Workgroup: IPsecme
Internet-Draft: draft-mglt-ipsecme-diet-esp-12
Published: 18 March 2024
Intended Status: Standards Track
Expires: 19 September 2024
Authors: D. Migault
Ericsson
T. Guggemos
LMU
C. Bormann
Universitaet Bremen TZI
D. Schinazi
Google LLC

ESP Header Compression Profile

Abstract

ESP Header Compression Profile (EHCP) defines a profile to compress communications protected with IPsec/ESP.

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 September 2024.

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

This document defines a profile to compress IPsec/ESP [RFC4301] / [RFC4303] traffic represented by Figure 1.

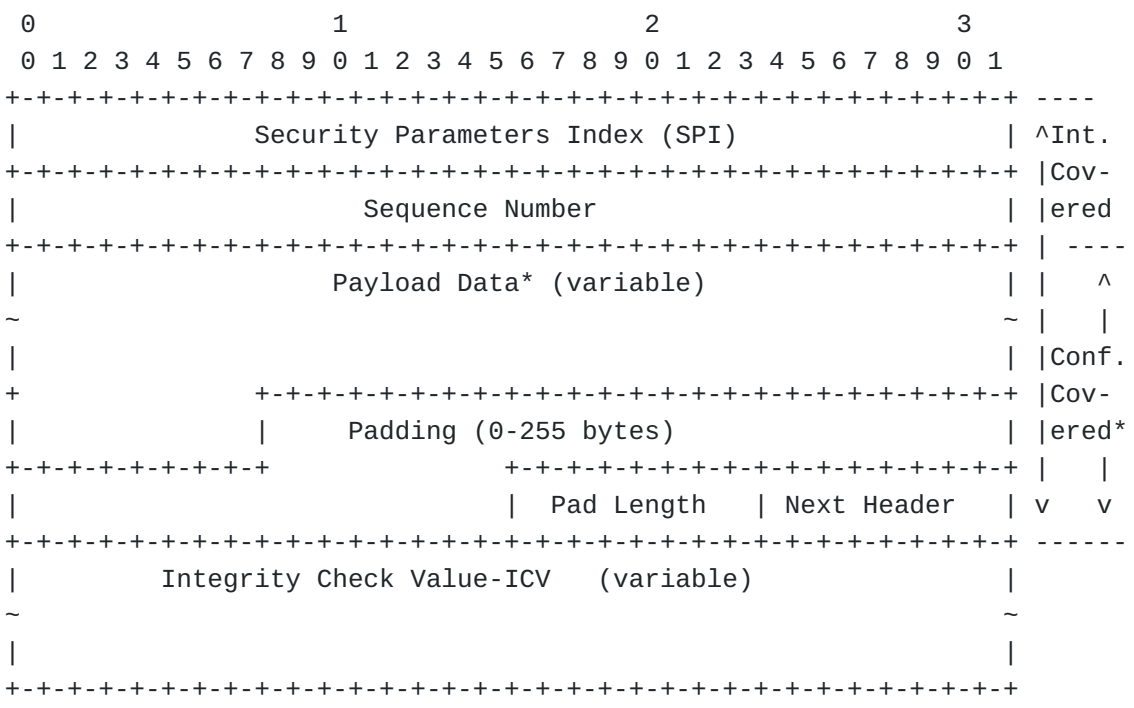

Figure 1: Top-Level Format of an ESP Packet

The main principle is to avoid sending information that has already been shared by the peers. As depicted in Figure 2, this profiles defines two levels of compression. The first level is the Clear Text ESP Compression (CTE C) compresses all fields that will later be encrypted by ESP - that is the Payload Data, the Padding, the Pad Length and the Next Header. The second is the Encrypted ESP Compression (EE C) compresses ESP remaining fields that is the Security Parameters Index (SPI) and Sequence Number (SN).

The decompression of the inbound packet follows the reverse path the Encrypted ESP Decompression (EE D) decompressed the unencrypted ESP header fields while the Clear Text ESP Decompression (CT D) is performed once the ESP packet is decrypted.

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

The CTE C/D and EE C/D are expressed via the Generic Framework for Static Context Header (SCHC) [RFC8724]. The SCHC rules are derived from the ESP Compression Header Context which includes the Security Association (SA) as well as an additional parameters. This is the main content of this document.

It is expected that all necessary arguments are agreed via IKEv2 [I-D.mglt-ipsecme-ikev2-diet-esp-extension].

In some case, additional compression may occur on the inner IP packet before being processed by IPsec/ESP as well as over the Outer IP packet. Such compression, decompression are outside the scope of this document.

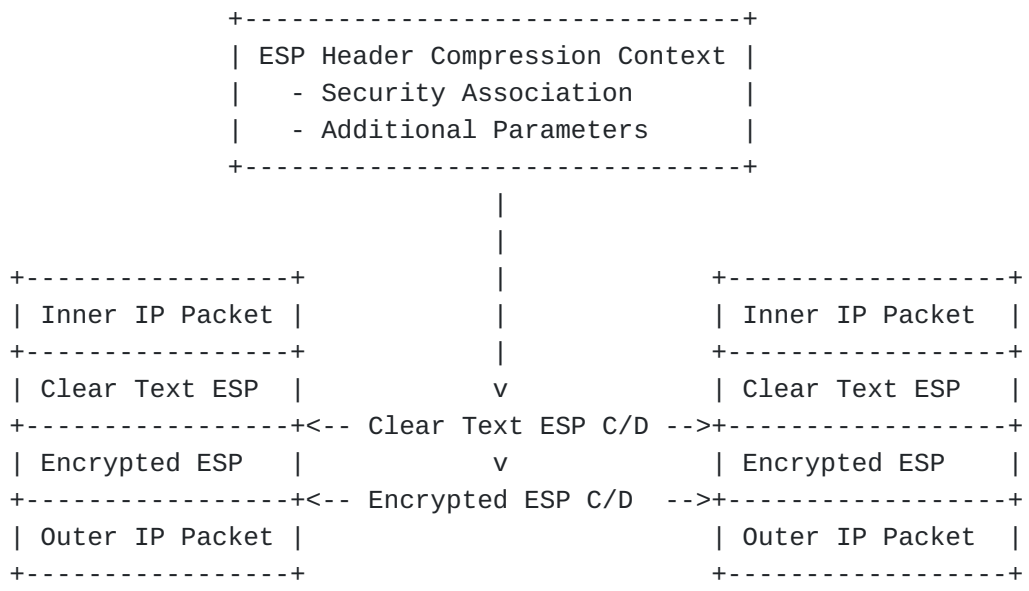

Figure 2: ESP Compression Architecture

3. ESP Header Compression Context

The EHC Context provides the necessary information to generate the SCHC Rules. Most pieces of information are already available from the negotiated SA [RFC4301]. Other pieces of information needs to be specifically configured or agreed via other mechanisms like for example [I-D.mglt-ipsecme-ikev2-diet-esp-extension].
The reference column of Figure 3 indicates how the information is defined.

The Compression / Decompression (C / D) column specifies in which of the compression the parameter is being used.

Note that additional Compression might be performed especially on the inner IP packet - for example, including the TCP layer. However, this profiles limits the scope of the compression to UDP packets as well as the inner IP header. We believe that is a reasonable scope for ESP to address both IoT UDP packets as well as large VPN traffic. If further compression are needed, this should be achieved by sending an IP packet with an SCHC payload where the expected compression is achieved outside ESP.

The following attributes are considered by this EHC Context. Implementations may consider different expression of the parameters

but their behavior is expected to remain compatible with this specification.

EHC Context	Possible Values	Reference	C / D
alignment	"8 bit", "32 bit"	ThisRFC	CT E
ipsec_mode	"Tunnel", "Transport"	RFC4301	CT E
tunnel_ip	IPv4, IPv6 address	RFC4301	CT E
esp_spi	ESP SPI	RFC4301	EE
esp_spi_lsb	0, 1, 2, 3, 4*	ThisRFC	EE
esp_sn	ESP Sequence Number	RFC4301	EE
esp_sn_lsb	0, 1, 2, 3, 4*	ThisRFC	EE
esp_encr	ESP Encryption Algorithm	RFC4301	CT E
ts_flow_label	True, False	ThisRFC	CT E
ts_ip_version	4, 6	ThisRFC	CT E
ts_ip_src_start	IP4 or IPv6 address	ThisRFC	CT E
ts_ip_src_end	IP4 or IPv6 address	ThisRFC	CT E
ts_ip_dst_start	IPv4 or IPv6 address	ThisRFC	CT E
ts_ip_dst_end	IPv4 or IPv6 address	ThisRFC	CT E
ts_proto_list	TCP, UDP, ..., 0	ThisRFC	CT E
ts_port_src_start	Port number	ThisRFC	CT E
ts_port_src_end	Port number	ThisRFC	CT E
ts_port_dst_start	Port number	ThisRFC	CT E
ts_port_dst_end	Port number	ThisRFC	CT E
ts_dsp_list	DSCP number	RFCYYYY	CT E

Figure 3: EHC ESP related parameter

alignment: indicates the byte alignement supported by the OS for the ESP extension. By default, the alignement is 32 bit for IPv6, but some systems may also support a 8 bit alignement. Note that when a block cipher such as AES-CCM is used, an 8 bit alignment is overwritten by the block size.

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

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

esp_spi_lsb: designates the LSB to be considered for the compressed SPI. This parameter is defined by this specification and can take the following values 0, 1, 2, 4 respectively meaning that the

compressed SPI will consist of the esp_spi_lsb LSB bytes of the original SPI. A value esp_spi_lsb will let the SPI unchanged.

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

esp_sn_lsb: designates the LSB to be considered for the compressed SN and is defined by this specification. It works similarly to esp_spi_lsb.

esp_encr: designates the encryption algorithm used. For the purpose of compression is is RECOMMENDED to use [RFC8750].

ts_ * parameters are associated to the Traffic Selectors of the SA and introduces by this specification. This specification limits the expression of the Traffic Selector to be of the form (IP source range, IP destination range, Port source range, Port destination range, Protocol ID list, DSCP list). This limits the original flexibility of the expression of TS, but we believe that provides sufficient flexibility.

ts_flow_label: indicates the Flow Label field of the inner IPv6 or the Identification field of the IPv4 is copied from the outer IP address.

ts_ip_version: designates the IP version of the Traffic Selectors and its values is set to 4 when only IPv4 IP addresses are considered and to 6 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. When the traffic selectors result in a combination of IPv4 and IPv6 addresses, ts_ip_version is undefined.

ts_ip_src_start: designates the starting value range of source IP addresses of the inner packet and has the same meaning as the Starting Address field of the Traffic Selector payload defined in [RFC7296], Section 3.13. Note however that in this specification, ts_ip_src_start applies for all agreed Traffic Selector payloads. When the IP addresses cannot be expressed as a range, that exactly expressed as [ts_ip_src_start, ts_ip_src_end], ts_ip_src_start is undefined.

ts_ip_src_end: designates 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 Traffic Selector payload defined in [RFC7296], Section 3.13. Similarly to ts_ip_src_end, when the IP addresses cannot be expressed as a range, ts_ip_src_end is undefined.

ts_port_src_start:
designates the starting value of the port range of the inner packet and has the same meaning as the Start Port field of the Traffic Selector payload defined in [RFC7296], Section 3.13.

ts_port_src_end: designates the starting value of the port range of the inner packet and has the same meaning as the End Port field of the Traffic Selector payload defined in [RFC7296], Section 3.13.

ts_proto_list: designates the list of Protocol ID field whose meaning is defined in [RFC7296], Section 3.13.

ts_dscp_list: designates the list of DSCP values used by the Traffic Selector and have the same meaning as the List of DSCP Values defined in [I-D.mglt-ipsecme-ts-dscp].

Ports and IP addresses and ports are defined as range and compressed using the LSB. For a range defined by a start and end value, let define msb(start, end) the function that returns the MSB that remains unchanged while the value evolves between start and end. Similarly, let define lsb(start, end) the function that returns the LSB that change while the value evolves between start and end. Fnally, let's consider len(x) the function that returns the number of bits of the bit array x.

We note for convenience:

*msb(ip_src) = msb(ts_ip_src_start, ts_ip_src_end) the MSB bits of the IP address range.

*msb(ip_dst) = msb(ts_ip_dst_start, ts_ip_dst_end) the MSB bits of the IP address range.

*lsb(ip_src) = msb(ts_ip_src_start, ts_ip_src_end) the LSB bits of the IP address range.

*lsb(ip_dst) = msb(ts_ip_dst_start, ts_ip_dst_end) the LSB bits of the IP address range.

*msb(port_src) = msb(ts_port_src_start, ts_port_src_end) the MSB bits of the source port range.

*msb(port_dst) = msb(ts_port_dst_start, ts_port_dst_end) the MSB bits of the destination port range.

*lsb(port_src) = msb(ts_port_src_start, ts_port_src_end) the LSB bits of the source port range.

*lsb(port_dst) = msb(ts_port_dst_start, ts_port_dst_end) the LSB bits of the destination port range.

Protocol IDs and DSP are defined as list of non consecutive values. A target value is defined when the list contains a single element.

4. New SCHC Compression / Decompression Actions (CDA)

In addition to the Compression / Decompression Action defined in [RFC8724], Section 7.4, this specification uses the CAD as presented in Figure 4. These CDA are either refinement of the compute- * CDA or result in a combination CDA and are mostly used for convenience.

Action	Compression	Decompression
lower	elided	Get from lower layer
checksum	elided	Compute checksum
padding	elided	Compute padding

Figure 4: EHC ESP related parameter

More specifically, when the list contains 0 or a single element, that value can be decompressed without ambiguity and as such an index does not need to be sent. When more than one value is present in the list, the index needs to be sent.

lower: designates an action where the compression consists in eliding 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.

checksum: designates an action where the compression consists in eliding 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.

padding: designates an action where the compression consists in eliding the padding field. The decompression consists in re-computing the padding field as described in ESP [RFC4303].

5. Clear Text ESP Compression / Decompression

The Clear Text ESP Compression is performed on the ESP fields not yet encrypted, that is the ESP Payload Data, the ESP padding field,

the Pad Length field as well as the Next Header field which indicates the type of the inner packet.

When ipsec_mode is set to "Transport", the Clear Text ESP packet that corresponds to an IPv4 packet will have the Payload Data set to the IPv4 Payload and the Next Header set to the Protocol ID - that is typically UDP, TCP or SCHC when the payload results from an SCHC compression. The Clear Text ESP packet that corresponds to an IPv6 packet will have the Payload Data set may include some IPv6 extensions that precede the IP payload. In that case, the Next Header will have the value that corresponds to that first IPv6 extension being encrypted.

When ipsec_mode is set to "Tunnel", the Clear Text ESP packet has the Payload Data set to the IP packet with the Next Header field indicating whether this is an IPv4, an IPv6 or an SCHC packet..

SA are unidirectional and the Direction Indicator (DI) reflects that direction and is set to Up for outbound SA and Down for inbound SA. Fields that are not compressed have no Target Value (TV), their Matching Operator (MO) is set to ignore and Compression/ Decompression Actions (CDA) to "value-sent". Unless specified the Field Position (FP) is set to 1.

Note that for both the IP payload and the IP header, some fields are Compressed / Decompressed independently of the value of Traffic Selectors EHC Context, while some other fields require the Traffic Selectors to be expressed under a specific format.

5.1. Inner Packet Payload Compression

An SCHC payload is not compressed.

If the inner IP payload is an UDP or TCP packet the checksum is elided. For both TCP or UDP, FL is set to 16 bit, TV is not set, MO is set to "ignore" and CDA is et to "checksum". This may result is decompressing a zero-checksum UDP packet with a valid checksum, but this has no impact as valid checksum are universally accepted.

If the inner packet is an UDP or UDP-Lite the length field is elided. FL is set to 16, TV is not set, MO is set to "ignore" and CDA is set to "lower" as the length field of the decompressed UDP packet is expressed in bytes and is derived from the length of the compressed UDP packet by adding the 16 bit UDP Checksum, the 16 bit UDP Length field as well as the respective length of the respective source MSB port and destination MSB ports.

```
UDP.Length = ( len( compressed UDP) + 16 + 16 + len( lsb( port_src ) ) \
               + len( lsb( port_src ) ) ) / 8
```

Note that for each SA, LSB and MSB are of fixed length. When the port has a single value this is equivalent to TV containing the port value, MO is set to "equal" and CDA set to not_sent.

5.2. Inner IPv4 Compression

When ts_ip_src/dst range is defined and ts_ipversion is set to "IPv4", IPv4 addresses of the inner IP packet are compressed. FL is set to 32, TV to msb(ip_src) or msb(ip_dst), the MO is set to "MSB" and the CDA is set to "LSB".

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".

The Protocol field sets FL to 8 bits. If ts_proto_list contains the value 0, TV is not set, MO is set to ignore and CDA is set to "value-sent". If "proto_id" does not contain 0 and the list contains less or exactly 1 value, TV is set to that value, MO is set to "equal" and CDA is set to "mot-sent". In any other case, TV is set to the proto_list, MO is set to "match-mapping" and CDA is set to "mapping-sent".

The IPv4 TTL field is derived from the IPv4 TTL field of the outer IPv4 address or the IPv6 Hop limit. FL is set to 8 bits, TV is omitted, MO is set to ignore and CDA is set to lower.

The IPv4 Total Length is elided. FL is set to 16 bits, TV is not set, MO is set to "ignore" and CDA is set to "lower".

DSP, ECN are either retrieved from the SA or from the outer IP header. Fl is set to 8. When the DSP, ECN are defined by the SA via [I-D.mglt-ipsecme-ts-dscp] and ts_dsp_list contains a single element, TV is set to that element MO is set to "equal" and CDA is set to "not-sent". When the DSP, ECN are defined by the SA via [I-D.mglt-ipsecme-ts-dscp] and ts_dsp_list contains more than one element, TV is set to the list, MO is set to "match-mapping" and CDA is set to "mapping-sent". When the DSP, ECN are not defined by the SA, MO is set to "ignore" and the CDA is set to "lower".

When ts_ip_version can be inferred from the ts, the IP version is elided. FL is set to 4 bits, the TV is set to ts_ip_version, MO is set to "equal" and CDA to "not-sent".

When the inner IP address has the same version as the outer_ip and ts_traffic_flow is defined and set to True, the Identification field of the IPv4 inner packet or the Traffic Flow field of the IPv6 packet is elided and read from the outer IP address field. For IPv4, FL is set to 16 bits, TV is ignored, MO is set to "ignore" and CDA is set to "lower". For IPv6, FL is set to 20 bits, TV is ignored, MO is set to "ignore" and CDA is set to "lower".

When the inner is IPv4 and the outer IP is IPv6 and ts_traffic_flow is set to True, the LSB 16 bits of the outer IP address are considered. This results in a lossless compression. When the inner is IPv6 and the outer IP is IPv4 and ts_traffic_flow is set to True, the LSB 16 bits of inner Traffic Flow fields are set to the outer Identification field and the remaining 4 MSB bits are set to 0. Such compression is not lossless and needs to be considered cautiously. Note that the Flow Label of the inner packet arriving at the destination may have another value than the initial Flow Label. However, the Flow Label value set at the source ends up with the same value at the destination, with of course a lower entropy.

5.3. Inner IPv6 Compression

The compression / decompression of the IPv6 fields are compressed / decompressed in a similar way as in IPv4 (see Section 5.2). IPv6 addresses are compressed decompressed as IPv4 addresses except that FL is set to 128. IPv6 Hop limit is compressed / decompressed as the IPv4 TTL field. The last Next Header with a transport protocol value is compressed / decompressed as IPv4 Protocol field. The Total Length is compressed / decompressed similarly to the IPv4 Length except that the IPv6 length includes the IPv6 header. Traffic Class is compressed / decompressed similarly to the DSP,ECN field. IP version is compressed / decompressed as in IPv4. The Traffic Flow field is compressed / decompressed similarly to the IPv4 Identification field except that FL is set to 20 bits.

5.4. ESP Compression

When ipsec_mode is set to "Tunnel" and ts_ip_version can be determined, the Next Header Field is elided. FL is set to 8 bits, TV is set to IPv4 or IPv6 depending on the ts_ip_version, MO is set to "equal" and CDA is set to "not-sent".

If the esp_encr does not require a specific block size, Padding and Pad Length are elided. FL is defined by the type that is to (Pad Length + 1) * 8 bits, TV is unset, MO is set to "ignore" and CDA is set to padding.

Encryption may require require the clear text to respect a given size block. In addition, IP networking may also require a special alignment which is 32 bits by default for IPv6 Extensions, but may also be overwritten by the EHC Context. The Padding is defined by pad_value and pad_size appended to the clear text payload - similarly to what ESP does with Padding and Pad Len. An 8 bit alignment is interpreted by SCHC as a Word of 8 bits, and a 32 bit alignment is interpreted as a Word of 32 bits. The padding size pad_size is defined by the alignment and set to 3 bits for an 8 bit alignment (2**3) and 5 bits for 32 bit alignement (2**5). If pad

designates the number of bits to be padded, the pad value is set to pad_value = (pad + len(pad_size) % Word. This results in an additional pad_value + pad_size bits.

6. Encrypted ESP Compression

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".

If the esp_encr considers implicit IV [RFC8750], Sequence Number are not compressed. Otherwise, SN are compressed to their LSB similarly to the SPI. 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".

Note that the use of implicit IV always result in a better compression as an 64 bit IV to be sent while compression of the SN alone results at best in a reduction of 32 bits.

The IPv6 Next Header field or the IPv4 Protocol that contains the "ESP" value is changed to "SCHC".

7. IANA Considerations

There is no IANA parameters to be registered.

8. Security Considerations

There is no specific considerations associated to the profile other than the security considerations of ESP [RFC4303] and those of SCHC [RFC8724].

9. Acknowledgements

We would like to thank Laurent Toutain for its guidance on SCHC. Robert Moskowitz for

10. References

10.1. Normative References

[I-D.mglt-ipsecme-ts-dscp] Migault, D., Halpern, J. M., Parkholm, U., and D. Liu, "Traffic Selector for Internet Key Exchange version 2 to add support Differentiated Services Field Codepoints (DSCP)", Work in Progress, Internet-Draft, draft-mglt-ipsecme-ts-dscp-03, 26 July 2023, <https://datatracker.ietf.org/doc/html/draft-mglt-ipsecme-ts-dscp-03>.

[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>.

[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>.

[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>.

10.2. Informative References

[I-D.mglt-ipsecme-ikev2-diet-esp-extension] Migault, D., Guggemos, T., and D. Schinazi, "Internet Key Exchange version 2 (IKEv2) extension for the ESP Header Compression (EHC)", Work in Progress, Internet-Draft, draft-mglt-ipsecme-ikev2-diet-esp-extension-03, 28 June 2023, <https://datatracker.ietf.org/doc/html/draft-mglt-ipsecme-ikev2-diet-esp-extension-03>.

[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>.

Appendix A. Illustrative Example

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 with a server, and sets a VPN to connect its secure domain - like a home gateway. The home gateway will be responsible to decompress the compress packet and provides interoperability with standard application server.

The EHC Context is defined as mentioned below:

- *alignment is set to 8 bits

- *ipsec_mode is set to "Tunnel"

- *tunnel_ip_srct is set to the IPv6_m, the IPv6 address of the mote.

- *tunnel_ip_dst is set to IPv6_gw, the IPv6 of the security gateway.

- *esp_spi is agreed by the IKEv2.

- *esp_spi_lsb is set to 0 as IPv6_m provides sufficient context to associate the right SA.

- *esp_sn results from the standard IPsec, and not impacted.

- *esp_sn_lsb is set to 2 even though we are considering AES-CCM_8_IIV [RFC8750] which uses the ESP Sequence Number to generated the IV. This results in a 8 bytes reduction compared to the AES-CCM_8 [RFC4309].

- *esp_encr is configured with AES-CCM_8_IIV [RFC8750]. This cipher suite does not require a block size and so no padding is required and does not support SN compression.

- *ts_flow_label As the inner traffic and the encrypted traffic are very correlated, it makes sense to re-use the flow label and ts_flow_label is set to True.

- *ts_ip_version is set to IPv6.

- *ts_ip_src_start is set to IPv6_m. In this example, the SA is associated to messages sent by the mote to the application server (IPv6_server)

- *ts_ip_src_end is set to IPv6_m

*ts_ip_dst_end the IPv6 address of the application server (IPv6_server).

*ts_ip_dst_end IPv6_server

*ts_proto_list [UDP], in the case of a very constraint mote, only UDP messages are considered.

*ts_port_src_start port_m. The mote and the application server are using dedicated ports.

*ts_port_src_end port_m. The mote and the application server are using dedicated ports. The use of a specific single port enables their elision.

*ts_port_dst_end port_server

*ts_port_dst_end port_server

*ts_dsp_list [0] the default standard value, we MAY assume that value has been negotiated via IKEv2 or that it as been set as the default value left to the lower layers.

Figure 5 illustrates an UDP packet being protected by ESP in the tunnel mode using AES-CCM_8_IIV. This packet is compressed as depicted in Figure 6.
EHC reduces the packet size by 53 bytes.

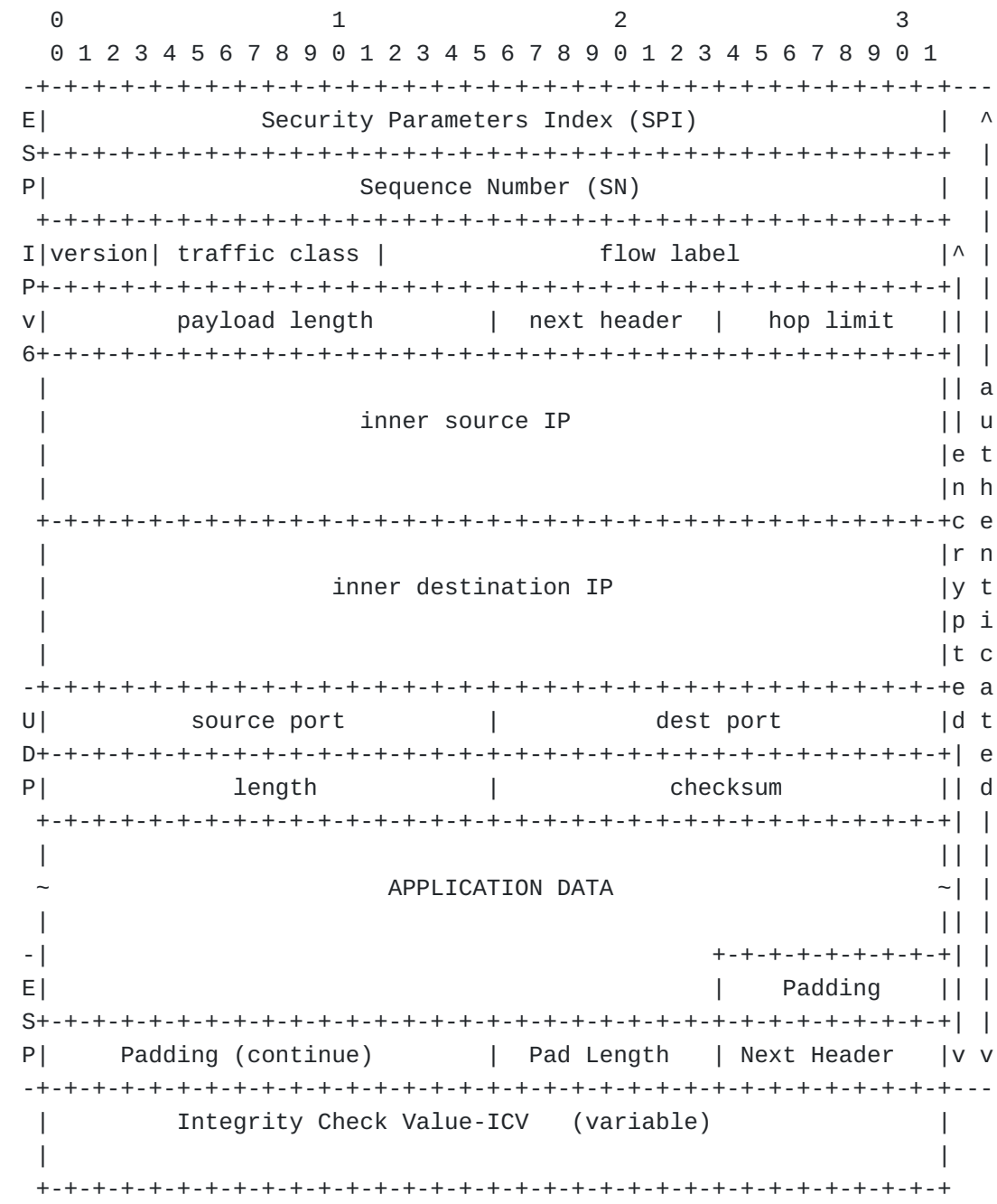

Figure 5: Standard ESP packet for IoT UDP in Tunnel mode more with AES-CCM_8_IIV

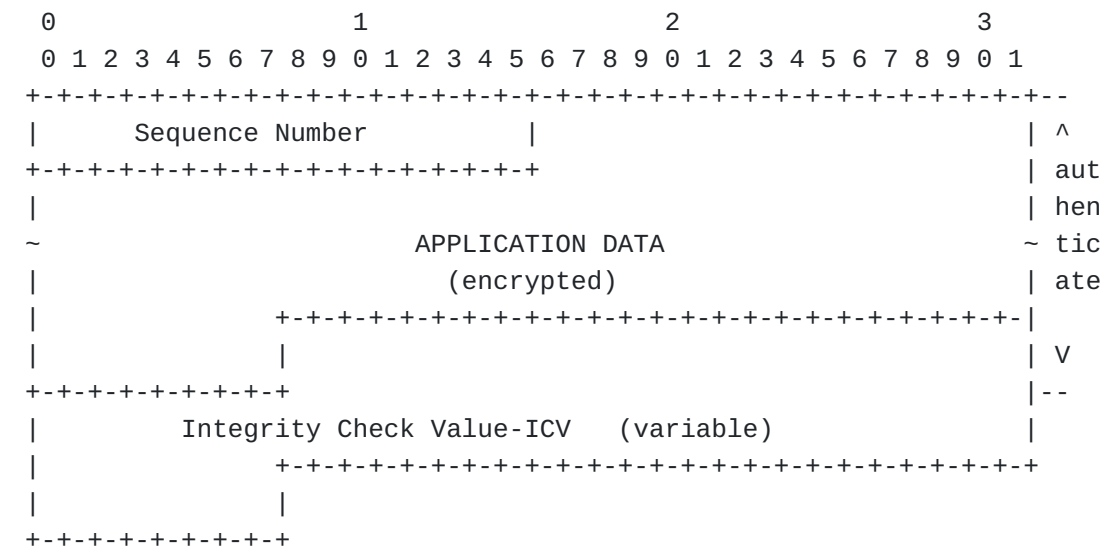

Figure 6: EHC ESP packet for IoT UDP in Tunnel mode more with AES-CCM_8_IIV

A.2. Single TCP session IoT VPN

This section is very similar to Appendix A.1 except that a TCP session is used instead.

The compression on the TCP payload is very limited, and in a case where the TCP end point is the same as the ESP end point additionnal compression could be performed. Additional fields such as TCP options, urgent pointers, the SN and ACK Number could be compressed by a specific profile agreed at the TCP level as opposed to the ESP level.

The ESP encapsulated TCP packet described in Figure 7 is compressed by EHCP using th esam eEHCP context as in Appendix A.1 and EHCP reduces that packet by 55 bytes, as depicted in Figure 6.

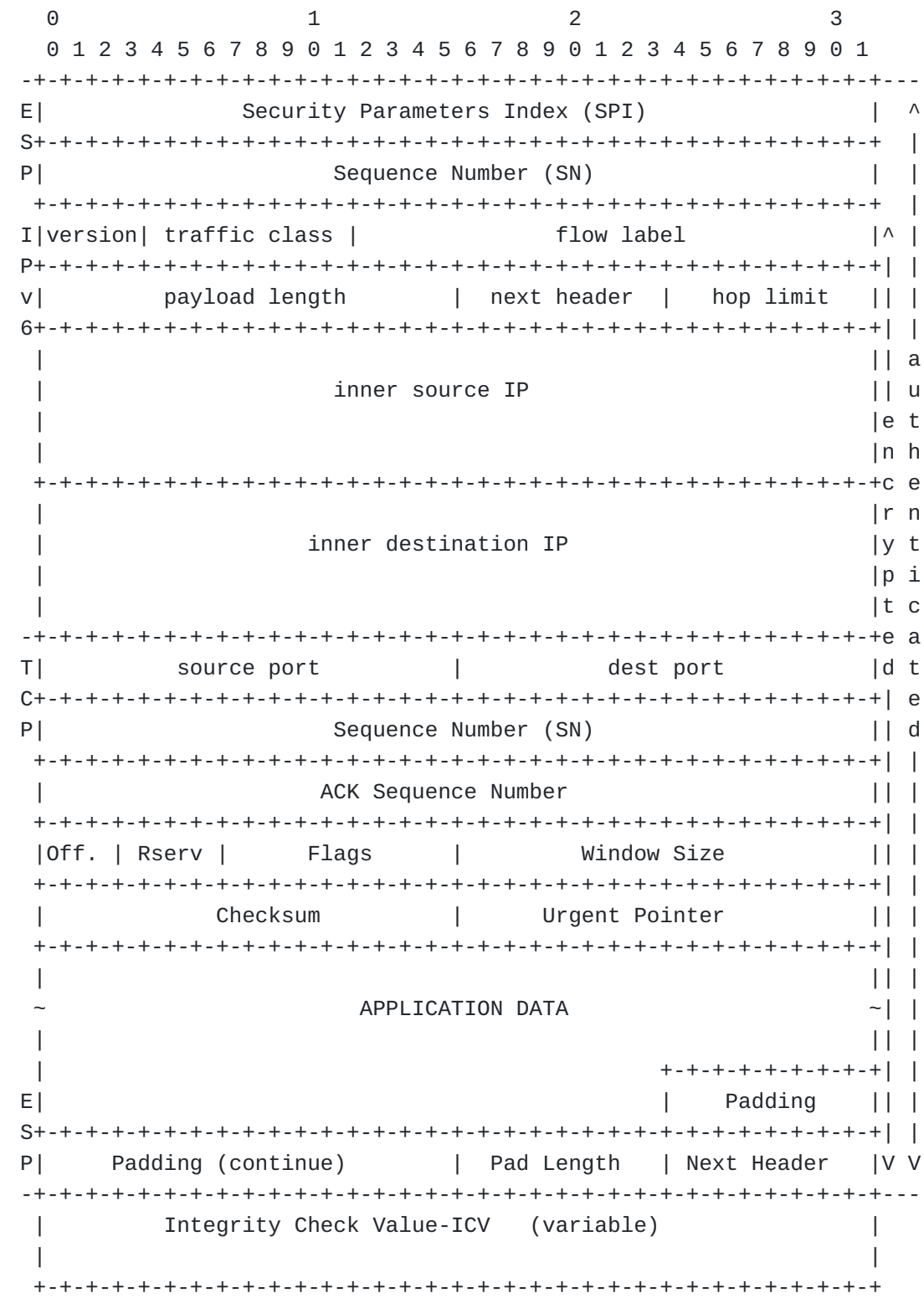

Figure 7: Standard ESP packet for IoT TCP in Tunnel mode more with AES-CCM_8_IIV

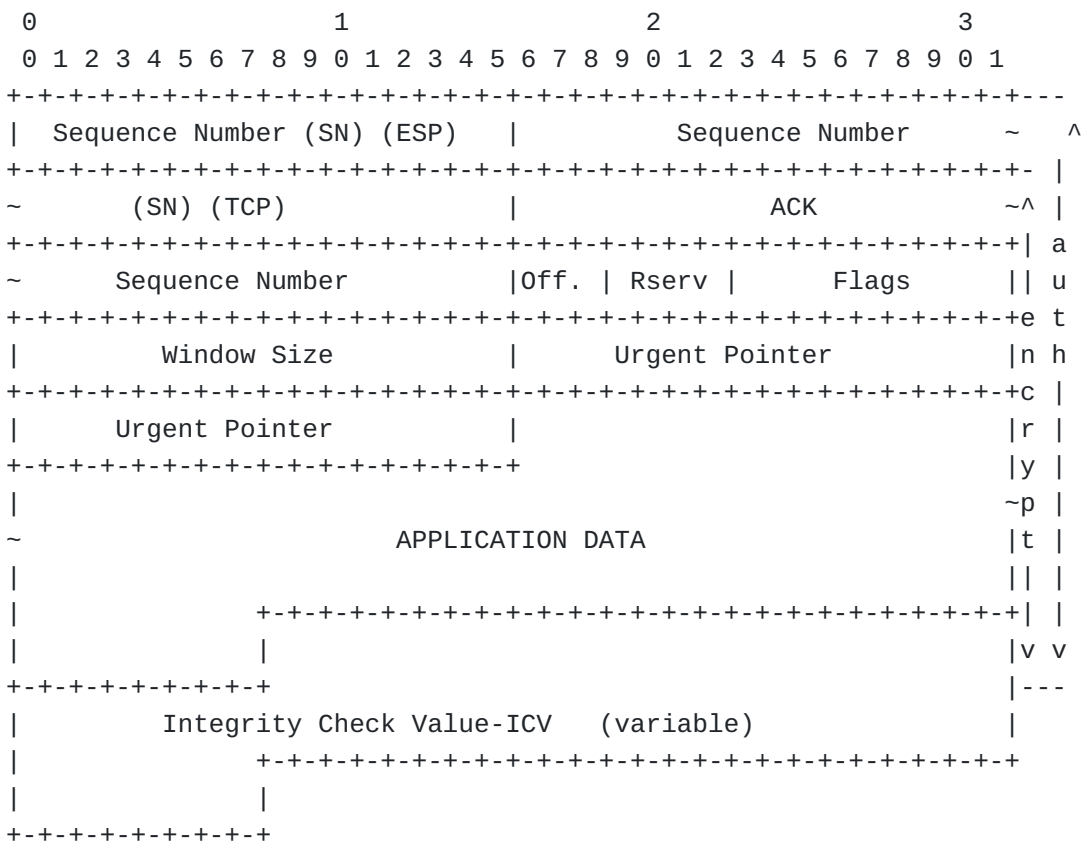

Figure 8: EHC ESP packet for IoT TCP in Tunnel mode more with AES-CCM_8_IIV

A.3. Traditional VPN

This section illustrates the case of an company VPN that allows web browsing. The VPN is typically set by a remote host that forwards all its traffic to the security gateway.
In this case, the SA does not specify the protocol (TCP and UDP packet can be sent), nor the ports. Regarding ports it could be possible to restrict the user to only use high range ports (0 - 2 ** 10) - especially if the VPN is only supporting web browsing - but we did not consider this in this example. The destination IP address is also expect to take any value, while the IPv6 source in the case of a road warrior scenarios us expected to take a single value. We consider the VPN client is using an IPv4 or an IPv6 address.
Regarding ESP, we considered the VPN client is using AES-GCM_16, though AES-GCM_IIV would be the RECOMMENDED transform. The VPN client is also expected to have a reasonably low throughput which enables the SN to be coded over 16 bits as opposed to 32 bits. Similarly, the number of connection is expected to remain sufficiently low so that a 16 bit SPI remains sufficient.

The EHC Context is defined as mentioned below:

- *alignment is set to 8 bits

- *ipsec_mode is set to "Tunnel"

- *tunnel_ip_src is set to the IPv6_user, the IPv6 address of the mote.

- *tunnel_ip_dst is set to IPv6_gw, the IPv6 of the security gateway.

- *esp_spi: is agreed by the IKEv2.

- *esp_spi_lsb: is set to 2 bytes.

- *esp_sn: results from the standard IPsec, and not impacted.

- *esp_sn_lsb: is set to 16 bits. Note that such compression is possible since AES-GCM_16 is used instead of AES-GCM_16_IIV. While this results in better performances for EHC, it is not an optimal choice as IIV transforms results always in better comprehensions.

- *esp_encr: is configured with AES-GCM_16 [RFC8750].

- *ts_flow_label: is set to True, note as the outer IP address is IPv6, the compression is lossless.

- *ts_ip_version: is set not set as the VPN user can use either an IPv4 or an IPv6 address.

- *ts_ip_src_start: is set to IPv6_user or IPv4_user. Note that the version can be inferred by the Next Header, and the version can deterministically determine the IP in use.

- *ts_ip_src_end: is set to IPv6_user or IPv4_user

- *ts_ip_dst_end: IP destination is set to take any value, so the range is unspecified and the start/ end addresses are undefined.

- *ts_ip_dst_end: undefined.

- *ts_proto_list: undefined

- *ts_port_src_start: undefined.

- *ts_port_src_end: undefined.

- *ts_port_dst_end: undefined

*ts_port_dst_end: undefined

*ts_dsp_list: [0] the default standard value, we MAY assume that value has been negotiated via IKEv2 or that it as been set as the default value left to the lower layers.

A.3.1. IPv6 in IPv6

Figure 9 represents the original ESP TCP packet with IPv6 inner IP addresses and Figure 10 represents the corresponding packet compressed with EHC.

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

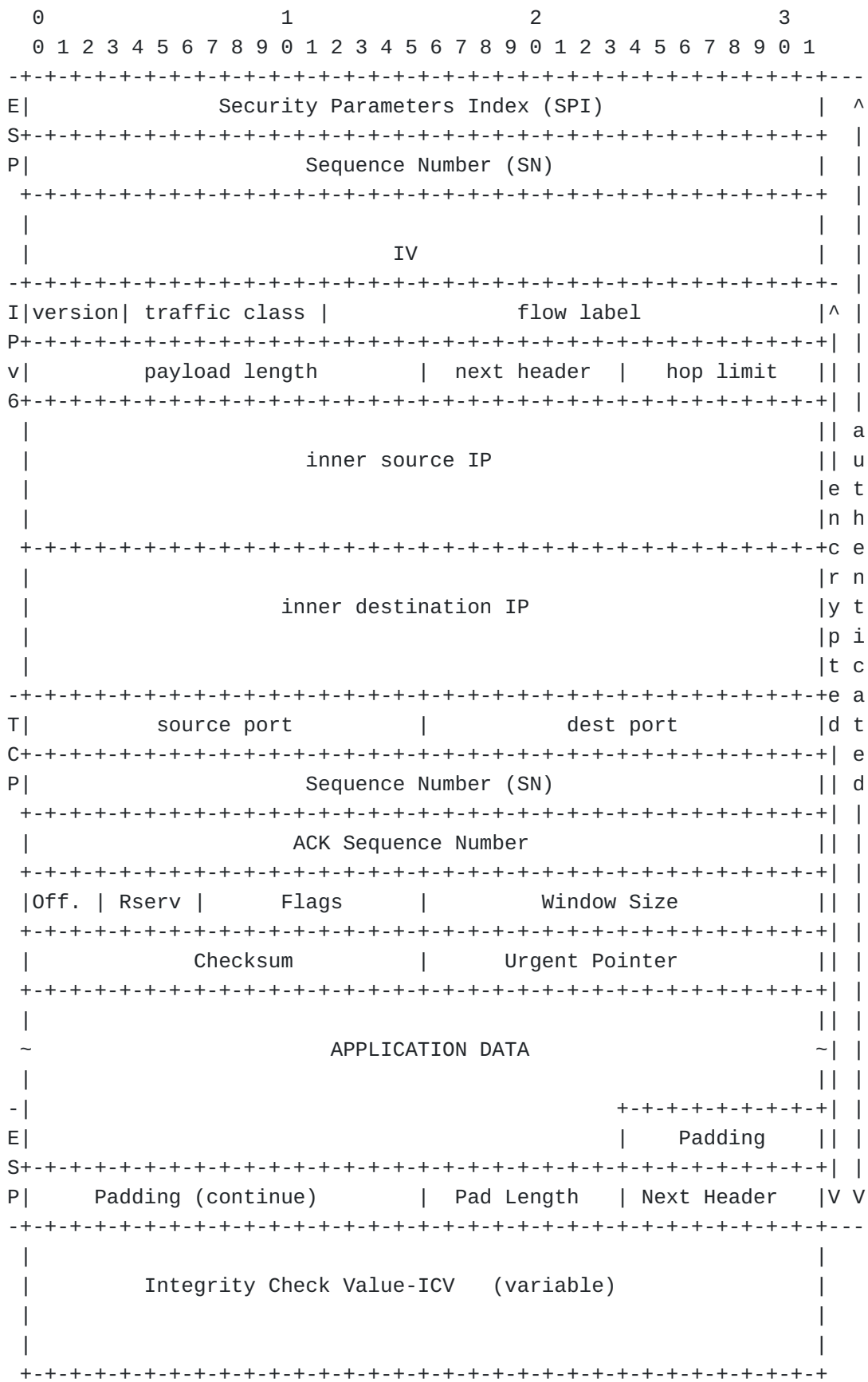

Figure 9: Standard ESP packet for VPN traffic mode with AES-GCM_16

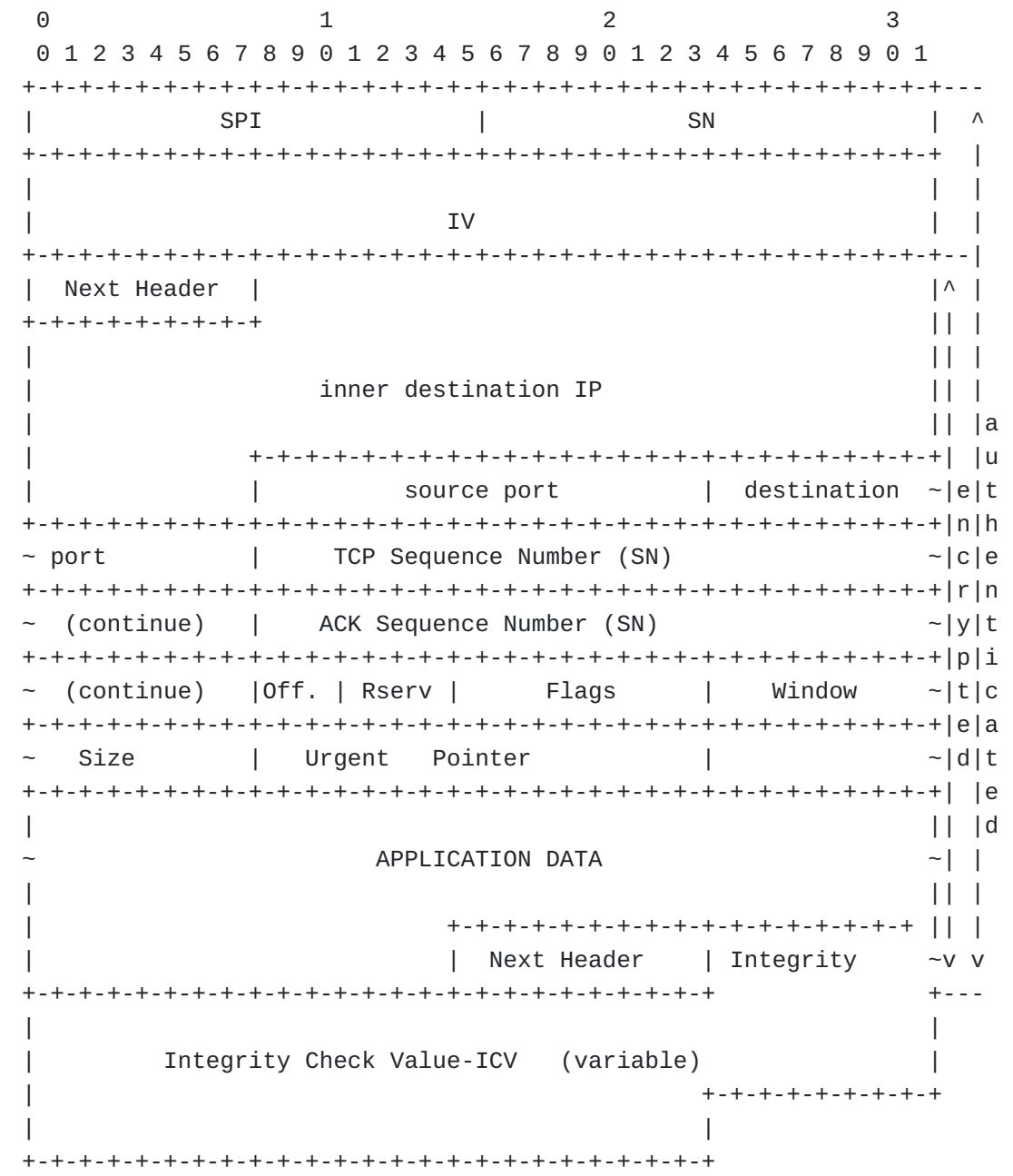

Figure 10: Compressed IPv6 in IPv6 ESP packet for VPN traffic mode with AES-GCM_16

A.3.2. IPv6 in IPv4

For IPv6 in IPv4, the compression is similar when ts_traffic_flow is set, otherwise these 20 bits needs to be provided explicitly.

When ts_traffic_flow is set to True, the resulting decompressed IPv6 packet will be as follows (see the flow label field):

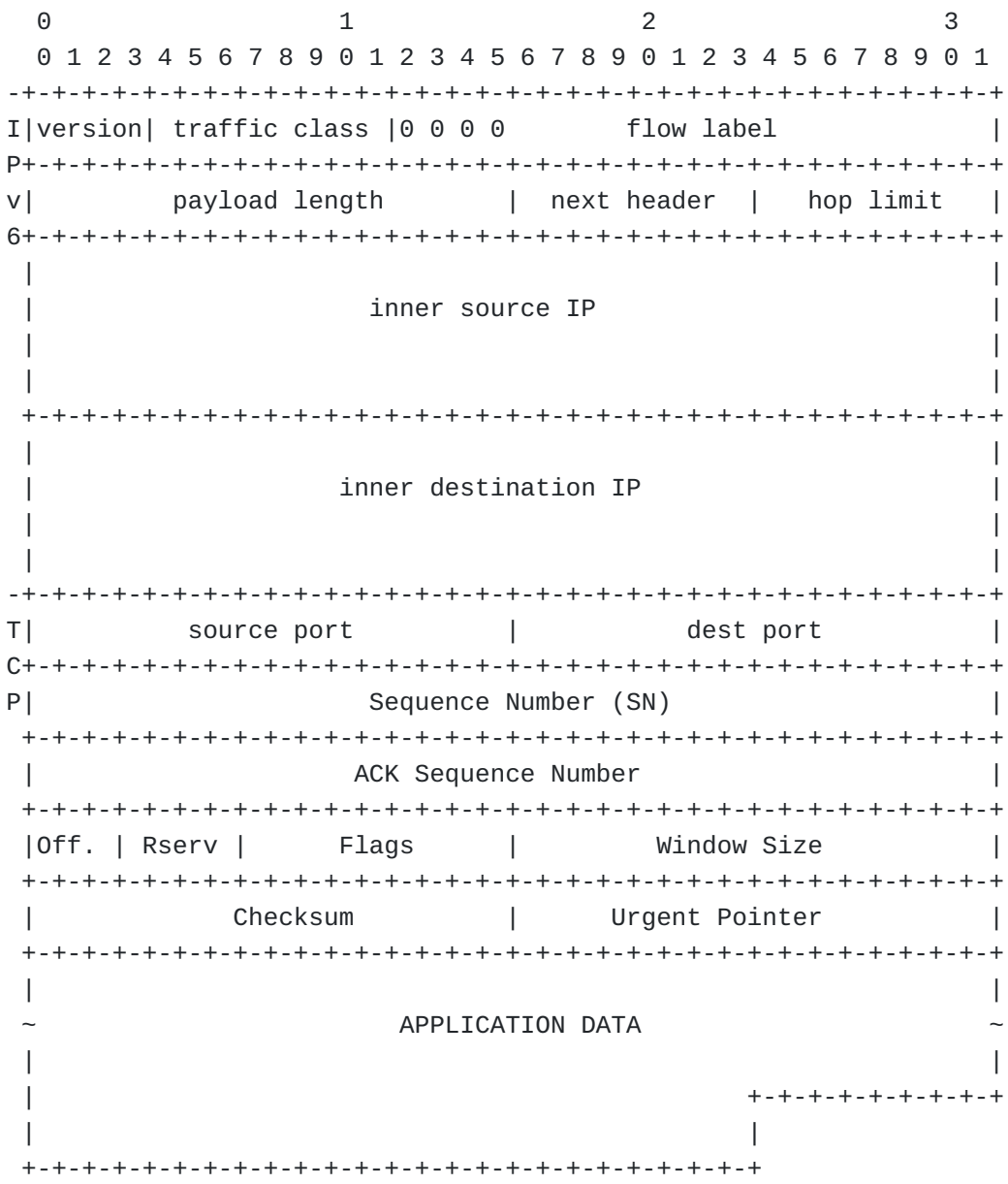

Figure 11: Decompressed IPv6 encapsulated in IPv4 ESP packet

A.3.3. IPv4 in IPv4

Figure 12 represents the original ESP TCP packet with IPv6 inner IP addresses and Figure 13 represents the corresponding packet compressed with EHC.

The compression with Diet-ESP results in a reduction of 24 bytes.

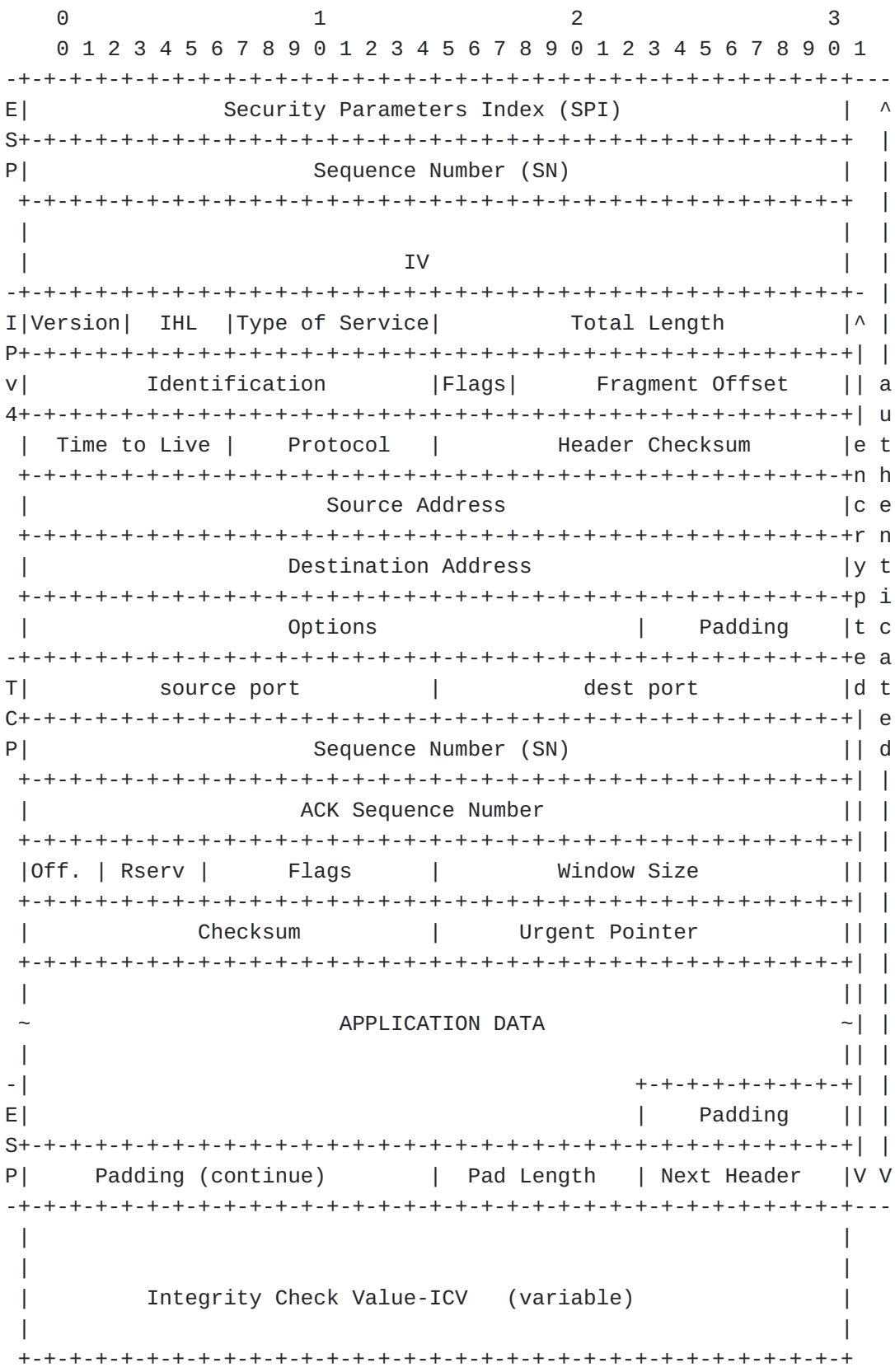

Figure 12: Standard IPv4 in IPv4 ESP packet for VPN traffic mode with AES-GCM_16

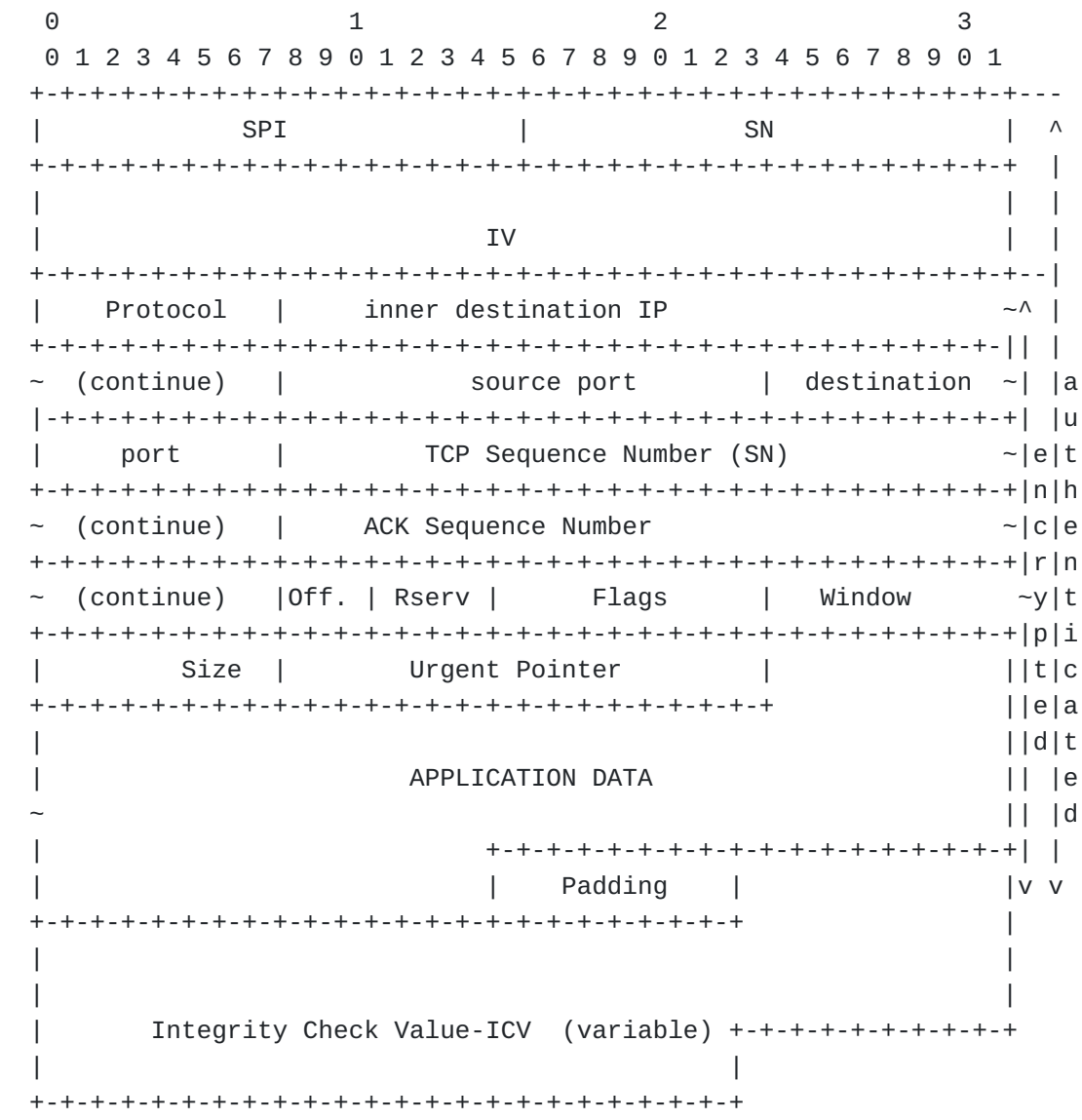

Figure 13: Compressed IPv4 in IPv4 ESP packet for VPN traffic mode with AES-GCM_16

A.3.4. IPv4 in IPv6

TBD

Authors' Addresses

Daniel Migault
Ericsson

Email: daniel.migault@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