included during the ESP encapsulation. A typical example is the later encrypted Tunnel IP header and the fields of the ESP trailer.

encryption: designates the encryption/decryption phase This layer could include compression of encryption information (e.g. Initialization Vector, etc.), but this is currently out of scope of this document.

post-esp the processing performed on an ESP encrypted packet. This layer includes compression of the ESP header.

EHC Rules may be processed at any of these layers and thus impact differently the standard ESP. More specifically, EHC Rules performed at the "pre-esp" or "post-esp" layer do not require the current ESP stack to be updated and can simply be appended to the current ESP stack. On the other hand, EHC Rules at the "clear text esp" may require modification of the current ESP stack.

The set of EHC rules described in this document as well as the EHC Strategies may be extended in the future. Nothing prevents such EHC Rules and Strategies to be updated.

5. IPsec Compression Mode

Signalling the compression of a certain ESP packet is crucial for correct decompression at the sender. Situation where decompression may fail unforeseen are various, such as IP fragmentation, UDP options [I-D.ietf-tsvwg-udp-options] just to name a few.

With EHC, the agreement of the level or occurrence of compression is left the negotiation protocol (e.g. IKEv2), contradicting the signalization of the level of compression for a certain packet send over the wire. In order to achieve per-packet signalization of the compression level, this document proposes new IPsec modes "Compressed Transport" and "Compressed Tunnel", which are meant to be agreed during the negotiation of the EHC Contex and EHC Strategy. This leads to multiple SAs, and thus, multiple SPIs for different levels of compression agreed with the EHC Context. The receiver can detect the level of compression of an incoming packet by looking up the used EHC Context and EHC strategy in the corresponding SA.

If the sender detects the de-compression can not be guaranteed with a given EHC Context and EHC Strategy, it MUST NOT apply compression. If an SA with IPsec Mode "Tunnel"/"Transport" is available, the sender SHOULD send the packet uncompressed, rather than discard the packet. When there is no uncompressed SA available, the packet MUST be dropped.

6. EHC Context

The EHC Context provides the necessary information so the two peers can proceed to the appropriated compression and decompression defined by the EHC Strategy.

The EHC Context is defined on a per-SA basis. A context can be defined for any protocol encapsulated with ESP and for ESP itself. For each header field, a context attribute is provided to the EHC Context in order to allow compression and decompression. Most power of EHC lies in the fact, that the attributes for some protocols are already available in the IPsec SA (e.g. IP addresses in the Traffic Selector). Such attributes are designated by "Yes" in the "In SA" column. All others need to be negotiated separately in order to allow EHC to work properly.

As this document is limited to the Diet-ESP strategy, the EHC Context in this section used by the Diet-ESP Strategy to activate specific

EHC Rules as well as to execute the EHC Rules by providing the necessary parameters..

6.1. EHC Context Parameters for ESP

Context Attribute	In SA	Possible Values
ipsec_mode	Yes	"Tunnel", "Transport"
outer_version	Yes	"IPv4", "IPv6"
esp_spi	Yes	ESP SPI
esp_spi_lsb	No	0, 1, 2, 3, 4
esp_sn	Yes	ESP Sequence Number
esp_sn_lsb	No	0, 1, 2, 3, 4
esp_sn_gen	No	"Time", "Incremental"
esp_align	No	8, 16, 24, 32
esp_encr	Yes	ESP Encryption Algorithm

6.2. EHC Context Parameters for Inner IP

Parameters associated to the Inner IP addresses are only specified when the SA has been configured with the tunnel mode. As a result when ipsec_mode is set to "Transport" the parameters below MUST NOT be considered and are considered as "Undefined"

Context Attribute	In SA	Possible Values
ip_version	Yes	"IPv4", "IPv6"

6.2.1. EHC Context Parameters for inner IPv6

Context Attribute	In SA	Possible Values
ip6_tcfl_comp	No	"Outer", "Value", "UnComp"
ip6_tc	No	IPv6 Traffic Class
ip6_fl	No	IPv6 Flow Label
ip6_hl_comp	No	"Outer", "Value", "UnComp"
ip6_hl	No	Hop Limit Value
ip6_src	Yes	IPv6 Source Address
ip6_dst	Yes	IPv6 Destination Address

ip6_tcfl_comp indicates how Traffic Class and Flow Label fields of the inner IP Header are expected to be compressed. "Outer" indicates

Traffic Class and Flow Label are read from the outer IP header, "Value" indicates these values are provided by the Diet-ESP Context, while "Uncompress" indicates that no compression occurs and these values are read in the inner IP inner header.

ip6_hl_comp indicates how Hop Limit field of the inner IP Header is expected to be compressed. (see ip6_tcfl_comp).

ip6_dst designates the Destination IPv6 Address of the inner IP header. The IP address is provided by the TS, and can be defined as a range of IP addresses. Compression is only considered when ip6_dst indicates a single IP Address. When the TS defines more than a single IP address ip6_dst is considered as "Unspecified" and its value MUST NOT be considered for compression.

6.2.2. EHC Context Parameters for inner IPv4

Context Attribute	In SA	Possible Values
ip4_options	No	"Options", "No_Options"
ip4_id	No	IPv4 Identification
ip4_id_lsb	No	0,1,2
ip4_ttl_comp	No	"Outer", "Value", "UnComp"
ip4_ttl	No	IPv4 Time To Live
ip4_src	Yes	IPv4 Source Address
ip4_dst	Yes	IPv4 Destination Address
ip4_frag_enable	No	"True", "False"

ip4_options specifies if the IPv4 header contains any options. If set to "No_Options", the first 8 bit of the IPv4 header (being the IP version and IP header length) are compressed. If set to "Options" this bits are sent uncompressed.

ip4_ttl indicates how the Time To Live field of the inner IP Header is expected to be compressed. (see ip6_hl_comp).

6.3. EHC Context Parameters for Transport Protocol

The following parameters are provided by the SA but the SA may specify single value or a range of values. When the SA specifies a range of values, these parameters MUST NOT be considered and are considered as Unspecified.

Context Attribute	In SA	Possible Values
l4_proto	Yes	IPv6/ESP Next Header,IPv4 Protocol
l4_src	Yes	UDP/UDP-Lite/TCP Source Port
l4_dst	Yes	UDP/UDP-Lite/TCP Destination Port

6.3.1. EHC Context Parameters for UDP

For UDP, there are no additional parameters necessary than the ones in Section 6.3

6.3.2. EHC Context Parameters for UDP-Lite

Context Attribute	In SA	Possible Values
udplite_coverage	No	8-6535, "Length", "uncompressed"

udplite_coverage: For UDP-Lite, the checksum can have different coverages, which is defined by the "Checksum Coverage" field which replaces the "Length" field of UDP. This context field defines the coverage in advance by either a specific value (8-16535), the actual length of the UDP-Lite payload ("Length" or 0) or as uncompressed. Note that udplite coverage is indicated on a packet basis and cannot be greater than the UDP length. In this case udplite_coverage is negotiated for all packets and the actual coverage for a given UDP packet is derived as the minimum value between udplite_coverage and the length of the UDP packet.

6.3.3. EHC Context Parameters for TCP

Context Attribute	In SA	Possible Values
tcp_sn	No	TCP Sequence Number
tcp_ack	No	TCP Acknowledgment Number
tcp_lsb	No	0, 1, 2, 3, 4
tcp_options	No	"True", "False"
tcp_urgent	No	"True", "False"

tcp_sn holds the current Sequence Number of the TCP session.

tcp_ack holds the current Acknowledgement Number of the TCP session.

tcp_lsb holds the number of lsb of tcp_sn and tcp_ack sent on the wire.

tcp_options says if options are enabled in the current TCP session. If tcp_options is set to "False" the Options field in TCP can be elided.

tcp_urgent says if the urgent pointer is enabled in the current TCP session. If tcp_urgent is set to "False" the Urgent Pointer field in TCP can be elided.

7. EHC Rules

This section describes the EHC Rules involved in Diet-ESP. The EHC Rules defined by Diet-ESP may be used in the future by EHC Strategies other than Diet-ESP, so they are described in an independent way.

A EHC Rule defines the compression and decompression of one or more fields and EHC Rules are represented this way:

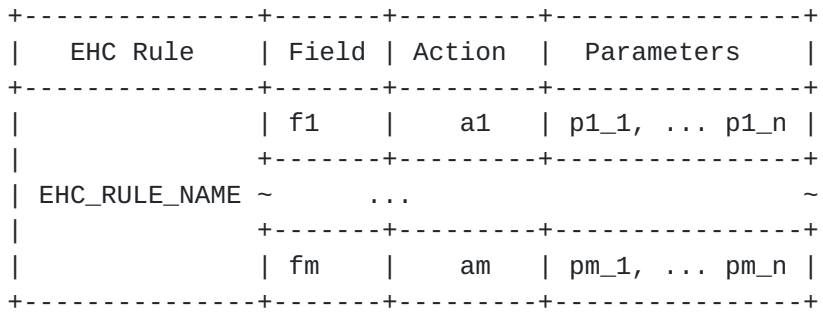

Figure 2: EHC Rules

The EHC Rule is designated by a name (EHC_RULE_NAME) and the concerned Fields (f1, ..., fm). Each field compression and decompression is represented by an Action (a1, ..., am). The Parameters indicate the necessary parameters for the action to perform both the compression and the decompression.

The table below provides a high level description of the Actions used by Diet-ESP. As these Action may take different arguments and may operate differently for each field a compete description is provided in the next sections as part of the EHC Rule description.

Function	Compression	Decompression
send-value	No	No
elided	Not send	Get from EHC Context
lsb(_lsb_size)	Sent LSB	Get from EHC Context
lower	Not send	Get from lower layer
checksum	Not send	Compute checksum.
padding(_align)	Compute padding	Get padding

a. send-value designates an action that does not perform any compression or decompression of a field.
b. elided designates an action where both peers have a local value of the field. The compression of the field consists in removing the field, and the decompression consists in retrieving the field value from a known local value. The local value may be stored in a EHC Context or defined by the EHC Rule (like a zero value for example).
c. lsb designates an action where both peers have a local value of the field, but the compression consists in sending only the LSB bytes instead of the whole field. The decompression consists in retrieving the field from the LSB sent as well as some other additional local values.
d. lower designates an action where the compression consists in not sending the field. The decompression consists in retrieving the field from the lower layers of the packet. A typical example is when both IP and UDP carry the length of the payload, then the length of the UDP payload can be inferred from the one of the IP layer.
e. checksum designates an action where the compression consists in not sending a checksum field. The decompression consists in re-computing the checksum. ESP provides an integrity-check based on signature of the ESP payload (ICV). This makes removing checksum possible, without harming the checksum mechanism.
f. padding designates an action that computes the padding of the ESP packet. The function is specific to the ESP.

For all actions, the function can be performed only when the appropriated parameters and fields are provided. When a field or a parameters does not have an appropriated value its value is designated as "Unspecified". Specifically some fields such as inner IP addresses, ports or transport protocols are agreed during the SA negotiation and are specified by the SA. Their value in the SA may take various values that are not appropriated to enable a compression. For example, when these fields are defined as a range of values, or by selectors such as OPAQUE or ANY these fields cannot be retrieved from a local value. Instead, when they are defined as a

"Single" value (i.e a single IP address, or a single port number or a single transport protocol number) compression and decompression can be performed. These SA related fields are considered as "Unspecified" when not limited to a "Single" value.

When a field or a parameter is "Unspecified", the EHC Rule MUST NOT be activated. This is the purpose of the EHC Strategy to avoid ending in such case. In any case, when one of these condition is not met, the EHC Rule MUST NOT perform any compression or decompression action and the packet MUST be discarded. When possible, an error SHOULD be raised and logged.

7.1. EHC Rules for ESP

This section describes the EHC Rules for ESP which are summed up in the table below.

EHC Rule	Field	Action	Parameters
ESP_SPI	SPI	lsb	esp_spi_lsb, esp_spi
ESP_SN	Sequence Number	lsb	esp_sn_lsb, esp_sn_gen, esp_sn
ESP_NH	Next Header	elided	l4_proto, ipsec_mode
ESP_PAD	Pad Length, Padding	padding	esp_align, esp_encr

ESP_SPI designates the EHC Rule compressing / decompressing the SPI. ESP_SPI is performed in the "post-esp" phase. The SPI is compressed using "lsb". The sending peer only places the LSB bytes of the SPI and the receiving peer retrieve the SPI from the LSB bytes carried in the packets as well as from the SPI value stored in the SA. The SPI MUST be retrieved as its full value is included in the signature check. The two peers MUST agree on the number of LSB bytes to be sent: "esp_spi_lsb". Upon agreeing on "esp_spi_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full SPI.

ESP_SN designates the EHC Rule compressing / decompressing the ESP Sequence Number. ESP_SN is performed in the "post-esp" phase. ESP_SN is only activated if the SN ("esp_sn"), the LSB significant bytes ("esp_sn_lsb") and the method used to generate the SN ("esp_sn_gen") are defined. The Sequence Number is compressed using "lsb". Similarly to the SPI, the Sequence Number MUST be retrieved in order to complete the signature check of the ESP packet. Unlike the SPI, the Sequence Number is not agreed by the peers, but is

changing for every packet. As a result, in order to retrieve the Sequence Number from the LSB "esp_sn_lsb", the peers MUST agree on generating Sequence Number in a similar way. This is negotiated with "esp_sn_gen" and the receiver MUST ensure that "esp_sn_lsb" is big enough to absorb minor packet losses or time differences between the peers.

ESP_NH designates the EHC Rule compressing / decompressing the ESP Next Header. ESP_NH is performed in the "clear text esp" phase. ESP_NH is only activated if the Next Header is specified. The Next Header can be specified as IP (IPv4 or IPv6) when the IPsec tunnel mode is used ("ipsec_mode" set to "Tunnel") or when the transport mode ("ipsec_mode" set to "Transport") is used when the Traffic Selector defines a "Single" Protocol ID ("l4_proto"). The Next Header, is compressed using "elided". The Next Header indicates the Header in the Payload Data. When the Tunnel mode is chosen, the type of the header is known to be an IP header. Similarly, the TS may also hold transport layer protocol, which specifies the Next Header value for Transport mode. The Next Header value is only there to provide sufficient information for decapsulating ESP. In other words decompressing this fields would occur in the "clear text esp" phase and striped but directly removed again by the ESP stack. For these reasons, implementation may simply omit decompressing this field.

ESP_PAD designates the EHC Rule compressing / decompressing the Pad Length and Padding fields. ESP_PAD is performed in the "clear text esp" phase. Pad Length and Padding define the padding. The purpose of padding is to respect a 32 bit alignment for ESP or block sizes of the used cryptographic suite. As the ESP trailer is encrypted, Padding and Pad Length MUST to be performed by ESP and not by the encryption algorithm. Thus, ESP_PAD always needs to respect the cipher alignment ("esp_encr"), if applicable. Compression may be performed especially when device support alignment smaller than 32 bit. Such alignment is designated as "esp_align" and the padding bytes are the necessary bytes so the ESP packet has a length that is a multiple of "esp_align".

When "esp_align" is set to an 8-bit alignment padding bytes are not necessary, and Padding as well as Pad Length are removed. For values that are different from 8-bit alignment, padding bytes needs to be computed according to the ESP packet length why ESP_PAD MUST be the last action of "clear text esp". The resulting number of padding byte is then expressed in Padding and Pad Length fields with Pad Length set to padding bytes number - 1 and Padding is generated as described in [RFC4303].

Combining the Pad Length and Padding fields could potentially add an overhead on fixed size padding. In fact some applications may only

send the same type of fixed size data, in which case the Pad Length would not be necessary to be specified. However, the only corner case Pad Length fields would actually add an overhead is when padding is expected to be of zero size. In this case, specifying an 8-bit alignment solve this issue.

7.2. EHC Rules for inner IPv4

All IPv4 EHC Rules MUST be performed during the "clear text esp" phase. The EHC Rules are only defined for compressing the inner IPv4 header and thus can only be used when the SA is using the Tunnel mode.

EHC Rule	Field	Action	Parameters
IP4_OPT_DIS	Version	elided	ip_version
	Header Length	elided	
IP4_LENGTH	Total Length	lower	
IP4_ID	Identification	lsb	ip4_id, ip4_id_lsb
IP4_FRAG_DIS	Flags	elided	
	Fragment Offset	elided	
IP4_TTL_OUTER	Time To Live	elided	ip4_ttl
IP4_TTL_VALUE	Time To Live	elided	ip4_ttl
IP4_PROT	Protocol	elided	l4_proto
IP4_CHECK	Header Checksum	checksum	
IP4_SRC	Source Address	elided	ip4_src
IP4_DST	Dest. Address	elided	ip4_dst

IP4_OPT_DIS designates that the IPv4 header does not include any options and indicates if the first byte of the IPv4 header - consisting of IP version and IPv4 Header Length, are compressed. The Version "ip_version" is defined by the SA and is thus compressed using "elided". If the header does not contain any options, it is compressed with "elided" and decompressed to "20", the default length of the IPv4 header. If the header does contains some options, the length is not compressed.

IP4_LENGTH designates the EHC Rule compressing / decompressing the Total Length Field of the inner IPv4 header. The Total Length is compressed by the sender and not sent. The receiver decompresses it by recomputing the Total Length from the outer IP header. The outer IP header can be IPv4 or IPv6 and IP4_LENGTH MUST support both versions if both versions are supported by the device. Note that the length of the inner IP payload may also be subject to updates if decompression of the upper layers occurs.

IP4_ID designates the EHC Rule compressing / decompressing the Identification Field. IP4_ID is only activated if the ID ("ip4_id"), the LSB significant bytes ("ip4_id_lsb") are defined. Upon agreeing on "ip4_id_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full IP Identification. Note also that unlike the ESP SN, the IPv4 Identification is not part of the SA. As a result, when the ID is compressed, its value MUST be stored in the EHC Context. The reserved attribute for that is "ip4_id"

IP4_FRAG_DIS designates that the inner IPv4 header does not support fragmentation. If activated, IP4_FRAG_DIS indicates compression of Flags and Fragment Offset field in the IPv4 header which consists of 2 bytes. Both fields are compressed with "elided" and decompressed with their default value according to [RFC0791], which is 0b010 for Flags and 0 for Fragment Offset.

IP4_TTL_OUTER designates an EHC Rule compressing / decompressing the Time To Live field of the inner IP header. If the outer IP header is an IPv6 header, the Hop Limit is used for decompression. The Time To Live field is compressed / decompressed using "lower", thus the field is not sent. The receiver decompresses it by reading its value from the outer IP header (TTL in case of IPv4 or HL in case of IPv6).

IP4_TTL_VALUE designates an EHC Rule compressing / decompressing the Time To Live field of the inner IP header. IP4_TTL_VALUE is only activated when the Hop Limit ("ip4_ttl") has been agreed. Time To Live is compressed / decompressed using the "elided" method.

IP4_PROTO designates the EHC Rule compressing / decompressing the Protocol field of the inner IPv4 header. IP4_PROTO is only activated if the Protocol is specified, that is when the Traffic Selectors defines a "Single" Protocol ID ("l4_proto"). When the Protocol ID identified by the SA has a "Single" value, the Protocol is compressed and decompressed using the "elided" method.

IP4_CHECK designates the EHC rule compressing / decompressing the Header Checksum field of the inner IPv4 header. The IPv4 header checksum is not sent by the sender and the receiver computes from the decompressed inner IPv4 header. IP4_CHECK MUST compute the checksum and not fill the checksum field with zeros. As a result, IP4_CHECK is the last decompressing EHC Rule to be performed on the decompressed IPv4 header.

IP4_SRC compresses the source IP address of the inner IPv4 header. IP4_SRC_IP is only be activated when the Traffic Selectors agreed by the SA defines a "Single" source IP address ("ip4_src"). The Source IP address is compressed / decompressed using the "elided" method.

IP4_DST works in a similar way as IP4_SRC_IP but for the destination IP address ("ip4_dst")

7.3. EHC Rules for inner IPv6

All IPv6 EHC Rules MUST be performed during the "clear text esp" phase. The EHC Rules are only defined for compressing the inner IPv6 header and thus can only be used when the SA is using the Tunnel mode.

EHC Rule	Field	Action	Parameters
IP6_OUTER	Version	elided	ip_version
	Traffic Class	lower	
	Flow Label	lower	
IP6_VALUE	Version	elided	ip_version
	Traffic Class	elided	ip6_tc
	Flow Label	elided	ip6_fl
IP6_LENGTH	Payload Length	lower	
IP6_NH	Next Header	elided	l4_proto
IP6_HL_OUTER	Hop Limit	lower	
IP6_HL_VALUE	Hop Limit	elided	ip6_hl
IP6_SRC	Source Address	elided	ip6_src
IP6_DST	Dest. Address	elided	ip6_dst

IP6_OUTER designates an EHC Rule for compressing / decompressing the first 32 bits of the inner IPv6 header formed by the Version, Traffic Class and Flow Label. IP6_OUTER only proceeds to compression when both the outer and inner IP header are IPv6 header. When the outer IP header is an IPv4, the compression is bypassed. Bypassing enables to proceed to compression of IPv4 and IPv6 traffic in a VPN use case with a single SA. The Version "ip_version" is defined by the SA and is thus compressed using "elided". The other parameters Traffic Class and Flow Label are compressed using "lower". More specifically, the fields are not sent. The receiver decompresses them by reading their value from the outer IPv6 header.

IP6_VALUE designates an EHC Rule for compressing / decompressing the first 32 bits of the inner IPv6 header formed by the Version, Traffic Class and Flow Label. IP6_VALUE is only activated if the Version of the inner IP header agreed by the SA is set to "Version 6" ("ip_version" set to "Version 6") and the specific values of the Traffic Class ("ip6_tc") and the Flow Label ("ip6_fl") are specified. With IP6_VALUE all fields are compressed and decompressed using "elided". Version is provided by the SA ("ip_version") while other fields are explicitly provided (ip6_tc, ip6_fl.

IP6_LENGTH designates the EHC Rule compressing / decompressing the Payload Length Field of the inner IPv6 header. The Payload Length is compressed by the sender and is not sent. The receiver decompress it by recomputing the Payload Length from the outer IP header. The IP header can be IPv4 or IPv6 and IP6_LENGTH MUST support both versions if both versions are supported by the device. Note that the length of the inner IP payload may also be subject to updates if decompression of the upper layers occurs.

IP6_NH designates the EHC Rule compressing / decompressing the Next Header field of the inner IPv6 header. IP6_NH is only activated if the Next Header is specified, that is when the Traffic Selectors defines a "Single" Protocol ID ("l4_proto"). When the Protocol ID identified by the SA has a "Single" value, the Next Header is compressed and decompressed using the "elided" method.

IP6_HL_OUTER designates an EHC Rule compressing / decompressing the Hop Limit field of the inner IP header. If the outer IP header is an IPv4 header, the Time To Live is used for decompression. The Hop Limit field is compressed / decompressed using the "lower". More specifically, the fields are not sent. The receiver decompresses them by reading their value from the outer IPv6 header.

IP6_HL_VALUE designates an EHC Rule compressing / decompressing the Hop Limit field of the inner IP header. IP6_HL_VALUE is only activated when the Hop Limit ("ip6_hl") has been agreed. The Hop Limit is compressed / decompressed using the "elided" method.

IP6_SRC compresses the source IP address of the inner IP header. IP6_SRC_IP is only be activated when the Traffic Selectors agreed by the SA defines a "Single" source IP address ("ip6_src"). The Source IP address is compressed / decompressed using the "elided" method.

IP6_DST works in a similar way as IP6_SRC_IP but for the destination IP address ("ip6_dst")

7.4. EHC Rules for UDP

All UDP EHC Rules MUST be performed during the "pre-esp" phase. The EHC Rules are only defined when the Traffic Selectors agreed during the SA negotiation results in "Single" Protocol ID ("l4_proto") which is set to UDP (17).

EHC Rule	Field	Action	Parameters
UDP_SRC	Source Port	elided	l4_source
UDP_DST	Dest. Port	elided	l4_dest
UDP_LENGTH	Length	lower	
UDP_CHECK	UDP Checksum	checksum	

UDP_SRC designates the EHC Rule that compresses / decompresses the UDP Source Port. UDP_SRC is only activated when the Source Port agreed by the SA negotiation ("l4_src") is "Single". The Source Port is then compressed / decompressed using the "elided" method.

UDP_DST works in a similar way as UDP_SRC but for the Destination Port ("l4_dst").

UDP_LENGTH designates the EHC Rule compressing / decompressing the Length Field of the UDP header. The length is compressed by the sender and is not sent. The receiver decompresses it by recomputing the Length from the IP address header. The IP address can be IPv4 or IPv6 and UDP_LENGTH MUST support both versions if both versions are supported by the device.

UDP_CHECK designates the EHC Rule compressing / decompressing the UDP Checksum. The UDP Checksum is not sent by the sender and the receiver computes from the decompressed UDP payload. UDP_CHECK MUST compute the checksum and not fill the checksum field with zeros. As a result, UDP_CHECK is the last decompressing EHC Rule to be performed on the decompressed UDP Payload.

7.5. EHC Rules for UDP-Lite

All UDP-lite EHC Rules MUST be performed during the "pre-esp" phase. The EHC Rules are only defined when the Traffic Selectors agreed during the SA negotiation results in a "Single" Protocol ID ("l4_proto") which is set to UDPLite (136).

EHC Rule	Field	Action	Parameters
UDP-LITE_SRC	Source Port	elided	l4_source
UDP-LITE_DST	Dest. Port	elided	l4_dest
UDP-LITE_COVERAGE	Checksum Coverage	elided	udplite_coverage
UDP-LITE_CHECK	UDP-Lite Checksum	checksum	

UDP-LITE_SRC works similarly to UDP_SRC

UDP-LITE_DST works similarly to UDP_DST

UDP-LITE_COVERAGE designates the EHC Rule compressing / decompressing the UDP-Lite Coverage field. UDP-LITE_COVERAGE is only activated when the Coverage ("udplite_coverage") has been agreed with a valid value. The Coverage is compressed / decompressed using the "elided" method.

UDP-LITE_CHECK designates the EHC Rule compressing / decompressing the UDP-Lite checksum. UDP-LITE_CHECK is only activated if the Coverage is defined either elided or sent. UDP-LITE_CHECK computes the checksum using "checksum" according to the uncompressed UDP packet and the value of the Coverage.

7.6. EHC Rules for TCP

All TCP EHC Rules MUST be performed during the "pre-esp" phase. The EHC Rules are only defined when the Traffic Selectors agreed during the SA negotiation results in a"Single" Protocol ID ("l4_proto") which is set to TCP (6).

EHC Rule	Field	Action	Parameters
TCP_SRC	Source Port	elided	l4_source
TCP_DST	Dest. Port	elided	l4_dest
TCP_SN	Sequence Number	lsb	tcp_sn, tcp_lsb
TCP_ACK	Acknowledgment Number	lsb	tcp_ack, tcp_lsb
TCP_OPTIONS	Data Offset	elided	tcp_options
	Reserved Bits	elided	
TCP_CHECK	TCP Checksum	checksum	
TCP_URGENT	TCP Urgent Field	elided	tcp_urgent

TCP_SRC works similarly to UDP_SRC.

TCP_DST works similarly to UDP_DST.

TCP_SN designates the EHC Rule compressing / decompressing the TCP Sequence Number. TCP_SN is only activated if the SN ("tcp_sn") and the LSB significant bytes ("tcp_lsb") are defined. The TCP SN is compressed using "lsb". The sending peer only places the LSB bytes of the TCP SN ("tcp_sn") and the receiving peer retrieve the TCP SN from the LSB bytes carried in the packets as well as from the TCP SN value stored in EHC Context ("tcp_sn"). The two peers MUST agree on the number of LSB bytes to be sent: "tcp_lsb". Upon agreeing on

"tcp_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full TCP SN. Note also that unlike the ESP SN, the TCP SN is not part of the SA. As a result, when the SN is compressed, the value of the TCP SN MUST be stored in the EHC Context. The reserved attribute for that is "tcp_sn"

TCP_ACK designates the EHC Rule compressing / decompressing the TCP Acknowledgment Number and works similarly to TCP SN. Note that "tcp_lsb" is agreed for both TCP SN and TCP Acknowledgment. Similarly the value of the complete TCP Acknowledgment Number MUST be stored in the "tcp_ack" attribute of the EHC Context.

TCP_OPTIONS designates the EHC Rule compressing / decompressing TCP options related fields such as Data Offset and Reserved Bits. TCP_OPTION can only be activated when the TCP Option ("tcp_options") is defined. When "tcp_options" is set to "False" and indicates there are no TCP Options, the Data Offsets and Reserved Bits are compressed / decompressed using the "elided" method with Data Offset and Reserved Bits set to zero.

TCP_CHECK designates the EHC Rule compressing / decompressing the TCP Checksum. TCP_CHECK works similarly as UDP_CHECK.

TCP_URGENT designates the EHC Rule compressing / decompressing the urgent related information. When "tcp_urgent" is set to "False" and indicates there are no TCP Urgent related information, the Urgent Pointer is then "elided" and filled with zeros.

8. Diet-ESP EHC Strategy

From the attributes of the EHC Context, Diet-ESP defined as an EHC Strategy, which EHC Rules to apply. The EHC Strategy is defined for outbound packets which compresses the packet as well as for inbound packet where the decompression occurs.

Diet-ESP results from a compromise between compression efficiency, ease to configure Diet-ESP and the various use cases considered. In order to achieve a great simplicity,

- Diet-ESP favors compression methods that required fewer configuration: For IPv6, ip6_tcfl_comp and ip6_hl_com to "Outer" so that ip6_tc, ip6_fl and ip6_hl can be derived from the packet. Similarly, ip4_ttl_comp has is set to "Outer" so ip4_tll can be derived from the packet.
- Diet-ESP limits compression method to those foreseen as the most commonly used. As such, esp_sn_gen has been set to "Incremental" as this is the most common method used to generate SN. The other method would be "Time".

- Diet-ESP limits compression to the most foreseen scenarios. IPv4 compression has been limited in favor of IPv6 as constraint devices have largely adopted IPv6, and the gain versus the complexity to deploy IPv4 inner IP addresses has not been proved. As a result some compressions for IPv4 are not considered by DIet-ESP. This involved compression of the IPv4 options by setting ip4_options to "No_Options". Similarly IPv4 ID compression has not been enabled by setting ip4_id and ip4_id_lsb to "Unspecified".
- Diet-ESP negotiated values shared by different rules such as tcp_lsb which is shared for TCP ACK as well as for the TCP SN.
- Diet-ESP defines a logic to set the necessary parameters from those agreed by the standard ESP agreement, which limits the setting of parameters.

The following tables shows, which EHC Rules are activated by default for the supported protocols ESP, IPv4, IPv6, UDP, UDP-Lite and TCP when using the Diet-ESP strategy and which ones are activated due to certain circumstances or explicit negotiation

ESP:

EHC Rule	Activated if	Parameter	Value
ESP_SPI	Diet-ESP	esp_spi_lsb	Negotiated
		esp_spi	In SA
ESP_SN	Diet-ESP	esp_sn_lsb	Negotiated
		esp_sn_gen	Negotiated
		esp_sn	In SA
ESP_NH	Diet-ESP	ipsec_mode	In SA
		l4_proto	In SA
ESP_PAD	Diet-ESP	esp_align	Negotiated
		esp_encr	In SA

IPv4:

EHC Rule	Activated if	Parameter	Value
IP4_OPT_DIS	ip_version==4	ip_version	In SA
IP4_LENGTH	ip_version==4	None	
IP4_FRAG_DIS	ip_version==4	None	
IP4_TTL_OUTER	ip_version==4	None	
IP4_TTL_OUTER	ip_version==4	l4_proto	In SA
IP4_CHECK	ip_version==4	None	
IP4_SRC	ip_version==4	ip4_src	In SA
IP4_DST	ip_version==4	ip4_dst	In SA

IPv6:

EHC Rule	Activated if	Parameter	Value
IP6_OUTER	ip_version==6	ip_version	In SA
IP6_LENGTH	ip_version==6	None	
IP6_NH	ip_version==6	l4_proto	In SA
IP6_HL_OUTER	ip_version==6	None	
IP6_SRC	ip_version==6	ip6_src	In SA
IP6_DST	ip_version==6	ip6_dst	In SA

UDP:

EHC Rule	Activated if	Parameter	Value
UDP_SRC	l4_proto==17	l4_source	In SA
UDP_DST	l4_proto==17	l4_dest	In SA
UDP_LENGTH	l4_proto==17	None	
UDP_CHECK	l4_proto==17	None	

UDP-Lite:

EHC Rule	Activated if	Parameter	Value
UDP_LITE_SRC	l4_proto==136	l4_source	In SA
UDP_LITE_DST	l4_proto==136	l4_dest	In SA
UDP_LITE_COVERAGE	l4_proto==136	udplite_coverage	Negotiated
UDP_LITE_CHECK	l4_proto==136	None	

TCP:

EHC Rule	Activated if	Parameter	Value
TCP_SRC	l4_proto==6	l4_source	In SA
TCP_DST	l4_proto==6	l4_dest	In SA
TCP_SN	l4_proto==6	tcp_sn	In SA
		tcp_lsb	Negotiated
TCP_ACK	l4_proto==6	tcp_ack	In SA
		tcp_lsb	Negotiated
TCP_OPTIONS	l4_proto==6	tcp_options	Negotiated
TCP_CHECK	l4_proto==6	None	
TCP_URGENT	l4_proto==6	tcp_urgent	Negotiated

Thus, the parameters that the two peers needs to agree on are:

- esp_sn_lsb
- esp_spi_lsb
- esp_align
- udplite_coverage
- tcp_lsb
- tcp_options
- tcp_urgent

Implementation may differ from the description below. However, the outcome MUST remain the same.

8.1. Outbound Packet Processing

Diet-ESP compression is defined as follows:

1. In phase "pre-esp": Match the inbound packet with the SA and determine if the Diet-ESP EHC Strategy has been activated. If the Diet-ESP EHC Strategy has been activated proceed to next

step, otherwise skip all steps associated to Diet-ESP and proceed to the standard ESP as defined in [RFC4303]
2. In phase "pre-esp": If "l4_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), proceed to the compression of the specific layer. Otherwise, the transport layer is not compressed.
3. In phase "clear text esp": If "ipsec_mode" is set to "Tunnel" mode, determine "ip_version" the IP version of the inner IP addresses and proceed to the appropriated inner IP address compression.
4. In phase "clear text esp" and "post-esp": Proceed to the ESP compression.

UDP compression is defined as below:

1. If "l4_src" designates a "Single" Source Port, apply UDP_SRC to compress the Source Port.
2. If "l4_dst" designates a "Single" Destination Port, apply UDP_DST to compress the Destination Port.
3. Apply UDP_CHECK to compress the Checksum.
4. Apply UDP_LENGTH to compress the Length.

UDP-lite compression is defined as below:

1. If "l4_src" designates a "Single" Source Port, apply the UDP-LITE_SRC to compress the Source Port.
2. If "l4_dst" designates a "Single" Destination Port, apply the UDP-LITE_DST, to compress the Destination Port.
3. If "udplite_coverage" is specified, apply the UDP-LITE_COVERAGE, to compress the Coverage.
4. Apply UDP-LITE_CHECK to compress the Checksum.

TCP compression is defined as below:

1. If "l4_src" designates a "Single" Source Port than apply the TCP_SRC to compress the Source Port.
2. If "l4_dst" designates a "Single" Destination Port than apply the TCP_DST to compress the Destination Port.
3. If "tcp_lsb" is lower than 4, then "tcp_sn" "tcp_ack" attributes of the Diet-ESP Context are updated with the value provided from the packet before applying the TCP_SN and the TCP_ACK EHC Rules.
4. If "tcp_options" is set to "False" apply the TCP_OPTIONS EHC Rule.
5. If "tcp_urgent" is set to "False" apply the TCP_URGENT EHC Rule.
6. Apply TCP_CHECK to compress the Checksum.

Inner IPv6 Header compression is defined as below:

1. If "ip6_src" designates a "Single" Source IP address, apply the IP6_SRC to compress the IPv6 Source Address.
2. If "ip6_dst" designates a "Single" Destination IP address, apply the IP6_DST to decompress the IPv6 Destination Address.
3. Apply IPv6_HL_OUTER to compress the Hop Limit.
4. If "l4_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP6_NH to compress the Next Header.
5. Apply, IP6_LENGTH to compress the Length.
6. Apply IP6_OUTER to compress Version, Traffic Class and Flow Label.

Inner IPv4 Header compression is defined as below:

1. Apply, IP4_LENGTH to compress the Length.
2. Apply IP4__TTL_OUTER to compress Time To Live.
3. Apply, IP4_CHECK to compress the IPv4 header checksum.
4. If "ip4_src" designates a "Single" Source IP address, apply the IP4_SRC to compress the IPv4 Source Address.
5. If "ip4_dst" designates a "Single" Destination IP address, apply the IP4_DST to decompress the IPv4 Destination Address.

ESP compression is defined as below:

1. In phase "clear text esp": If "ipsec_mode" is set to "Tunnel" or "l4_proto" is set to a "Single value - eventually different from TCP, UDP or UDP-Lite, apply ESP_NH, to compress the Next Header.
2. In phase "clear text esp": If "esp_encr" specify an encryption algorithm that does not provide padding, then apply ESP_PAD to compress the Pad Length and Padding.
3. Proceed to the ESP encryption as defined in [RFC4303].
4. In phase "post-esp: If "esp_sn_lsb" is different from 4, then apply ESP_SN. To compress the ESP SN.
5. In phase "post-esp": If "esp_spi_lsb" is different from 4, then apply ESP_SPI to compress the SPI.

8.2. Inbound Packet Processing

Diet-ESP decompression is defined as follows:

1. Match the inbound packet with the SA and determine if the Diet-ESP EHC Strategy has been activated. When Diet-ESP is activated this means that the "esp_spi_lsb" are sufficient to index the SA and proceed to next step, otherwise skip all steps associated to Diet-ESP and proceed to the standard ESP as defined in [RFC4303]
2. In phase "clear text esp" and "post-esp": Proceed to the ESP decompression.
3. In phase "clear text esp": If "ipsec_mode" is set to "Tunnel" mode, determine "ip_version" the IP version of the inner IP

addresses and proceed to the appropriated inner IP address decompression, except for the computation of the checksums and length.
4. In phase "pre-esp": If "l4_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), proceed to the decompression of the specific layer, except for the computation of the checksums and length replaced by zero fields.
5. In phase "pre-esp": Proceed to the decompression of the checksums and length.

ESP decompression is defined as follows:

1. In phase "post-esp": If "esp_spi_lsb" is different from 4, then apply ESP_SPI to decompress the SPI.
2. In phase "post-esp: If "esp_sn_lsb" is different from 4, then apply ESP_SN. To decompress the ESP SN.
3. Proceed to the ESP signature validation and decryption as defined in [RFC4303].
4. In phase "clear text esp": If "ipsec_mode" is set to "Tunnel" or "l4_proto" is set to a "Single value - eventually different from TCP, UDP or UDP-Lite, apply ESP_NH, to decompress the Next Header.
5. In phase "clear text esp": If "esp_encr" specify an encryption algorithm that does not provide padding, then apply ESP_PAD to compress the Pad Length and Padding.
6. Extract the ESP Data Payload and apply decompression EHC Rule to the ESP Data Payload.

UDP decompression is defined as follows:

1. If "l4_src" designates a "Single" Source Port, apply UDP_SRC to decompress the Source Port.
2. If "l4_dst" designates a "Single" Destination Port, apply UDP_DST to decompress the Destination Port.
3. Apply UDP_LENGTH to compress the Length. The length value is computed from the length provided by the lower layer, with the additional added bytes during the UDP decompression including the length size.
4. Apply UDP_CHECK to decompress the Checksum.
5. Update the Length of the lower layers:

    1. If "ipsec_mode" is set to "Transport" mode, update the Length of the outer IP header (IPv4 or IPv6). The Length is incremented by the number of bytes generated by the decompression of the transport layer.
    2. If "ipsec_mode" is set to "Tunnel" mode, update the Length of the inner IP address (IPv4 or IPv6) as well as the outer IP header (IPv4 or IPv6). The Length is incremented by the

number of bytes generated by the decompression of the transport layer.

UDP-Lite decompression is defined as follows:

1. If "l4_src" designates a "Single" Source Port, apply the UDP-LITE_SRC to decompress the Source Port.
2. If "l4_dst" designates a "Single" Destination Port, apply the UDP-LITE_DST, to decompress the Destination Port.
3. If "udplite_coverage" is specified, apply the UDP-LITE_COVERAGE, to decompress the Coverage.
4. Apply UDP-LITE_CHECK to compress the Checksum.
5. Update the Length of the lower layers as defined in UDP.

TCP decompression is defined as follows:

1. If "l4_src" designates a "Single" Source Port than apply the TCP_SRC to decompress the Source Port.
2. If "l4_dst" designates a "Single" Destination Port than apply the TCP_DST to decompress the Destination Port.
3. If "tcp_lsb" is lower than 4, apply TCP_SN and the TCP_ACK to decompress the TCP Sequence Number and the TCP Acknowledgment Number.
4. If "tcp_options" is set to "False" apply TCP_OPTIONS to decompress Data Offset and Reserved Bits.
5. If "tcp_urgent" is set to "False" apply the TCP_URGENT to decompress the Urgent Pointer.
6. Apply TCP_CHECK to decompress the Checksum.

Inner IPv6 decompression is defined as follows:

1. Apply IP6_OUTER to decompress Version, Traffic Class and Flow Label.
2. Set the Length to zero.
3. If "l4_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP6_NH to decompress the Next Header.
4. Hop Limit is decompressed with IP6_HL_OUTER (with "ip6_hl_comp" set to "Outer").
5. If the "ip6_src" designates a "Single" Source IP address, apply the IP6_SRC to de compress the IPv6 Source Address.
6. If the "ip6_dst" designates a "Single" Destination IP address than apply the IP6_DST to decompress the IPv6 Destination Address.
7. Apply, IP6_LENGTH to provide the replace the zero length value by its appropriated appropriated value. The Length value considers the length provided by the lower layers to which are added the additional bytes due to the decompression, minus the length of the inner IP6 Header.

Inner IPv4 decompression is defined as follows:

1. Apply, IP4_LENGTH to provide the replace the zero length value by its appropriated appropriated value. The Length value considers the length provided by the lower layers to which are added the additional bytes due to the decompression, minus the length of the inner IPv4 Header. The value computed from the lower layer will have to be overwritten in case further decompression occurs.
2. Apply IP4_TTL_OUTER to decompress Time To Live.
3. If "l4_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP4_PROT to decompress the Protocol Field.
4. If "ip4_src" designates a "Single" Source IP address, apply the IP4_SRC to de compress the IPv4 Source Address.
5. If "ip4_dst" designates a "Single" Destination IP address than apply the IP4_DST to decompress the IPv4 Destination Address.
6. Apply IP4_CHECK to decompress the checksum of the IPv4 header

9. IANA Considerations

There are no IANA consideration for this document.

10. Security Considerations

This section lists security considerations related to the Diet-ESP protocol.

Security Parameter Index (SPI):

The Security Parameter Index (SPI) is used by the receiver to index the Security Association that contains appropriated cryptographic material. If the SPI is not found, the packet is rejected as no further checks can be performed. In EHC, the value of the SPI is not reduced, but compressed why the SPI value may not be fully provided between the compressor and the de-compressor. On the other hand, its uncompressed value is provided to the ESP-procession and no weakness is introduced to ESP itself. On an implementation perspective, it is strongly recommended that decompression is deterministic. Compression and decompression adds some additional treatment to the ESP packet, which might be used by an attacker. In order to minimize the load associated to decompression, decompression is expected to be deterministic. The incoming compressed SPI with the associated IP addresses should output a single and unique uncompressed SPI value. If an uncompressed SPI values have to be considered, then the receiver could end in n signature checks which may be used by an attacker for a DoS attack.

Sequence Numer (SN):

The Sequence Number (SN) is used as an anti-replay attack mechanism. Compression and decompression of the SN is already

part of the standard ESP namely the Extended Sequence Number (ESN). The SN in a standard ESP packet is 32 bit long, whether EHC enables to reduce it to 0 bytes and the main limitation to the compression a deterministic decompression. SN compression consists in indicating the least significant bits of the uncompressed SN on the wire. The size of the compressed SN must consider the maximum reordering index such that the probability that a later sent packet arrives before an earlier one. In addition the size of SN should also consider maximum consecutive packets lost during transmission. In the case of ESP, this number is set to 2^32 which is, in most real world case, largely over-provisioned. When the compression of the SN is not appropriately provisioned, the most significant bit value may be de-synchronized between the sending and receiving parties. Although IKEv2 provides some re-synchronization mechanisms, in case of IoT the de-synchronization will most likely result in a renegotiation and thus DoS possibilities. Note that IoT communication may also use some external parameters, i.e. other than the compressed SN, to define whether a packet be considered or not and eventually derive the SN. One such scenario may be the use of time windows. Suppose a device is expected to send some information every hour or every week. In this case, for example, the SN may be compressed to zero bytes. Instead the SN may be derived by incrementing the SN every hour after the last received valid packet. Considering the time the packet is received make it possible to consider the time derivation of the sensor clock. If TIME is used as the method to generate the SN, the receiver MUST ensure that the esp_sn_lsb is big enough to resist time differences between the nodes. Note also that the anti-replay mechanism needs to define the size of the anti-replay window.[RFC4303] provides guidance to set the window size and are similar to those used to define the size of the compressed SN.

11. Privacy Considerations

Security Parameter Index (SPI):
Until Diet-ESP is not deployed outside the scope of IoT and small devices, the use of a compressed SPI may provide an indication that one of the endpoint is a sensor. Such information may be used, for example, to evaluate the number of appliances deployed, or - in addition with other information, such as the time interval, the geographic location - be used to derive the type of data transmitted.

Sequence Number (SN): If incremented for each ESP packet, the SN may leak some information like the amount of transmitted data or the age of the sensor. The age of the sensor may be correlated with the software used and the potential bugs. On the other hand, re-keying will re-initialize the SN, but the cost of a re-keying may

not be negligible and thus, frequent re-keying can be considered. In addition to the re-key operation, the SN may be generated in order to reduce the accuracy of the information leaked. In fact, the SN does not have to be incremented by one for each packet it just has to be an increasing function. Using a function such as a TIME may prevent characterizing the age or the use of the sensor. Note that the use of such function may also impact the compression efficiency and result in larger compressed SN.

12. Acknowledgment

We would like to thank Orange and Universitee Pierre et Marie Curie for initiating the work on Diet-ESP. We Would like to thank Sylvain Killian for implementing an open source Diet-ESP on Contiki and testing it on the FIT IoT-LAB [fit-iot-lab] funded by the French Ministry of Higher Education and Research. We thank the IoT-Lab Team and the INRIA for maintaining the FIT IoT-LAB platform and for providing feed backs in an efficient way.

We would like to thank Bob Moskowitz for not copyrighting Diet HIP. The "Diet" terminology is from him.

We would like to thank those we received many useful feed backs among others: Dominique Bartel, Anna Minaburo, Suresh Krishnan, Samita Chakrabarti, Michael Richarson, Tero Kivinen.

13. References

13.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[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>.

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

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <https://www.rfc-editor.org/info/rfc4309>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <https://www.rfc-editor.org/info/rfc4944>.

[RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, DOI 10.17487/RFC5795, March 2010, <https://www.rfc-editor.org/info/rfc5795>.

[RFC5858] Ertekin, E., Christou, C., and C. Bormann, "IPsec Extensions to Support Robust Header Compression over IPsec", RFC 5858, DOI 10.17487/RFC5858, May 2010, <https://www.rfc-editor.org/info/rfc5858>.

[RFC7400] Bormann, C., "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 7400, DOI 10.17487/RFC7400, November 2014, <https://www.rfc-editor.org/info/rfc7400>.

[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>.

13.2. Informational References

[I-D.toutain-6lpwa-ipv6-static-context-hc]
Minaburo, A. and L. Toutain, "6LPWA Static Context Header Compression (SCHC) for IPV6 and UDP", draft-toutain-6lpwa-ipv6-static-context-hc-01 (work in progress), June 2016.

[I-D.mglt-ipsecme-implicit-iv]
Migault, D., Guggemos, T., and Y. Nir, "Implicit IV for Counter-based Ciphers in IPsec", draft-mglt-ipsecme-implicit-iv-04 (work in progress), June 2017.

[I-D.ietf-tsvwg-udp-options]
Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-07 (work in progress), March 2019.

[fit-iot-lab]
"Future Internet of Things (FIT) IoT-LAB", <https://www.iot-lab.info>.

Appendix A. Illustrative Examples

A.1. Single UDP Session IoT VPN

This section considers a IoT IPv6 probe hosting a UDP application. The probe is dedicated to a single application and establishes a single UDP session. As a result, inner IP addresses and UDP Ports have a "Single" value and can be easily compressed. The probes sets an IPsec VPN using IPv6 addresses in order to connect its secure domain - typically a Home Gateway. The use of IPv6 for inner and outer IP addresses, enables to infer inner IP fields from the outer IP address. The probes encrypts with AES-CCM_8 [RFC4309]. AES-CCM does not have padding, so the padding is performed by ESP. The probes uses an 8 bit alignment which enables to fully compress the ESP Trailer. In addition, as the probe SA is indexed using the outer IP addresses (or eventually the radio identifiers) which enables to fully compress the SPI. As the probe provides information every hour, the Sequence Number using time can be derived from the received time, which enables to fully compress the SN.

Figure 3 represents the original UDP packet and Figure 4 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 61 bytes overhead. With IPv4 inner IP addressed Diet-ESP results in an 45 byte overhead reduction.

Further compression may be done for example by using an implicit IV [I-D.mglt-ipsecme-implicit-iv] and by compressing the outer IP addresses (not represented) on the figure. In addition, application data may also be compressed with mechanisms outside of the scope of Diet-ESP.

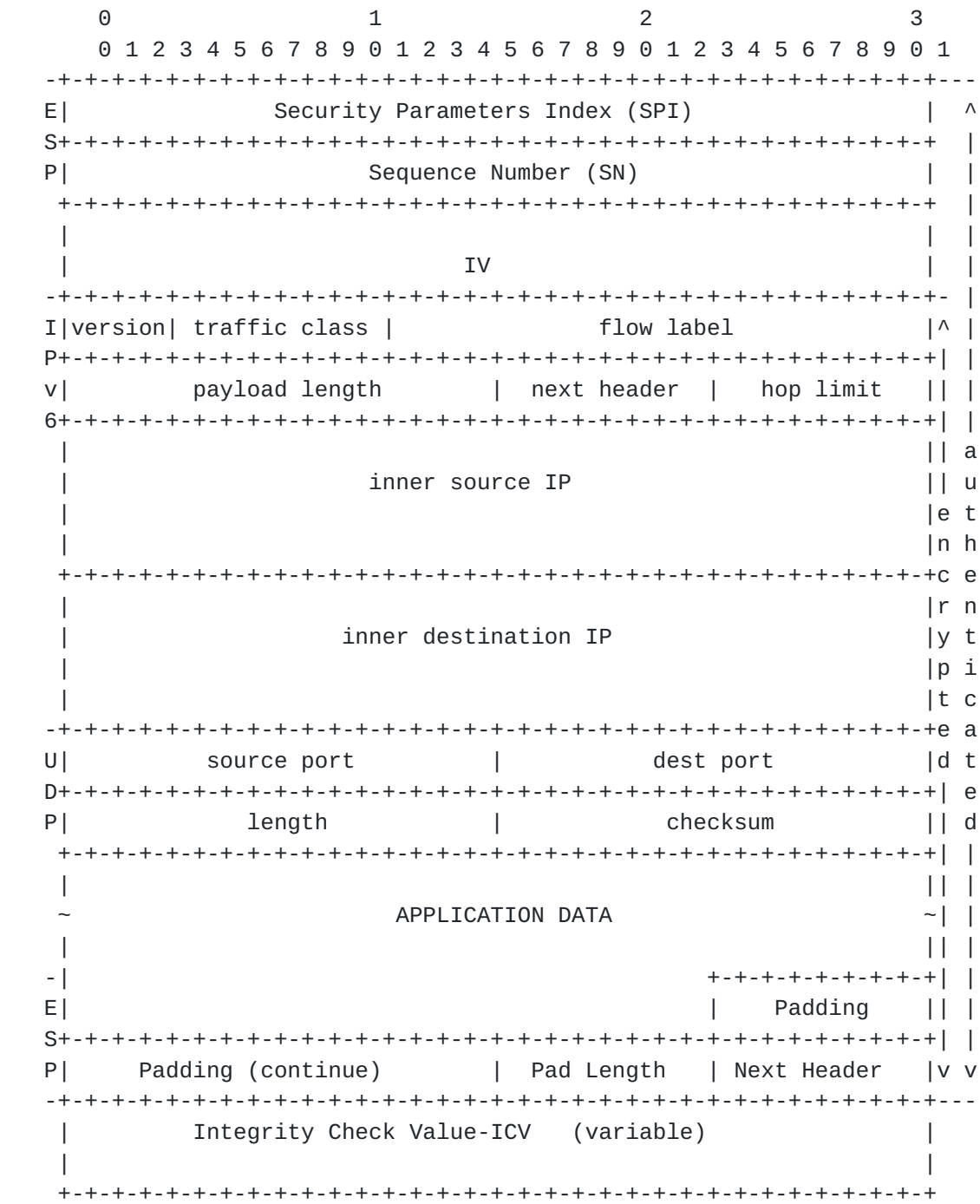

Figure 3: Standard ESP VPN Packet Description

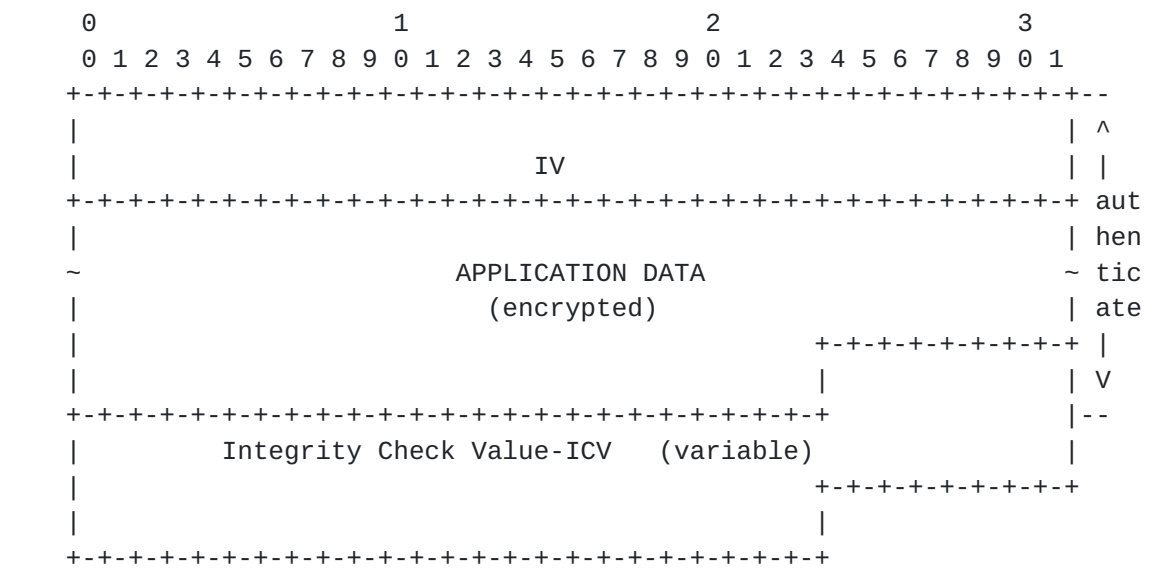

Figure 4: Diet-ESP Single UDP Session IoT VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified".

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	0
ESP_SN	esp_sn_lsb	0
	esp_sn_gen	
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	
	ip6_hl_comp	
IP6_LENGTH		
IP6_NH		
IP6_HL_OUTER		
IP6_SRC		
IP6_DST		
UDP_SRC		
UDP_DST		
UDP_LENGTH		
UDP_CHECK		

A.2. Single TCP session IoT VPN

This section considers the same probe as described in Appendix A.1 but instead of using UDP as a transport layer, the probe uses TCP. In this case TCP is used with no options, no urgent pointers and the SN and ACK Number are compressed to 2 bytes as the throughput is expected to be low.

Figure 5 represents the original TCP packet and Figure 6 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 66 bytes overhead. With IPv4 inner address Diet-ESP results in a 50 byte overhead reduction.

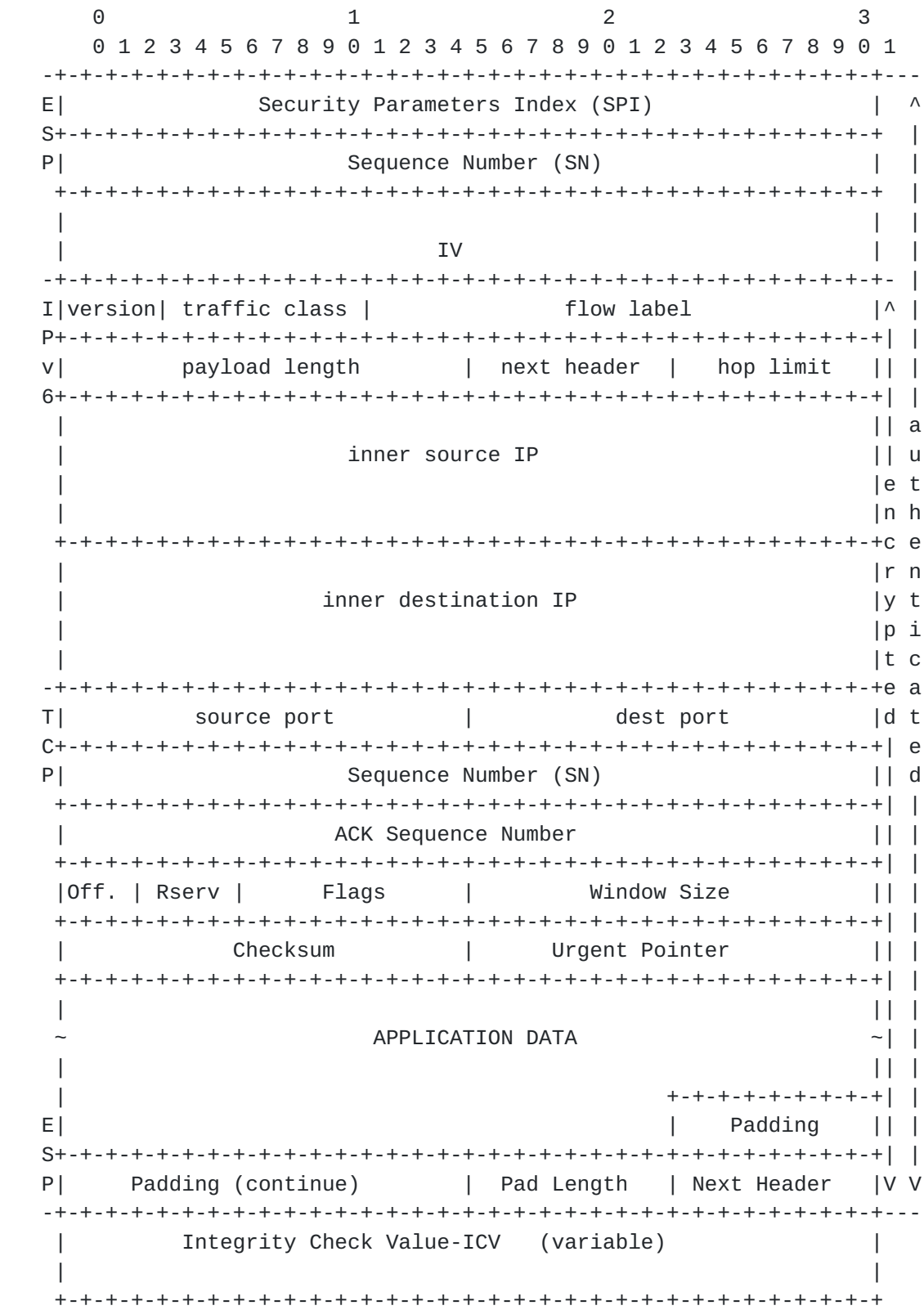

Figure 5: Standard IoT Single TCP Session VPN Packet Description

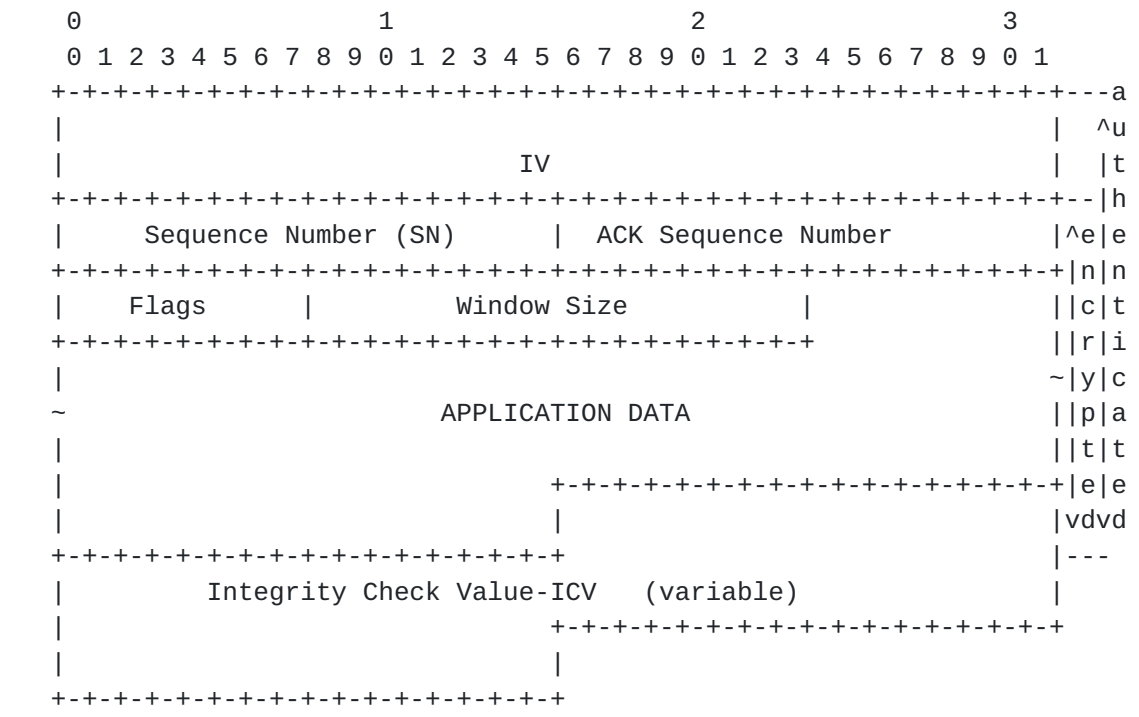

Figure 6: Diet-ESP Single TCP Session IoT VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified". Note for simplicity, tcp_sn and tcp_ack are negotiated to start with 0, but it could be any other value as well.

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	0
ESP_SN	esp_sn_lsb	0
	esp_sn_gen	
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	
	ip6_hl_comp	
IP6_LENGTH		
IP6_NH		
IP6_HL_OUTER		
IP6_SRC		
IP6_DST		
TCP_SRC		
TCP_DST		
TCP_SN	tcp_lsb	2
	tcp_sn	0
TCP_ACK	tcp_lsb	2
	tcp_ack	0
TCP_OPTIONS	tcp_options	"False"
TCP_CHECK		
TCP_URGENT	tcp_urgent	"False"

A.3. Traditional VPN

This section illustrates the case of an company VPN. The VPN is typically set by a remote host that forwards all its traffic to the security gateway. As transport protocols are "Unspecified", compression is limited to ESP and the inner IP header. For the inner IP header, the Destination IP address is "Unspecified" so the compression of the inner IP address excludes the Destination IP address. Similarly, the inner IP Next Header cannot be compressed as the transport layer is not specified. For ESP, the security gateway may only have a sufficiently low number of remote users with relatively low throughput in which case SPI and SN can be compressed to 2 bytes. As throughput remains relatively low, the alignment may also set to 8 bits.

A.3.1. IPv6 in IPv6

Figure 7 represents the original TCP packet with IPv6 inner IP addresses and Figure 8 represents the corresponding packet compressed

with Diet-ESP. The compression with Diet-ESP results in a reduction of 32 bytes.

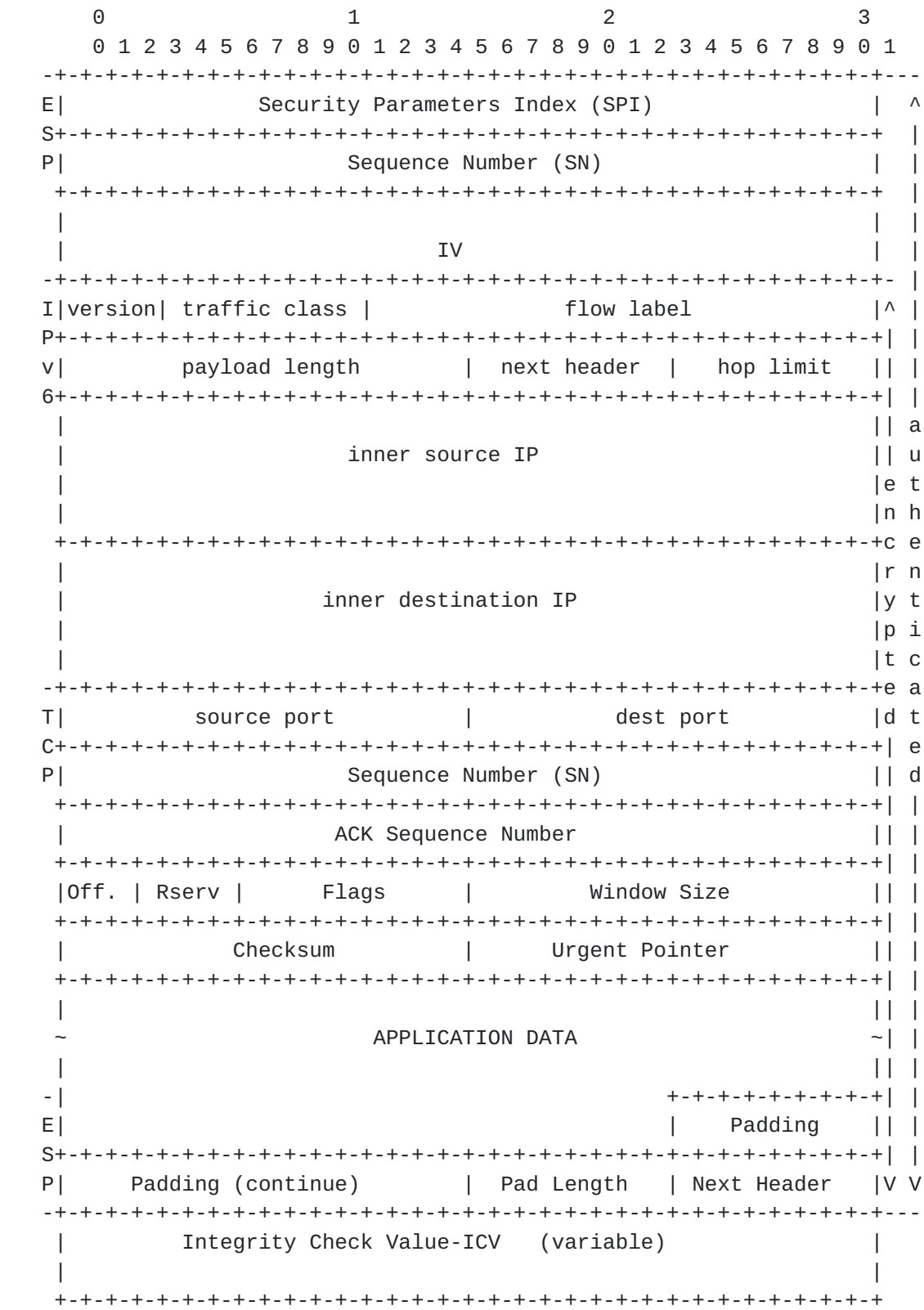

Figure 7: Standard ESP VPN Packet Description

Figure 8: Diet-ESP VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified".

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	
IP6_LENGTH		
IP6_HL_OUTER	ip6_hl_comp	
IP6_SRC		

A.3.2. IPv6 in IPv4

If the compressed inner IP header is an IPv6, but the outer IP header is an IPv4 header, the activated rules differ, as IP6_OUTER cannot be used. Instead, ip6_tcfl_comp and ip6_hl_comp are set to "Value". The resulting ESP packet is the same as in Figure 8.

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	
IP6_LENGTH		
IP6_HL_OUTER	ip6_hl_comp	
IP6_SRC		

A.3.3. IPv4 in IPv4

Figure 9 represents the original TCP packet with IPv4 inner IP addresses and Figure 10 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 24 bytes.

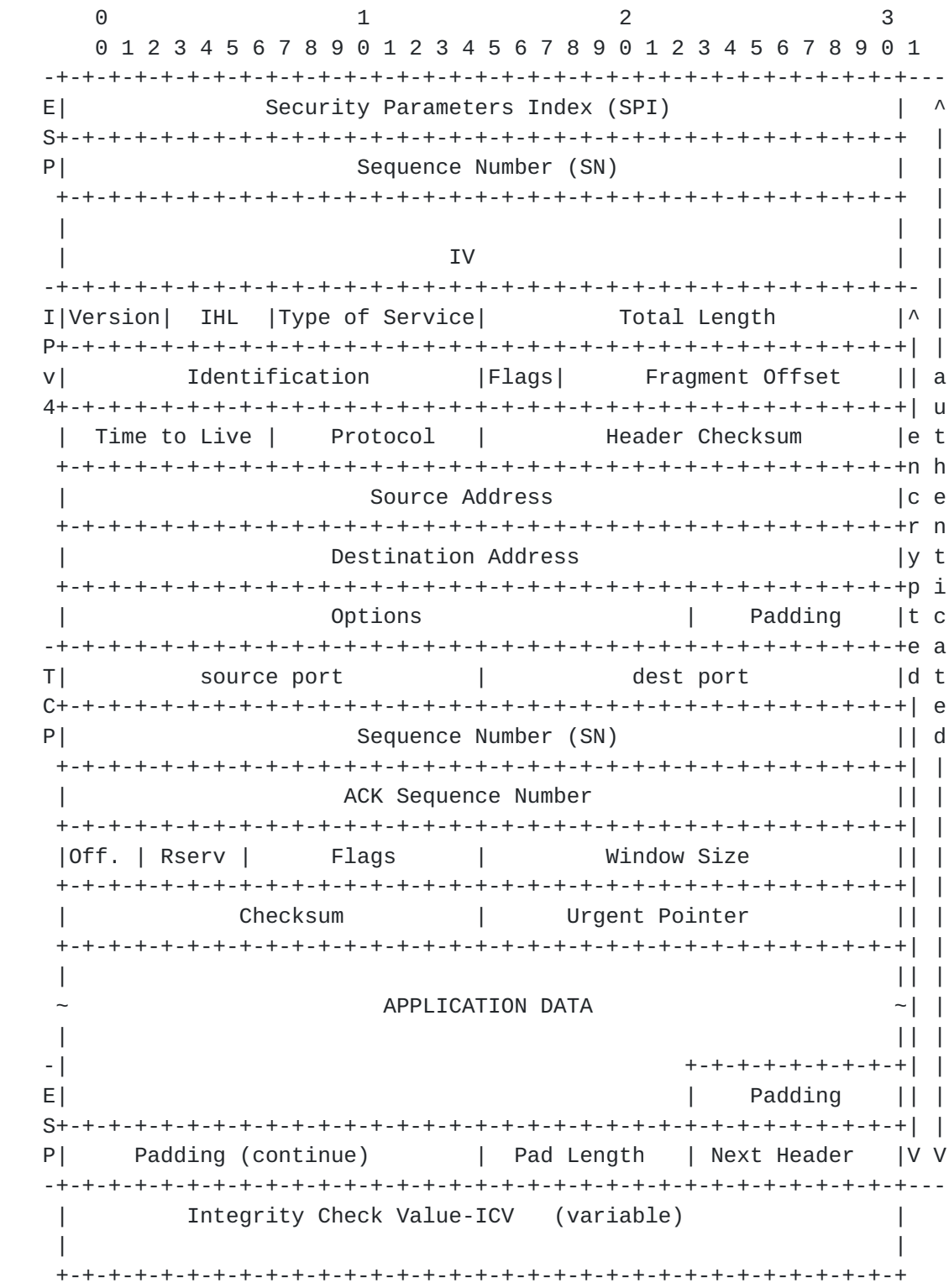

Figure 9: Standard ESP VPN Packet Description

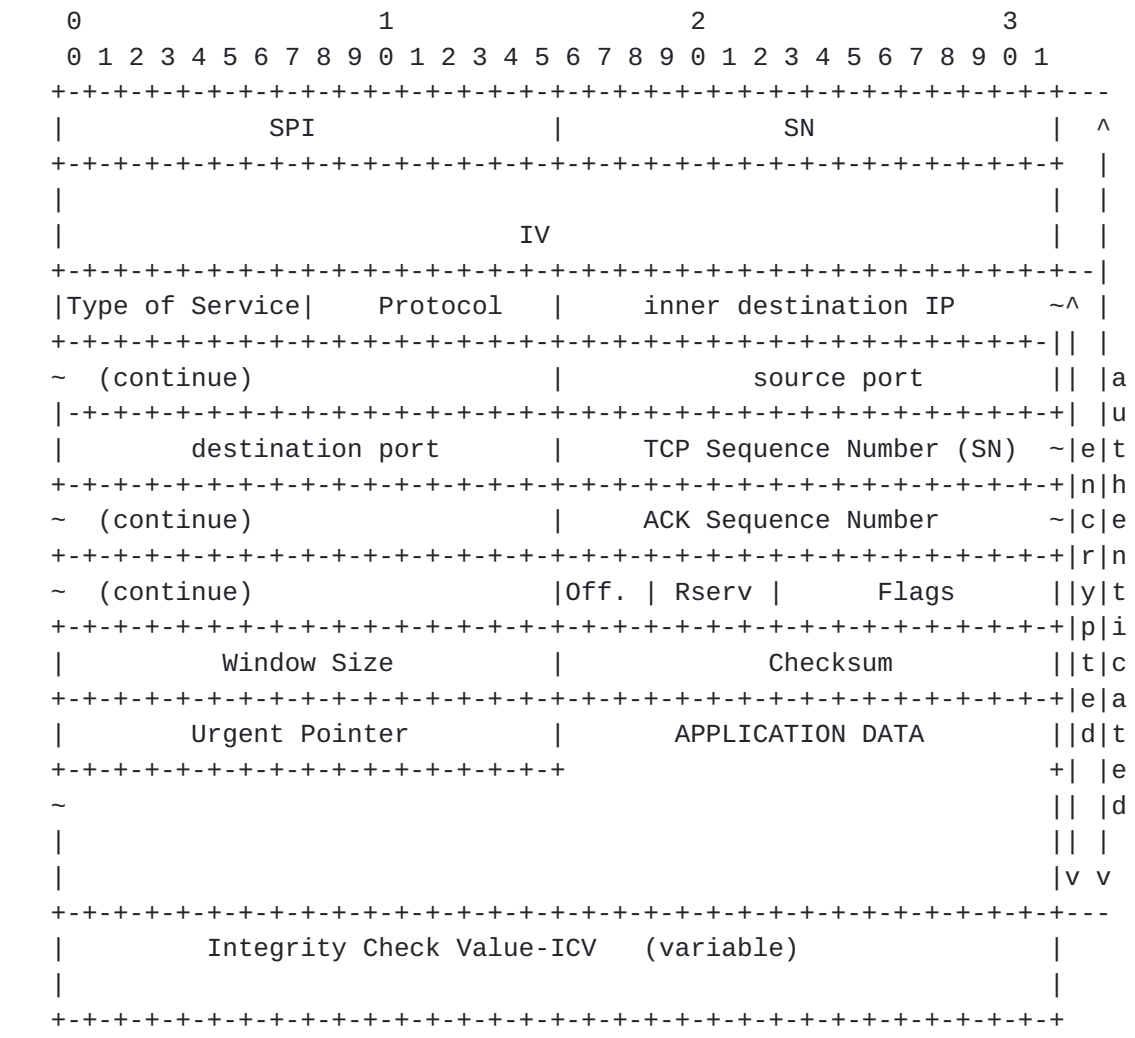

Figure 10: Diet-ESP VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified".

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP4_OPT_DIS		
IP4_LENGTH		
IP4_FRAG_DIS		
IP4_TTL_OUTER		
IP4_CHECK		
IP4_SRC		

A.3.4. IPv4 in IPv6

If the compressed inner IP header is an IPv4, but the outer IP header is an IPv6 header, the activated rules differ, as IP4_TTL_OUTER cannot be used. Instead, IP4_TTL_VALUE is used. The resulting ESP packet is the same as in Figure 10.

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP4_OPT_DIS		
IP4_LENGTH		
IP4_FRAG_DIS		
IP4_CHECK		
IP4_SRC		

Authors' Addresses

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Email: daniel.migault@ericsson.com

Tobias Guggemos
LMU Munich
Oettingenstr. 67
80538 Munchen, Bavaria
Germany

Email: guggemos@nm.ifi.lmu.de
URI: http://www.nm.ifi.lmu.de/~guggemos

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043
USA

Email: dschinazi.ietf@gmail.com