IPSECME
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2015

D. Migault, Ed.
Orange
T. Guggemos, Ed.
Orange / LMU Munich

July 2, 2014

# Diet-ESP: a flexible and compressed format for IPsec/ESP
draft-mglt-ipsecme-diet-esp-01.txt

## Abstract

IPsec/ESP secure every single IP packets exchanged between two nodes. This makes security transparent to the applications, as opposed to TLS or DTLS for example.

IPsec/ESP has not widely been used to secure application because IPsec is implemented in the kernel space, and IPsec/ESP security rules are defined on the device -- similarly to firewall. In addition, IPsec/ESP introduces network overhead on an IP packet basis, as opposed as TLS/DTLS that introduces network overhead on an UDP or TCP segment basis. This mostly impacts devices that do not perform IP fragmentation.

Such drawbacks are not anymore valid for IoT, and the IPsec/ESP may even better fits IoT usage and security requirements. IoT device are usually hardware dedicated for a given task or a given application which makes Kernel / user land split less significant. IoT devices send data that is most likely expected to fit in a single IP packet. Eventually, configuring IPsec/ESP security rules provides the ability to enforce the security of the device, as security is not handled on a per-application basis. Then the database structure of the IPsec/ESP security policies perfectly match sleeping nodes.

This document defines Diet-ESP that adapts IPsec/ESP for IoT. The goal of Diet-ESP is to reduce the size of the IPsec/ESP packet sent on the wire. As a result Diet-ESP is expected to compress traditional IPsec/ESP packet without impacting the security provided by 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 http://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 January 3, 2015.



Table of Contents

## 1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 2. Introduction

### 2.1. IoT context

The IPsec/ESP [RFC4303] is represented in Figure 1. It was designed to: 1) provide high level of security as a basis, 2) favor interoperability between implementations 3) scale on large infrastructures.

In order to match these goals, ESP format favor mandatory fields with fixed sizes that are designed for the worst case scenarios. This results in a kind of "unique" packet format common to all considered scenarios using ESP. These specific scenarios may result in carrying "unnecessary" or "larger than required" fields. This cost of additional bytes were considered as negligible versus interoperability, making ESP very successful over the years.

With IoT, requirements become slightly different. For most devices, like sensors, sending extra bytes directly impacts the battery and so the life time of the sensor. As a result, IoT may look at reducing the number of bytes sent on the wire. As sensors may belong to different specific network topologies, compression of the IPsec/ESP

packet may differ from one network to the another. The use of different compressed IPsec/ESP packets may increase the code complexity of Diet-ESP versus the standard IPsec/ESP. Code complexity may directly impacts interoperability. In addition, in order to reduce the amount of code embedded on the sensors, some sensors may only embed the code associated to a specific compressed IPsec/ESP packet format. This may also impact interoperability between these specific sensors. The fact that Diet-ESP, more specifically IPsec/ESP compression may limit any sensor-to-any-sensor IPsec/ESP communication is not an issue. First, it is mainly an implementation issue, not a protocol issue, as implementation design may prefer limiting the code embedded on the device vs. interoperability. Secondly, very constrained devices are more likely to be connected to an IPsec/ESP Security Gateway. In this document, we consider that interoperability is provided as long as a generic Security Gateway is able to set a secure connection with any IPsec/ Diet-ESP or IPsec/ESP sensor.

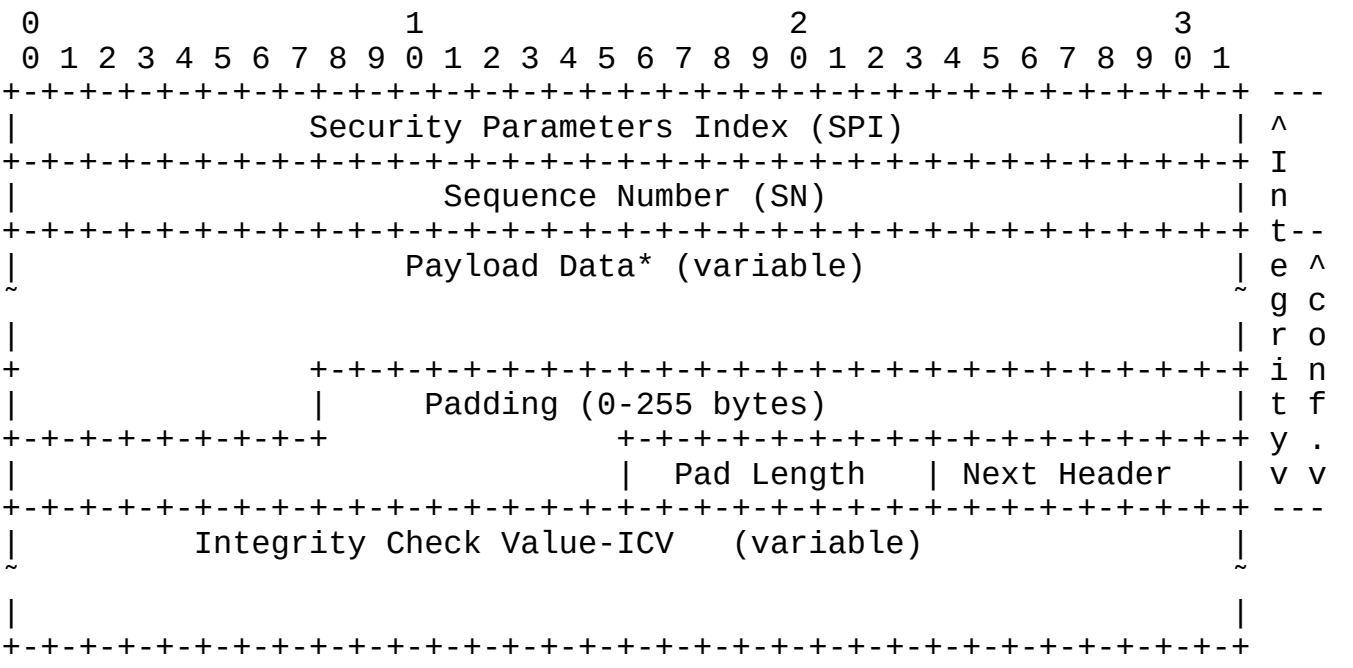


Figure 1: ESP Packet Description

## 2.2. Position of Diet-ESP suite documents toward IPsec

This document is part of the Diet-ESP document suite that addresses the use of IPsec/ESP for IoT. Requirements for IPsec/ESP for IoT are expressed in [draft-mglt-ipsecme-diet-esp-requirements].

Standard IPsec/ESP [RFC4303] defines a protocol and a packet format that carries an encrypted Payload Data. More specifically, the

Payload Data is placed between an ESP Header and an ESP Trailer. Eventually, an Integrity Check Value (ICV) is appended to the ESP Trailer to authenticate the packet. The encryption algorithm (like [RFC3686] and [RFC3602]) defines how to encrypt the concatenation of the Payload Data and the Trailer. In fact, encryption or decryption is performed using a shared secret key as well as additional data such as the Initialization Vector (IV). AES-CTR [RFC3686] and AES-CBC [RFC3602] both define protocols to encrypt or decrypt the IV, the Payload Data and the Trailer: the Encrypted Payload. In order to agree on cryptographic material -- like the shared key --, the encryption mode -- like AES-CTR or AES-CBC -- and the protocol -- ESP -- peers use IKEv2 [RFC5996].

This document [draft-mglt-ipsecme-diet-esp] describes Diet-ESP which compresses fields of the Standard ESP [RFC4303]. [draft-mglt-ipsecme-diet-esp-iv] describe how to compress the IV embedded in the Encrypted Payload. [draft-mglt-diet-esp-inner-compression] addresses the compression of the Clear Text Data, that is the Payload Data before compression.

[draft-mglt-ipsecme-diet-esp-ikev2-extention] defines how all necessary parameters for Diet-ESP can be negotiated using IKEv2.

## 2.3. Document Overview

This document describes how to compress ESP fields sent on the wire. Concerned fields are those of the ESP Header, the ESP Trailer and the ICV as represented in Figure 1. Compression of the Payload Data, including the IV is out of scope of the document.

The compression mechanisms defined in this document are based on ROHC [RFC3095], [RFC5225] and ROHCoverIPsec [RFC5856], [RFC5857], [RFC5858].

ROHC defines mechanisms to compress/decompress fields of an IP packet. These compressors are placed between the MAC layer and the IP layer. In the case of ESP, ROHC can be used to compress/ decompress SPI, SN. However, ROHC cannot be used to compress encrypted fields like Padding, Pad Length, Next Header -- and later the Clear Text Data before encryption. In fact, at the MAC layer, these fields are encrypted and their encrypted value is used generate the ICV. As a result, compression an ESP packet at the MAC layer requires to decrypt the packet to be able to compress fields like Clear Text Data, Pad Length in order to be able to eventually remove the Padding Field. Similarly, decompressing a compress ESP packet at the MAC layer would require to decrypt the received packet, decompress the packet the Clear Text Data as well as the other ESP fields, before forwarding the ESP packet to the IP stack. Note that

in some case decompression is not feasible. Consider for example an ESP implementation that generates a random Padding. If this field is removed by the compressor, it can hardly be recovered by the decompressor. Using a different Padding field would result is ESP rejecting the packet as the ICV check will not succeed. As a result ROHC cannot be used alone.

On the other hand, ROHCoverIPsec makes compression possible before the ESP payload is encrypted, and so the Clear Text Data can be compressed, but not the ESP related fields like Padding, Pad Length and Next Header.

ROHC and ROHCoverIPsec have been designed for bandwidth optimization, but not necessarily for constraint devices. As a result, defining ROHC and ROHCoverIPsec profiles is not sufficient to fulfill the complete set of Diet-ESP requirements listed in [draft-mglt-ipsecme-diet-esp-requirements]. In fact Diet-ESP MUST result in an light implementation that does not require implementation of the full ROHC and ROHCoverIPsec frameworks.

In order to achieve ESP field compression, this document describes the Diet-ESP Context. This context contains all necessary parameters to compress an ESP packet. This Diet-ESP Context can be provided as input to proceed to Diet-ESP compression / decompression. This document uses the ROHC and ROHCoverIPsec framework to compress the ESP packet. The advantage of using ROHC and ROHCoverIPsec is that compression behavior follows a standardized compression framework. On the other hand, ROHC and ROHCoverIPsec frameworks are used in a stand alone mode, which means no ROHC communications between compressor and decompressor are considered. This enables specific and lighter implementations to perform Diet-ESP compression without implementing the ROHC or ROHCoverIPsec frameworks. All Diet-ESP implementations only have to agree on the Diet-ESP Context to become inter-operable.

The remaining of the document is as follows. Section 4 described the Diet-ESP Context. Section 5 describes how the parameters of the Diet-ESP Context are used by the ROHC and ROHCoverIPsec framework to compress the ESP packet. This requires definition of new profiles and extensions. Section 6 describes the interactions of Diet-ESP interacts with other compression protocols such as 6lowPAN and ROHC compression for other protocols than ESP. Finally, Section 7 describes how Diet-ESP matches the requirements for Diet-ESP [draft-mglt-ipsecme-diet-esp-requirements]. Appendix A is an informational section that illustrates how an minimal Diet-ESP implementation may be used in IoT devices. Appendix B lists the differences between Diet-ESP and Standard ESP.

## 3. Terminology

This document uses the following terminology:

- IoT: Internet of Things

- LSB: Last Significant Bytes

- IP alignment: The necessary alignment for IPv4 (32 bits) resp. IPv6 (64 bits)

- Clear Text Data: designates the original data that are carried by ESP.

- Encrypted Payload: carries the encrypted Data Payload including cryptographic material like the IV and the ESP Trailer.

## 4. Diet-ESP Context

The Diet-ESP context provides the necessary parameters for the compressor and decompressor to perform the appropriated compression and decompression of the ESP packet. Table 1 in Section 4.1 describes the different parameters. Section 4.2 describes the Diet-ESP Context that makes Diet-ESP compliant with ESP. Finally, Section 4.3 provides the default Diet-ESP Context. This describes default values when not explicitly set by the developer. Motivation for default values is to make the use of Diet-ESP simple for developers while still providing a secure framework for IoT communications. It is also expected to simplify negotiation of a Diet-ESP Context between peers.

### 4.1. Description

| Context Field Name | Overview |
|---|---|
| ALIGN | Necessary Alignment for the specific device. |
| SPI_SIZE | Size in bytes of the SPI field in the sent packet. |
| SN_SIZE | Size in byte of the SN field in the sent packet. |
| NH | Presence of the Next Header field in the ESP Trailer. |
| PAD | Presence of the Pad Length field present in the ESP trailer. |
| Diet-ESP_ICV_SIZE | Size of the Diet-ESP ICV in the sent packet. |

Table 1: Diet-ESP Context.

ALIGN:

Alignment is the minimum alignment accepted by the hardware. Constrains may come from various reasons. Hardware may have some specific requirements, but also operating systems. For most servers CPU and OS have been designed with 32 bit or 64 bit alignment. As a result, IP headers have been standardized with 32 bits (resp. 64 bits in IPv6) alignment for each IP extension header. ESP is one of these extension headers with an Header (SPI and SN) of 64 bits and the Trailer (NH, PL, PAD) of (2 + PL) bytes. Since the trailer is part of the ESP extension header, it must provide the necessary padding for a correct alignment of the NH field to 32 (resp. 64) bits. The alignment may also be relevant if Block-Ciphers like AES-CBC needs an aligned payload to perform the encryption.

Diet-ESP reduces the ALIGN value from 32 bits for IPv4 or 64 bits for IPv6 to 8, 16, 32 and 64 bit alignment.

Motivations to do so is to remove the Padding and other mandatory fields of the ESP packet. Then, most IoT embeds small 8 or 16 bit CPUs. Finally, even though ESP is an extension header, it is often the last extension header of a header-only IP packet. The ESP header is only read by the real receiver and is uninteresting for other devices like routers, placed between the to peers. As a result, there seems no real impact on the system if ESP extension header is not aligned.

Note that the benefices of ALIGN also depends on the used cryptographic mode. More specifically AES-CTR has a 8 bit block whereas AES-CBC has a 128 bit block. As a result the use of AES-

CBC with small Clear Text Data results in large encrypted Data with embedded padding. In other words, the alignment for one packet is always MAX(CIPHER_BLOCK_SIZE, ALIGN).

SPI_SIZE:
ESP Security Policy Index is 4 byte long to identify the SAD-entry for incoming traffic.
Diet-ESP omits, leaves unchanged, or reduces the SPI sent on the wire to the 0, 1, 2, 3 or 4 LSB.
Compression only impacts the data sent on the wire and therefore OS SHOULD only deal with 4 byte decompressed SPIs in the SAD. This allows systems to send and receive multiple SPI_SIZE with different hosts. Decompressing the SPI at the receiver may involve IP addresses (see Appendix B.2.1).
Compressing the SPI has significant security impacts as detailed in Section 9. It should be guided by 1) the number of simultaneous inbound SA the device is expected to handle and 2) reliability of the IP addresses in order to identify the proper SA for incoming packets. More specifically, a sensor with a single connection to a Security Gateway, may bind incoming packets to the proper SA based only in its IP addresses. In that case, the SPI may not be necessary. Other scenarios may consider using the SPI to index the SAs or may consider having multiple ESP channels with the same host from a single host. In that case it may choose a reduced length for the SPI. Note also that the value 0 for the SPI is not allowed to be sent on the wire as described in [RFC4303].

SN_SIZE:
ESP Sequence Number is 32 bit and extended SN is 64 bit long and used for anti-replay protection.
Diet-ESP omits, leaves unchanged or reduces SN sent on the wire to 0, 1, 2, 3 or 4 LSB.
Decompressing the SN at the receiver is guided by a linear extrapolation of the expected received Sequence Number and the LSB-SN sent on the wire. To avoid packet overhead, this configuration is stored within the SA, whereas it remains valid during its lifetime. Therefore an implementer should consider the LSB window such that two consecutive received SN should not present a difference of more than the LSB window.
In some cases, the received SN may increase by a high number e.g. using the time as the SN or because of a high number of packet loss. See Section 9.3 for the related security considerations for this case.

Note that SN and SPI MUST be aligned to a multiple of the Alignment value (ALIGN).

NH:

Next Header in ESP is used to identify the first header inside the ESP payload.

Diet-ESP is able to remove the Next Header field from the ESP-Trailer.

Removing the Next Header is possible only if the underlying protocol can be derived from the Traffic Selector (TS) within the Security Association (SA). More specifically, the Next Header indicates whether the encrypted ESP payload is an IP packet, a UDP packet, a TCP packet or no next header. The NH can only be removed if this has been explicitly specified in the SA or if the device has a single application.

Note that removing the Next Header impacts how encryption is performed. For example, the use of AES-CBC [RFC3602] mode requires the last block to be padded, reaching a 128 bit alignment. In this case removing the Next Header increases the padding by the Next Header length, which is 8 bits. In this case, removing the Next Header provides few advantages, as it does not reduce the ESP packet length. With AES-CBC, the only advantage of removing the Next Header would be for data with the last block of 15 bytes. In that case, ESP pads with 15 modulo 16 bytes, sets the 1 byte pad length field to 15 and add the one byte Next Header field. This leads to 15 + 15 + 1 + 1 = 32 bytes to be sent. On the other hand, removing the Next Header would require only the concatenation of the pad length byte with a 0 value, which leads to 16 bytes to be sent.

Other modes like AES-CTR [RFC3686] do not have block alignment requirements, so the only alignment constraint comes from the device hardware alignment (ALIGN). Suppose A designates the alignment constraint from OS, hardware, encryption, packet format...). A is fixed and consider then any data of length k * A + A - 1 bytes with k an integer. Sending this data using ESP takes advantage of removing the Next Header as it reduces the number of bytes to be sent by A over the traditional ESP. As a result, for 8 bit alignment hardware (A = 1) removing the Next Header always prevent an unnecessary byte to be sent.

PAD:

With ESP, all packets have a Pad Length field. This field is usually present because ESP requires IP alignment which is ensured with padding.

Diet-ESP considers removing the Padding and the Pad Length field. If PAD is present, then it is computed according to ALIGN.

In fact, some devices might use an 8 bits alignment, in which case padding is not necessary. Similarly, sensors may send application data of fixed length matching the alignment. Note that alignment may be required by the device (8-bit, 16-bit, or more generally 32-bit), but it may also be required by the encryption block size

(AES-CBC uses 128 bit blocks). With ESP these scenarios would result in an unnecessary Pad Length field always set to zero. Diet-ESP considers those case with no padding, and thus the Pad Length field can be omitted.

Diet-ESP_ICV_SIZE:
Integrity Check Value (ICV) is used to authenticate the Diet-ESP Payload.
Diet-ESP considers sending the whole ICV or the first 1 byte resp (2, 4, 8, 12, 16, 32) bytes.
ESP negotiates authentication protocols for every SA. These protocols generate an ICV of a length defined by the authentication protocol. These authentication protocols do not provide ways to perform weak authentication, as there is no way to reduce the size of the ICV. IoT is interested in weak authentication as it may send a small amount of bytes, and the trade-of between battery life time and security may be worth. As a result Diet-ESP indicates the number of bytes of the ICV. Note that reducing the size of the ICV may expose the system to security flows. See Section 9 for more details. Note that ICV is optional so if one chooses not to perform authentication, he MUST negotiate the authentication algorithm to NULL as defined in [RFC4835].
Note also that the Diet-ESP ICV value differs from the Standard ESP value since the authenticated data is not the same. In the case of the Diet-ESP ICV, the ICV is computed over the compressed Diet-ESP payload, whereas in the case of the Standard ESP ICV the ICV is computed over the uncompressed packet. This means that the decompress Diet-ESP ICV is not expected to match the Standard ESP ICV value.

Some additional parameters may be added to the Diet-ESP Context. Such parameters are defined in other documents, like [draft-mglt-ipsecme-diet-esp-iv] to compress the Initialization Vector required by cipher algorithms or [draft-mglt-diet-esp-inner-compression] the compression of protocol headers inside the encrypted ESP payload.

## 4.2. Standard ESP compliant Diet-ESP Context

Table 2 defines the Diet-ESP Context that produces regular ESP packets. This makes Diet-ESP compatible with standard ESP.

| Context Field Name | Default Value |
|---|---|
| ALIGN | IP alignment (4 bytes for IPv4 and 8 bytes for IPv6) |
| SPI_SIZE | 4 bytes |
| SN_SIZE | 4 bytes |
| NH | Present |
| PAD | Present |
| Diet-ESP_ICV_SIZE | Not compressed |

Table 2: Diet-ESP Default Context for regular ESP

- ALIGN: IP alignment
  IP alignment is 32 bit for IPv4 and 64 bit for IPv6.

- SPI_SIZE: 4 bytes
  This is not problematic for devices that handle a few simultaneous connections.

- SN_SIZE: 4 bytes
  This causes a window of 2**32 lost packets for a regular increment of 1 for each packet.

- NH: Present
  Next Header remains uncompressed as compression cannot be performed in all scenarios.

- PAD: Present
  Padding is computed according to the IP alignment.

- Diet-ESP_ICV_SIZE: Not compressed
  The Diet-ESP ICV is computed. The length is determined by the authenticating algorithm negotiated by the SA. This value is not truncated.

## 4.3. Default Diet-ESP Context

This section defines a default Diet-ESP Context. IPsec/ESP has been designed to provide a secure framework that remains secure in any scenarios. As a result, specific scenarios may carry unnecessary bytes. In fact, compression is performed on a per-scenario basis, the Diet-ESP Context configuration for a given scenario may introduce vulnerabilities in another scenario. As a result, Diet-ESP can hardly define a optimized Diet-ESP Context that matches with any scenarios.

On the other hand, Diet-ESP is very flexible and make possible to compress any fields. Defining which field can be compressed without introducing vulnerabilities requires specific security knowledge we do not expect all application developers to have. Instead, we would like application developers to be able to simply mention that a given application needs to be secured with diet-esp without specifying any parameters. In that case, a Default Diet-ESP Context will be considered. This Diet-ESP Context is probably not optimized for the given scenario, but at least it does not introduce vulnerabilities. The values for the Default Diet-ESP Context are specified in Table 3.

| Context Field Name | Default Value |
|---|---|
| ALIGN | 8 bit Alignment |
| SPI_SIZE | 2 bytes |
| SN_SIZE | 2 bytes |
| NH | Present |
| PAD | Removed |
| ESP_ICV_SIZE | Not compressed |

Table 3: Diet-ESP Default Context

- ALIGN: 8 bit
  As most IoT devices CPUs are likely to deal with 8 bit alignment.

- SPI_SIZE: 2 bytes
  reduced to 2 bytes. This is not problematic for devices that handle a few simultaneous connections.

- SN_SIZE: 2 bytes
  reduced to 2 bytes which causes a window of 512 lost packets for a regular increment of 1 for each packet.

- NH: Present
  Next Header remains present in the packet as compression cannot be performed in all scenarios.

- PAD: Removed
  Padding is removed as AES-CTR (CCM*) is widely deployed for IoT, and most IoT devices can deal with 8 bit alignment. If AES-CBC is used, the padding is performed by the encryption mode itself.

- Diet-ESP_ICV_SIZE: Not compressed

The Diet-ESP ICV is computed. The length is determined by the authenticating algorithm negotiated by the SA. This value is not truncated in order to remain the security provided by the algorithm

## 5. Diet-ESP Protocol Description

This section defines Diet-ESP on the top of the ROHC and ROHCoverIPsec framework. Section 5.1 presents and explains the choice of these frameworks. This section is informational and its only goal is to position our work toward ROHC and ROHCoverIPsec. Section 5.2 defines profiles for all fields expect the Diet-ESP ICV field. Section 5.4 describes the Diet-ESP ICV field. In fact the Diet-ESP ICV field is not derived from the Standard ESP ICV field, but instead is built especially by Diet-ESP.

### 5.1. Robust Header Compression (ROHC)

ROHC enables the compression of different protocols of all layers. It is designed as a framework, where protocol compression is defined as profile. Each profile is defined for a specific layer, and ROHC compression in [RFC3095] defines profiles for the following protocols: uncompressed, UDP/IP, ESP/IP and RTP/UDP/IP. The compression occurs between the IP and the MAC layer, and so remains independent of an eventual IP alignment.

The general idea of ROHC is to classify the different protocol fields. According to the classification, they can either completely and always be omitted, omitted only after the fields has been sent once and registered by the receiver or partly sent and be regenerated by the receiver. For example, a static field value may be negotiated out of band (for example IP version) and thus not be sent at all. In some cases, the value is not negotiated out of band and is carried in the first packet (for example SPI, UDP ports). As a result, the first packet is usually not so highly compressed with ROHC. Finally, some variable fields (for example Sequence Number) can be represented partially by their Last Significant Bits (LSB) and regenerated by the receiver.

The main issue encountered with ESP and ROHC is that ESP may contain encrypted data which makes compression between the IP and MAC layer complex to achieve. Therefore ROHC defines different compression of the ESP protocol (see Figure 2), so compression of the Clear Text Data can occur before the ESP encryption. Regular ROHC can compress the ESP header. If the packet is not encrypted, the rate of compression is extremely high as the whole packet including padding can be compressed in the regular ROHC stack, too. For encrypted payload ROHC defines ROHCoverIPsec ([RFC5856], [RFC5857], [RFC5858])

to compress the ESP payload before it is going to be encrypted. This leads to a second ESP stack, where another ROHC compressor (resp. decompressor) works (see Figure 2). Excluding the first packet which initializes the ROHC context, this makes ESP compression highly efficient.

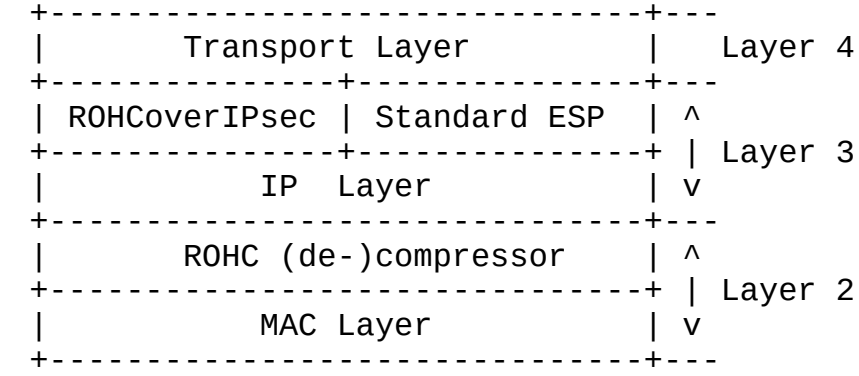


Figure 2: The two different ROHC layers in the TCP/IP stack.

The first drawback for ROHC and ROHCoverIPsec is, that it leads to two ROHC compression layers (ROHCoverIPsec before ESP encryption and ROHC before the MAC layer) in addition to two ESP implementations (Standard ESP and ROHCoverIPsec ESP). Both frameworks are quite complex and require a lot of resources which does not fit IoT requirements. Then ROHCoverIPsec also limits the compression of the ESP protocol, according to the IP restrictions. Padding remains necessary as IPsec is part of the IP stack which requires a 32 bits (resp. 64 bits for IPv6) aligned packet. This makes compression quite inefficient when small amount of data are sent.

Note that mechanism to compress encrypted fields may be possible with ROHC only and without ROHCoverIPsec. Such mechanisms may be possible for fields like the Next Header or the Padding and Pad Length when the data sent is of fixed size. As the sizes of the fields are known the compressor may simply remove these fields. However, even in this case, it almost doubles the amount of computation on the receiver's side. In fact, the ROHC compressor would almost decompress and re-encrypt the compressed ESP payload before forwarding it to the IP stack. In addition, since the receiver has to re-encrypt the decompressed information before integrity of the packet can be checked, one can easily construct a DoS attack. Flooding the receiver with invalid packet causes the receiver to perform the complex encryption and authentication algorithm for each packet.

## 5.2. Diet-ESP ROHC framework

This section defines how the compression of all ESP fields is performed within the ROHC and ROHCoverIPsec frameworks. More especially fields that are in the ESP Header (i.e. the SPI and the SN) and the ICV are compressed by the ROHC framework. The other fields, that is to say those of the ESP Trailer, are compressed by the ROHCoverIPsec framework. The specific Diet-ESP ICV field is detailed in Section 5.4 as more details are required.

Diet-ESP fits in the ROHC and the ROHCoverIPsec in a very specific way.

1 - Diet-ESP does not needs any ROHC signaling between the peers. More specifically, ROHC Initialization and Refresh (IR), or ROHC IR-DYN or ROHC Feed back packet are not considered with Diet-ESP. The first reason is that fields are either STATIC or PATTERN and their value or profile is defined through the Diet-ESP Context agreement. This agreement is out of scope of ROHC, it is expected to be agreed by other protocols like IKEv2 and thus is considered as an out-of band agreement by the peers. Then, the profiles are applied for each Security Association that is unidirectional. In fact an IKEv2 negotiation results in two unidirectional SA. As a result, each SA the packets are sent in one direction only, which corresponds to the Unidirectional mode -- U-mode of ROHC.

2 - Diet-ESP only exchanges compressed data. How the compression / decompression occurs is defined by the Diet-ESP Context. Once the Diet-ESP Context has been agreed, both peers are in a Second Order (SO) State and exchange only compressed data.

3 - Diet-ESP only compresses ESP packets, it may include inner packet compression, but Diet-ESP does not make any assumption on the IP compression. This is made in order to make Diet-ESP interoperable with multiple IP compression protocols.

4 - Diet-ESP compresses partially STATIC fields as they are used as indexes by the receiver, and may not completely be removed.

## 5.3. Diet-ESP header classification

The ROHC header field classifications are defined in Appendix A.1 of [RFC3095] and Appendix A of [RFC5225].

| Field | ROHC class | Framework | Encoding Method | Diet-ESP Context Parameters |
|---|---|---|---|---|
| SPI | STATIC-DEF | ROHC | LSB | SPI_SIZE |
| SN | PATTERN | ROHC | LSB | SN_SIZE |
| Padding | PATTERN | ROHCoverIPsec | Removed | PAD, ALIGN |
| Pad Length | PATTERN | ROHCoverIPsec | Removed | PAD, ALIGN |
| Next Header | STATIC-DEF | ROHCoverIPsec | Removed | NH |

Table 4: Diet-ESP ROHC profile.

SPI:
The SPI indexes the SA, is negotiated by the two peers (e.g. via IKEv2 or manually) and remains the same during the session. Therefore, as defined in Appendix A.6 of [RFC5225] this field is classified as STATIC-DEF. The compressed SPI consists in the SPI_SIZE LSB of the negotiated 32 bit SPI, and SPI_SIZE is provided by the Diet-ESP Context.

SN:
The SN is used for anti-replay protection and is modified in every packet. In default cases, the ESP Sequence Number will be incremented by one for each packet sent. Therefore, as defined in Appendix A.6 of [RFC5225] this field is classified as PATTERN. The compressed SN consists in the SN_SIZE LSB of the 32 bit or 64 bit SN, and SN_SIZE is provided by the Diet-ESP Context.

Padding:
Padding is used for alignment purposes and is computed on a per-packet basis. Therefore it is classified as PATTERN. The compressed Padding is defined by PAD and ALIGN provided by the the Diet-ESP Context. If PAD is set the Padding and Pad Length fields are removed. If PAD is unset, Padding is computed according to the ALIGN and the padding length is indicated in the PAD Length field.

Pad Length:
Pad Length indicates the length of the Padding field and is computed on a per-packet basis. Therefore it is classified as PATTERN. See Padding for the compressed Pad Length.

Next Header:

The Next Header indicates the next layer in the inner ESP Payload. To be compressed the Next Header MUST remain the same during the session. This means that it MUST have been negotiated (e.g. by IKEv2) and can be derived from the Traffic Selectors. If this condition is met and the Next Header compression is requested by the peers with NH set in the Diet-ESP Context, then the Next Header field MUST be removed.

### 5.4. Diet-ESP ICV

With Standard ESP the Standard ESP ICV is computed over the whole ESP Packet. If Diet-ESP were using the Standard ESP ICV value, then Diet-ESP would have to decompressed Diet-ESP to Standard ESP packet to check the Standard ESP ICV value. First, this is not in the scope of Diet-ESP that is looking for light implementation of ESP. Then, as there is no standard way to generate the Padding, this may be impossible in most cases.

The Diet-ESP ICV payload is defined to enable an integrity check without decompressing the Diet-ESP packet to a Standard ESP packet. In order to do so, the Diet-ESP ICV is computed over the Diet-ESP packet before the ESP Header is compressed. In other words, the Diet-ESP ICV is computed over the ROHCoverIPsec compressed ESP payload before ROHC compression.

The reason of building the Diet-ESP ICV over the uncompressed header is the anti-replay protection of IPsec. If anti-replay is enabled at the receiver of the packet, the ICV ensures the integrity of the SN sent on the wire. Suppose the SN is compressed to 0 byte and the Diet-ESP ICV is built over the compressed ESP Header. In that case, the Diet-ESP ICV would not consider the SN value and thus removes the ESP anti-replay mechanism, as the SN cannot be compared with the one chosen by the sender. The problem also occurs when the SN is compressed to less then 4 bytes and the SN has been increased by more then SN_SIZE. For example, if the SN_SIZE is 1 byte the maximum increasing can be 255. If the received SN is increased by 300, the receiver will recognize a increase of only 45. Integrity check of the ICV with the whole ESP Header can determine the new SN is 300 and not 45.

Due to this issues the SN has to be decompressed to 32 bit SN before the Diet-ESP ICV generation takes place (see Figure 4). More specifically the regular ESP header is used for the Diet-ESP ICV generation on sender and receiver. This mechanism ensures the correctness of the anti-replay mechanism and the possibility to sent Standard ESP conform packets remains. As the receiver includes the Diet-ESP header to the Diet-ESP ICV generation he always checks the whole 32 Bit SN.

The algorithm used to generate the Diet-ESP ICV is the same as the one negotiated for ESP. As this field is computed for every packet, it is classified as PATTERN. The compressed Diet-ESP ICV consists in the Diet-ESP_ICV_SIZE LSB of the Diet-ESP ICV. Diet-ESP_ICV_SIZE is provided by the Diet-ESP Context. Profile parameters are summed up in Table 5

| Field | ROHC class | Framework | Encoding Method | Diet-ESP Context Parameters |
|---|---|---|---|---|
| Diet-ESP ICV | PATTERN | ROHCoverIPsec / ROHC | LSB | Diet-ESP_ICV_SIZE |

Table 5: Diet-ESP ICV profile.

The Diet-ESP ICV differs from the ROHC ICV described in section 4.2 of [RFC5858]. The ROHC ICV is computed over the Clear Text Data, and encapsulated in the ESP payload. The goal of the ROHC ICV is to check the integrity of Clear Text Data output. It ensures that the compressed payload is not corrupted. As a result, the ROHC ICV authenticates the Clear Text Data over the whole chain compressor / network / decompressor. In contrast the Diet-ESP ICV authenticates the Diet-ESP packet over the network transmission. Note that the ROHC-ICV can be disabled by negotiating the algorithm NULL in the ROHC_INTEG notify payload [RFC5857].

Figure 3 illustrates the different Standard ESP ICV, ROHC ICV and Diet-ESP ICV. Note that ROHC ICV can be used with Diet-ESP ICV or Standard ESP ICV. Diet-ESP considers ROHC-ICV as disabled, that is to say that ROHC_INTEG algorithm is set to NULL.

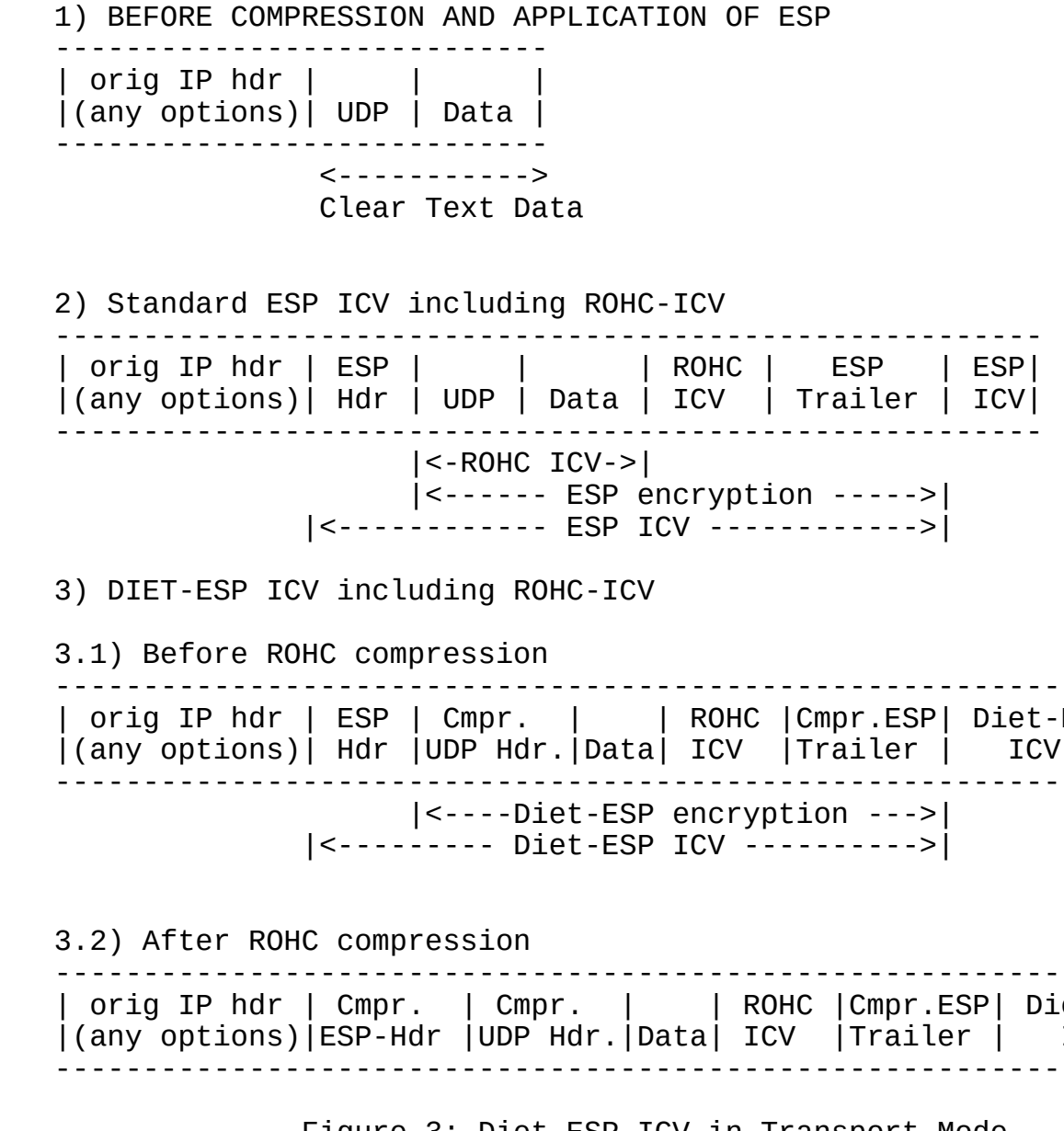


Figure 3: Diet-ESP-ICV in Transport Mode.

Upon receipt a Diet-ESP ICV, the receiver MUST compute the Diet-ESP ICV and compare with the LSB provided in the packet. If a match occurs, the Diet-ESP packet is authenticated, otherwise, the packet MUST be rejected as illustrated in figure Figure 4.

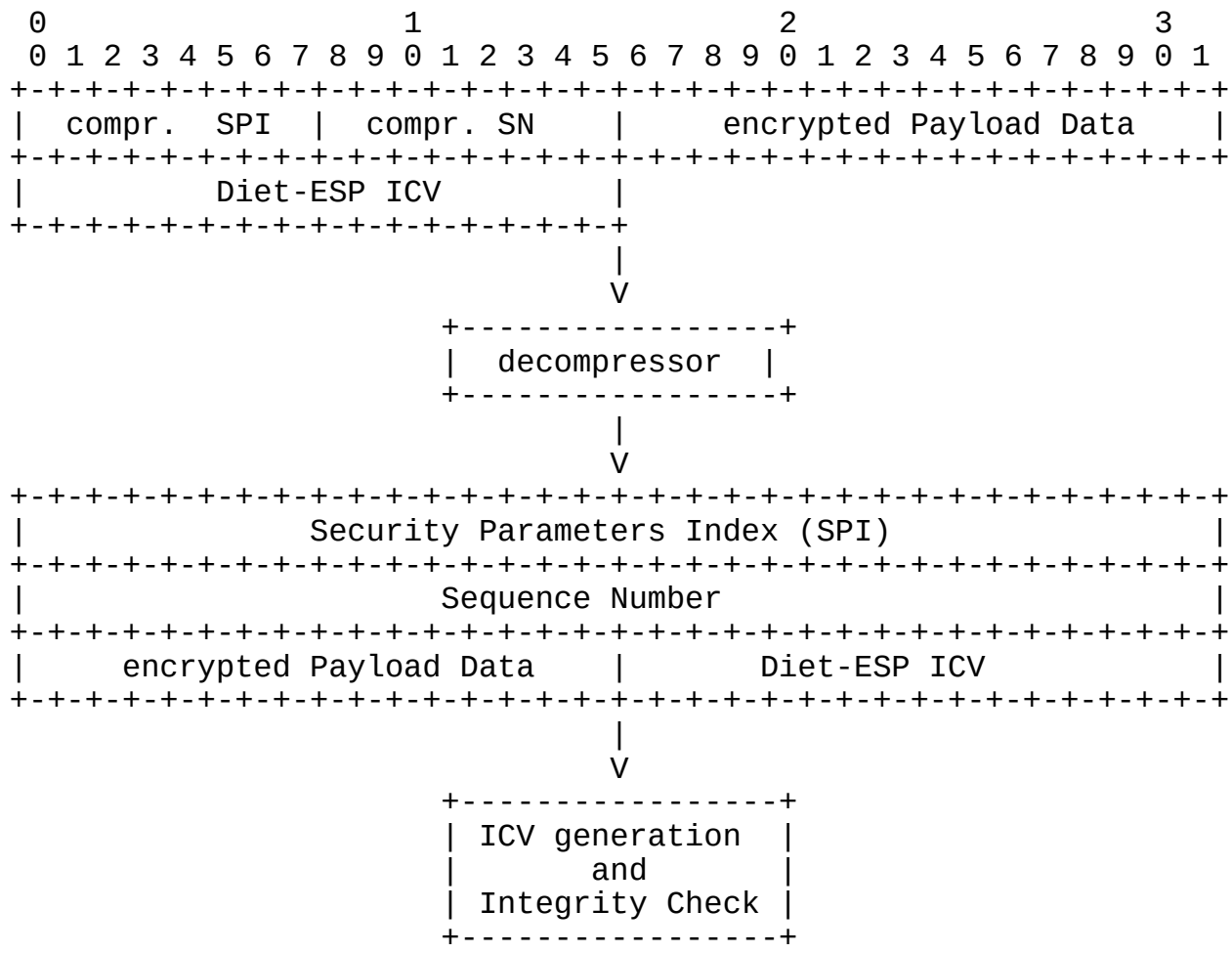


Figure 4: Example of decompression of the header, before ICV generation and checking.

## 6. Interaction with other Compression Protocols

Diet-ESP exclusively defines compression for the ESP protocol as well as the ESP payload. It does not consider compression of the IP protocol. ROHC or 6LoWPAN may be used by a sensor to compress the IP (resp. IPv6) header. Since compression usually occurs between the MAC and IP layers, there are no expected complications with this family of compression protocols.

### 6.1. 6LoWPAN

Diet-ESP smoothly interacts with 6LoWPAN. Every 6LoWPAN compression header (NHC_EH) has an NH bit. This one is set to 1 if the following header is compressed with 6LoWPAN. Similarly, the NH bit is set to 0 if the following header is not compressed with 6LowPAN. Thus, interactions between 6LowPAN and Diet-ESP considers two case: 1) NH

set to 0: 6LowPAN indicates the Diet-ESP payload is not compressed and 2) NH set to 1: 6LowPAN indicates the Diet-ESP payload is compressed.

Suppose 6LowPAN indicates the Next Header ESP is not compressed by 6LowPAN. If the peers have agreed to use Diet-ESP, then the ESP layer on each peers receives the expected Diet-ESP packet. Diet-ESP is fully compatible with 6LowPAN ESP compression disabled.

Suppose 6LowPAN indicates the Next Header ESP is compressed by 6LowPAN. ESP compression with 6LowPAN [I-D.raza-6lowpan-ipsec] considers the compression of the ESP Header, that is to say the compression of the SPI and SN fields. As a result 6LowPAN compression expects a 4 byte SPI and a 4 byte SN from the ESP layer. Similarly 6LowPAN decompression provides a 4 byte SPI and a 4 byte SN to the ESP layer. If the peers have agreed to use Diet-ESP and one of them uses 6LowPAN ESP compression, then the Diet-ESP MUST use SPI SIZE and the SN SIZE MUST be set to 4 bytes.

## 6.2. ROHC

ROHC and ROHCoverIPsec have been used to describe Diet-ESP. This means the ROHC and ROHCoverIPsec concepts and terminology have been used to describe Diet-ESP. In that sens Diet-ESP is compatible with the ROHC and ROHCoverIPsec framework. The remaining of the section describes how Diet-ESP interacts with ROHC and ROHCoverIPsec profiles and payloads.

ROHC compress packets between the MAC and the IP layer. Compression can only be performed over non encrypted packets. As a results, this section considers the case of an ESP encrypted packet and an ESP non encrypted packet.

For encrypted ESP packet, ROHC profiles that enable ESP compression (e.g. profile 0x0003 and 0x1003) compresses only the ESP Header and the IP header. To enable ROHC compression a Diet-ESP packet MUST present an similar header as the ESP Header, that is a 4 byte SPI and a 4 byte SN. This is accomplished by setting SPI_SIZE = 4 and SN_SIZE = 4 in the Diet-ESP Context. Reversely, if the Diet-ESP packet presents a 4 byte SPI and a 4 byte SN, ROHC can proceed to the compression. Note that Diet-ESP does not consider the IP header, then ESP and Diet-ESP are encrypted, thus ROHC can hardly make the difference between Diet-ESP and ESP packets. For encrypted packets, the only difference at the MAC layer might be the alignment.

For non encrypted ESP packet, ROHC MAY proceed to the compression of different fields of ESP and other layers, as the payload appears in clear. ROHC compressor are unlikely to deal with ESP fields

compressed by Diet-ESP. As a result, it is recommended not to combine Diet-ESP and ROHC ESP compression with non encrypted ESP packets.

### 6.3. ROHCoverIPsec and 6LoWPANoverIPsec

ROHC or 6LoWPAN are not able to compress the ESP payload, as long as it is encrypted. Diet-ESP describes how to compress the ESP-Trailer, which is part of the encrypted payload can be compressed. 6LowPANoverIPsec (section 2 of [I-D.raza-6lowpan-ipsec]) and ROHCoverIPsec define the compression of the ESP payload, more specifically the upper layer headers (e.g. IP header or Transport layer header). These protocols need a second, modified ESP stack in order to make the payload compression possible. Then the packets with compressed payload are forwarded to this second ESP stack which can compress or decompress the payload.

Diet-ESP and its extensions also needs a modified ESP stack in order to perform the compression of ESP payload possible. In addition, fields that are subject to compression are most likely to be the same with Diet6ESP and 6LowPANoverIPsec and/or ROHCoverIPsec. Therefore, if a device implements Diet-ESP and 6LowPANoverIPsec and/or ROHCoverIPsec the developer needs to define an order the various frameworks perform the compression. Currently this order has not been defined, and Diet-ESP is unlikely to be compatible with 6LowPANoverIPsec and/or ROHCoverIPsec. Integration of Diet-ESP and 6LowPANoverIPsec and/or ROHCoverIPsec has not been considered in the current document as Diet-ESP has been designed to avoid implementations of 6LowPANoverIPsec and/or ROHCoverIPsec frameworks to be implemented into the devices. Diet-ESP has been designed to be more lightweight than 6LowPANoverIPsec and/or ROHCoverIPsec by avoiding negotiations between compressors and decompressors.

## 7. Diet-ESP and Requirements

[draft-mglt-ipsecme-diet-esp-requirements] lists the requirements for Diet-ESP. This section position Diet-ESP described in this document toward these requirements.

R1: Diet-ESP is able to handle alignments of 8, 16, 32 and 64 bits.

R2: is not in the scope of Diet-ESP. Announcement of the Byte-Alignment should be performed by IKEv2.

R3: Diet-ESP does not modify how encryption occurs. It only changes the encrypted payload, which is one of the parameters for the encryption function. Therefore Diet-ESP is able to

work with any encryption defined in [RFC4835] which also includes AES-CCM [RFC4309].

Combined Mode algorithm (e.g. AES-CCM, AES-GCM) have an additional parameter, called Addition Authentication Data (AAD). This AAD requires the uncompressed ESP header that is to say the full SPI and SN. These parameters are not removed by Diet-ESP. There are well known by the two peer. The ESP Header MUST be uncompressed before proceeding to encryption/ decryption.

R4: Diet-ESP can remove all static and compress fields from the protocol.

R5: The inner payload compression mechanisms are not defined in this document. This aspect is the purpose of [draft-mglt-inner-compression]

R6: Diet-ESP compressed packet fields are always a number of bytes -- that is Diet-ESP do not result in compressed fields that are not expressed in a natural number of bytes.

R7: Diet-ESP allows the developer define the maximum compression within the Diet-ESP context. The way the agreement is done, is out of scope of this document and is described in [draft-mglt-diet-esp-ikev2].

R8: Each field in the packet can be compressed separately, which provides high flexibility.

R9: Since Diet-ESP does not propose compression method flexibility. The proposed methods are generic enough and there is not advantage for this flexibility and so it does not seems appropriated for Diet-ESP.

R10: Each Diet-ESP client can have his own set of supported contexts. The negotiation is out of scope of this document and described in [draft-mglt-diet-esp-ikev2].

R11: Diet-ESP adds small complexity to Standard ESP, like described in Appendix B. In- and Outbound packet procession is straight-forward, like shown in Appendix B.5 and Appendix B.5. Appendix A provides a implementation guideline for a minimal use case. This one can be ported to any other use case.

R12: Diet-ESP is easy to configure and provides a default-context if a developer does not want to dive into the details of Diet-ESP.

R13: Diet-ESP can interact with 6LoWPAN and ROHC IP compression, but SHOULD be able to interact with all future compression applying after the IP layer as well.

R14: Compatibility with Standard ESP 1: Diet-ESP can be implemented instead of, nearby or like an add-on to an existing Standard ESP implementation.

R15: Compatibility with Standard ESP 2: Diet-ESP is able to work without compression and works with 32 and 64 bits alignment, which makes it compatible with Standard ESP.

## 8. IANA Considerations

There are no IANA consideration for this document.

## 9. Security Considerations

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

### 9.1. Size of the SPI

Small SPI_SIZE exposes the device to DoS. For a device, the number of SA is related to the number of SPI. For systems using small SPI_SIZE values as index of their database, the number of simultaneous communications is limited by the SPI_SIZE. This means that a given device initiating SPI_SIZE communications can isolate the system. In order to leverage this vulnerability, one can consider receiving systems that generate 32 bits SPI with a hash function that considers different parameters associated to the reduced SPI. For example, if one use the IP addresses as well as the reduced SPI, the number of SPI becomes SPI_SIZE per IP address. This may be sufficient as sensors are not likely to perform multiple communications.

### 9.2. Size of the Diet-ESP ICV

Small size of ICV reduces the authentication strength. For example 8 bits mean that authentication can be spoofed with a probability of 1/256. Standard value considers a length of 96 bit for reliable authentication. If specified, the ICV field is truncated after the given number of bits which, for sure, has to be mentioned while incoming packet procession as well. For removing authentication ESP NULL has to be negotiated, as described in [RFC4303].

### 9.3. Size of the SN

This section describes the security consideration for two possible scenarios

Increasing by 1
: If the SN is increased by 1 for each packet, the last sent/ received SN is stored at the sender/receiver. In this case the device MAY negotiate a SN SIZE of 0 if receiving unordered packets or packet loss can be secured.

Using the Time
: A minimal ESP implementation MAY choose to use a always increasing, already existing and stored value in order to save the storage used by the SN in the SA. For this purpose it could use e.g. the current time, when the packet is send. For Diet-ESP this leads to problems, as the receiver is not able to know the current SN used in the packet. If one decides to use this mechanism, he has to deal with the following restrictions:

1) the SN SIZE MUST NOT be 0

2) If the SN SIZE is NOT 4:

    2.1) The SN SIZE MUST be chosen so that the time between two packet is less than 2 ** SN SIZE.

    2.2) The start value of the SN MUST be ensured to be the same, between the two peers. This may be done with time synchronization. Suppose a 1 byte SN is sent on the wire the sender decides to use the time to prevent the storage of the SN in the SA. Now the first packet is sent after a couple of seconds and only the 1LSB of the time is sent in the packet. If the two peers have a time difference of more than 255(let’s say seconds) the 3 Most Significant Bytes of the SN may be wrong and the ICV will not match.

Due to this restrictions, we highly RECOMMEND not to use the time as a sequence number, if the SN SIZE is NOT 4.

## 10. Acknowledgment

The current draft represents the work of Tobias Guggemos while his internship at Orange [GUGG14].

Diet-ESP is a joint work between Orange and Ludwig-Maximilians-Universitaet Munich. We thank Daniel Palomares and Carsten Bormann for their useful remarks, comments and guidance.

## 11. References

### 11.1. Normative References

[GUGG14] Guggemos, TG., "Diet-ESP: Applying IP-Layer Security in Constrained Environments (Masterthesis)", September 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3095] Bormann, C., Burmeister, C., Degermark, M., Fukushima, H., Hannu, H., Jonsson, L-E., Hakenberg, R., Koren, T., Le, K., Liu, Z., Martensson, A., Miyazaki, A., Svanbro, K., Wiebke, T., Yoshimura, T., and H. Zheng, "RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed", RFC 3095, July 2001.

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, September 2003.

[RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, January 2004.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005.

[RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006.

[RFC4835] Manral, V., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4835, April 2007.

[RFC5225] Pelletier, G. and K. Sandlund, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite", RFC 5225, April 2008.

[RFC5857] Ertekin, E., Christou, C., Jasani, R., Kivinen, T., and C. Bormann, "IKEv2 Extensions to Support Robust Header Compression over IPsec", RFC 5857, May 2010.

[RFC5858] Ertekin, E., Christou, C., and C. Bormann, "IPsec Extensions to Support Robust Header Compression over IPsec", RFC 5858, May 2010.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

## 11.2. Informational References

[I-D.raza-6lowpan-ipsec]
Raza, S., Duquennoy, S., and G. Selander, "Compression of IPsec AH and ESP Headers for Constrained Environments", draft-raza-6lowpan-ipsec-01 (work in progress), September 2013.

[RFC5856] Ertekin, E., Jasani, R., Christou, C., and C. Bormann, "Integration of Robust Header Compression over IPsec Security Associations", RFC 5856, May 2010.

# Appendix A. Example of light Diet-ESP implementation for sensor

Diet-ESP has been designed to enable light implementation. This section illustrates the case of a sensor sending a specific amount of data periodically. This section is not normative and has only an illustrative purpose. In this scenario the sensor measures a temperature every minute and sends its value to a gateway, which is assumed to collect the data. The data is sent in an UDP packet and there is no other connection between the two peers. The communication between the sensor and the gateway should be secured by a Diet-ESP connection in transport mode. Therefore the following context is chosen:

ALIGN: 8 bit
Sensors are not expected to be 32 or 64 bit CPU, and micro-controllers are expected to support 8 bit alignment.

SPI_SIZE: 0
As it is a single connection, the SA can be identified by using the IP addresses. As a result the SPI is not needed.

SN_SIZE: 0
Because only one packet every minute is sent, the packets will arrive at the receiver in an ordered way. The receiver can rebuild the SN which should be present in the packet, assuming the SN is incremented by one for each packet. Note that setting SN to 0 does not mean there is no anti replay protection. In fact, the SN is needed for the computation of the Diet-ESP ICV.

NH: Remove Next Header
Since the protocol is always UDP, the Next header can be omitted.

PAD: Remove Padding
With 8 bit alignment Padding has always a Pad Length of 0.
Setting PAD to "Remove Padding" removes the Pad Length field.

Diet-ESP_ICV_SIZE: 4 bytes
The ICV is chosen to be 32 bits in order to find a fair trade-off between security and energy costs.

Encapsulating the outgoing Diet-ESP packet is proceeded as follows:

1) SAD lookup for outgoing traffic

2) Compress ESP payload incl. Transport Header (UDP)

3) Encrypt IP payload

4) Build ESP header

5) Calculate Diet-ESP ICV

6) Compress ESP header

7) Add ${Diet-ESP_ICV_SIZE} LSB of ICV to the packet.

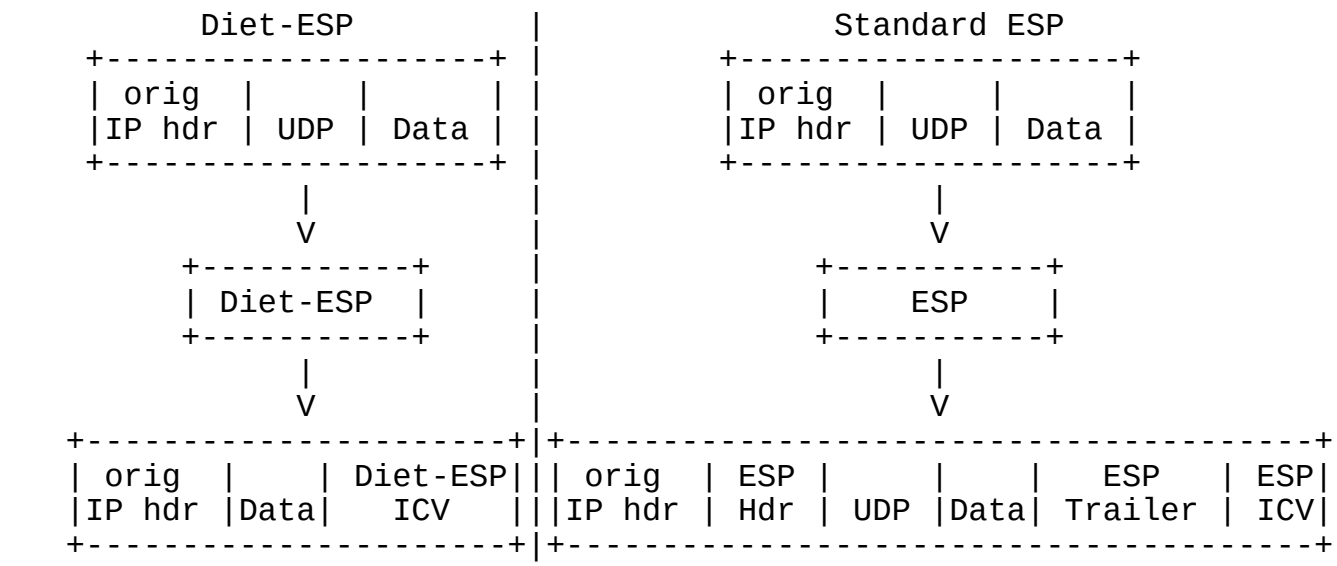


Figure 5: Minimal Example - Input and Output of the Diet-ESP function vs. Standard ESP.

Incoming Diet-ESP packet is processed as follows:

1) SAD lookup for incoming traffic traffic

2) Decompress ESP-header incl. Transport Header (UDP)

3) Calculate packet Diet-ESP ICV

4) Check integrity with ${Diet-ESP_ICV_SIZE} LSB of Diet-ESP ICV

5) Check anti-replay

6) Decrypt IP payload (excluding ICV)

7) Decompress ESP payload

## Appendix B. Difference between Diet-ESP and ESP

This section details how to use Diet-ESP to send and receive messages. The use of Diet-ESP is based on the IPsec architecture [RFC4301] and ESP [RFC4303]. We suppose the reader to be familiar with these documents and we list here possible adaptations that may be involved by Diet-ESP.

### B.1. Packet Alignment

Each ESP packet has a fixed alignment to 32 bits (resp. 64 bits in IPv6). For Diet-ESP each device has an internal parameter that defines the minimal acceptable alignment. ALIGN SHOULD be a the maximum of the peer's minimal alignment.

Diet-ESP Context with SPI_SIZE + SN_SIZE that is not a multiple of ALIGN MUST be rejected.

### B.2. SAD

#### B.2.1. Inbound Security Association Lookup

For devices that are configured with a single SPI_SIZE value can process inbound packet as defined in [RFC4301]. As such, no modifications is required by Diet-ESP.

Detecting Inbound Security Association: Identifying the SA for incoming packets is a one of the main reasons the SPI is send in each packet on the wire. For regular ESP (and AH) packets, the Security Association is detected as follows:

1. Search the SAD for a match on {SPI, destination address, source address}. If an SAD entry matches, then process the inbound ESP packet with that matching SAD entry. Otherwise, proceed to step 2.

2. Search the SAD for a match on {SPI, destination address}. If the SAD entry matches, then process the inbound ESP packet with that matching SAD entry. Otherwise, proceed to step 3.

3. Search the SAD for a match on only {SPI} if the receiver has chosen to maintain a single SPI space for AH and ESP, or on {SPI, protocol} otherwise. If an SAD entry matches, then process the inbound ESP packet with that matching SAD entry. Otherwise, discard the packet and log an audible event.

For device that are dealing with different SPI_SIZE SPI, the way inbound packets are handled differs from the [RFC4301]. In fact, when a inbound packet is received, the peer does not know the SPI_SIZE. As a result, it does not know the SPI that applies to the incoming packet. The different values could be the 0 (resp. 1, 2, 3 and 4) first bytes of the IP payload.

Since the size of the SPI is not known for incoming packets, the detection of inbound SAs has to be redefined in a Diet-ESP environment. In order to ensure a detection of a SA the above

described regular detection have to be done for each supported SPI size (in most cases 5 times). In most common cases this will return a unique Security Association.

If there is more than one SA matching the lookup, the authentication MUST be performed for all found SAs to detect the SA with the correct key. In case there is no match, the packet MUST be dropped. Of course this can lead into DoS vulnerability as an attacker recognizes an overlap of one or more IP-SPI combinations. Therefore it is highly recommended to avoid different values of the SPI_SIZE for one tuple of Source and Destination IP address. Furthermore this recommendation becomes mandatory if NULL authentication is supported. This is easy to implement as long as the sensors are not mobile and do not change their IP address.

The following optimizations MAY be considered for sensor that are not likely to perform mobility or multihoming features provided by MOBIKE [RFC4555] or any change of IP address during the lifetime of the SA.

Optimization 1 - SPI_SIZE is mentioned inside the SPI:
The SPI_SIZE is defined as part of the SPI sent in each packet. Therefore the receiver has to choose the most significant 2 bits of the SPI in the following way in order to recognize the right size for incoming Diet-ESP packets:

00: SPI_SIZE of 1 byte is used.

01: SPI_SIZE of 2 byte is used.

10: SPI_SIZE of 3 byte is used.

11: SPI_SIZE of 4 byte is used.

If the the value 0 is chosen for the SPI_SIZE this option is not feasible.

Optimization 2 - IP address based lookup:
IP address based search is one optimization one may choose to avoid several SAD lookups. It is based on the IP address and the stored SPI_SIZE, which MUST be the same value for each SA of one IP address. Otherwise it can’t neither be ensured that an SA is found nor that the correct one is found. Note that in case of mobile IP the SPI_SIZE MUST be updated for all SAs related to the new IP address which may cause renegotiation. Figure 6 shows this lookup described below.

1. Search most significant SA as follows:

1.1 Search the first SA for a match on {destination address, source address}. If an SA entry matches, then process to step 2. Otherwise, proceed to step 1.2.

1.2 Search the first SA for a match on {source address}. If an SA entry matches, then process to step 2. Otherwise, drop the packet.

2. Identify the size of the compressed SPI for the found SA, stored in the Diet-ESP context. Note that all SAs to one IP address MUST have the same value for the SPI_SIZE. Then go to step 3.

3. If the SPI_SIZE is NOT zero, read the SPI_SIZE SPI from the packet and perform a regular SAD lookup as described in [RFC4301]. If the SPI_SIZE is zero, the SA from step 1 is unique and can be used.

Note that some implementations may collect all SPI matching the IP addresses in step 2 to avoid an additional lookup over the whole SAD. This is implementation dependent.

If the sensor is likely to change its IP address, the outcome may be a given IP address associated to different SPI_SIZE. This case may occur if one IP address has been used by a device not anymore online, but the SA has not been removed. The IP has then been provided to another device. In this case the Diet-ESP Context SHOULD NOT be accepted by the Security Gateway when the new Diet-ESP Context is provided to the Security Gateway. At least the Security Gateway can check the previous peer is reachable and then delete the SA before accepting the new SA.

Another case may be that a sensor got two interfaces with different IP addresses, negotiates a different SPI_SIZE on each interface and then use MOBIKE to move the IPsec channels from one interface to the other. In this case, the Security Gateway SHOULD NOT accept the update, or force a renegotiation of the SPI_SIZE for all SAs, basically by re-keying the SAs.

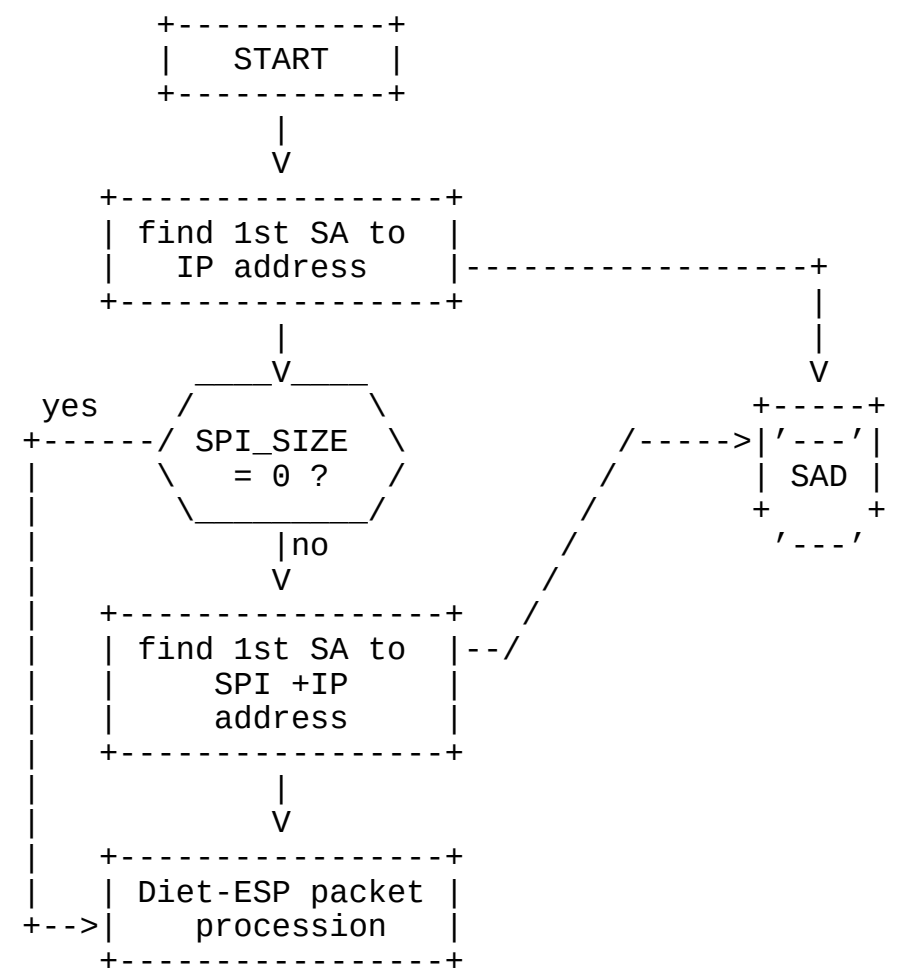


Figure 6: SAD lookup for incoming packets.

### B.2.2. Outgoing Security Association Lookup

Outgoing lookups for the SPI are performed in the same way as it is done in ESP. The Traffic Selector for the packet is searched and the right SA is read from the SA. The SPI used in the packet MUST be reduced to the value stored in SPI_SIZE.

## B.3. Sequence Number

Sequence number in ESP [RFC4303] can be of 4 bytes or 8 bytes for extended ESP. Diet-ESP introduces different sizes. One way to deal with this is to add a MAX_SN value that stores the maximum value the SN can have. Any new value of the SN will be check against this MAX_SN.

## B.4. Outgoing Packet processing

NH, TH, IH, P indicate fields or payloads that are removed from the Diet-ESP packet. How the Diet-ESP packet is generated depends on the length Payload Data LPD, BLCK the block size of the encryption algorithm and the device alignment ALIGN. We note M = MAX(BLCK, ALIGN).

- 1: Compress the headers inside the ESP payload.

- 2: if PAD and NH are set to present: Diet-ESP considers both fields Pad Length and Next Header. The Diet-ESP Payload is the encryption of the following clear text:
  Payload Data | Padding of Pad Length bytes | Pad Length field | Next Header field.
  The Pad Length value is (LPD + 2) mod [M].

- 3: if PAD is set to present and NH is set to removed: Diet-ESP considers the Pad Length field but removes the Next Header field. The ESP Payload is the encryption of the following clear text: Payload Data | Padding of Pad Length bytes | Pad Length field | Next Header field. The Pad Length value is (LPD + 1) mod [M].

- 4: if PAD is set to removed and NH is set to present: Diet-ESP considers the Next Header but do not consider the Pad Length field or the Padding Field. This is valid as long as (LPD + 1) mod [M] = 0. If M = 1 as it is the case for AES-CTR this equation is always true. On the other hand the use of specific block size requires the application to send specific length of application data.

- 5: if PAD and NH are set to removed: Diet-ESP does consider neither the Next Header field nor the Pad Length field nor the Padding Field. This is valid as long as LPD mod [M] = 0. If M = 1 as it is the case for AES-CTR this equation is always true. On the other hand the use of specific block size requires the application to send specific length of application data.

- 6: Encrypt the Diet-ESP payload.

- 7: Add ESP header.

- 8: Generate and add Diet-ESP ICV.

- 9: Compress ESP header.

## B.5. Inbound Packet processing

Decryption is for performed the other way around.

After SAD lookup, authenticating and decrypting the Diet-ESP payload the original packet is rebuild as follows:

- 1: Decompress ESP header.

- 2: Generate Diet-ESP ICV and check ICV send in the packet.

- 3: Check anti-replay

- 4: Remove compressed header.

- 5: Encrypt the Diet-ESP payload.

- 6: if PAD and NH are set to removed: Diet-ESP does consider neither the Next Header field nor the Pad Length field nor the Padding Field. The Next Header field of the IP packet is set to the protocol defined for incoming traffic within the Traffic Selector of the SA. Because there is no Padding it is disregarded.

- 7: if PAD is set to removed and NH is set to present: Diet-ESP considers the Next Header but do not consider the Pad Length field or the Padding Field. The Next Header field of the IP packet is set to the value within the Diet-ESP trailer.

- 8: if PAD is set to present and NH is set to removed: Diet-ESP considers the Pad Length field but removes the Next Header field. The Next Header field of the IP packet is set to the protocol defined for incoming traffic within the Traffic Selector of the SA. The Pad Length field is read and the Padding is removed from the Data Payload which results the original Data Payload.

- 9: if PAD and NH are set to present: Diet-ESP considers both fields Pad Length and Next Header. The Next Header field of the IP packet is set to the value within the Diet-ESP trailer. The Pad Length field is read and the Padding is removed from the Data Payload which results the original Data Payload.

- 10: Decompress the headers inside the ESP payload.

## Appendix C. Document Change Log

[draft-mglt-ipsecme-diet-esp-01.txt]:
Diet ESP described in the ROHC framework
ESP is not modified.

[draft-mglt-ipsecme-diet-esp-00.txt]:
NAT consideration added.
Comparison actualized to new Version of 6LoWPAN ESP.

[draft-mglt-dice-diet-esp-00.txt]: First version published.

## Authors' Addresses

Daniel Migault (editor)
Orange
38 rue du General Leclerc
92794 Issy-les-Moulineaux Cedex 9
France

Phone: +33 1 45 29 60 52
Email: daniel.migault@orange.com

Tobias Guggemos (editor)
Orange / LMU Munich
Am Osteroesch 9
87637 Seeg, Bavaria
Germany

Email: tobias.guggemos@gmail.com