

DICE
Internet-Draft
Intended status: Standards Track
Expires: August 4, 2014

D. Migault (Ed)
Orange
T. Guggemos
Orange / LMU Munich
D. Palomares
Orange / LIP6 - UMPC
January 31, 2014

Diet-ESP: a flexible and compressed format for IPsec/ESP
draft-mglt-dice-diet-esp-00.txt

Abstract

IPsec/ESP has been designed to secure IP packets exchanged between two nodes. IPsec implements security at the IP layer which makes security transparent to the applications, as opposed to TLS or DTLS that requires application to implement TLS/DTLS. As a result, IPsec enable to define the security rules in a similar way one establishes firewall rules.

One of the IPsec's drawbacks is that implementing security on a per packet basis adds overhead to each IP packet. Considering IoT devices, the data transmitted over an IP packet is expected to be rather small, and the cost of sending extra bytes is so high that IPsec/ESP can hardly be used for IoT as it is currently defined in RFC 4303.

This document defines Diet-ESP, a protocol that compress and reduce the ESP overhead of IPsec/ESP so that it can fit security and energy efficient IoT requirements. Diet-ESP use already existing mechanism like IKEv2 to negotiate the compression format. Furthermore a lot of information, already existing for an IPsec Security Association, are reused to offer light negotiation in addition to maximum compression.

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 August 4, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Requirements notation	2
2. Introduction	3
3. Terminology	4
4. Diet-ESP: Protocol Description	5
5. Diet-ESP Context: Format Description	8
6. Difference between Diet-ESP and ESP	11
7. IANA Considerations	14
8. Security Considerations	14
9. Acknowledgment	15
10. References	15
10.1. Normative References	15
10.2. Informational References	16
Appendix A. Comparison	16
A.1. Transmitting 1 Byte without anti-replay	16
A.2. Transmitting 1 Byte to multi directional connections.	18
Appendix B. Document Change Log	20
Authors' Addresses	21

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

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 then required" fields. This cost of additional bytes were than considered as negligible versus interoperability, and this made 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. Furthermore, IoT scenarios MAY consider that sensors MAY be designed not to interconnect between each other, but instead to be connected to a specific Security Gateway. These kind of dedicated connectivity, for example, does not impose the sensors to be fully interoperable with any other IPsec/ESP implementation. In contrast, it MAY be inter-operable with the Security Gateway and those devices supporting the same sensor's options.

In this document, we adapted ESP so IoT devices can use ESP designed for their specific needs or applications. Diet-ESP allows to reduce or remove all fields of the ESP format represented in figure 1. How the fields are reduced is defined in the Diet-ESP Context. This Diet-ESP Context MAY be announced or negotiated between the two peers. How the two devices agree on using the same Diet-ESP Context is out of scope of this document. Diet-ESP Context consist of a byte that fully defines the parameters present in a Diet-ESP packet, creating a Diet-ESP packet format agreement between compliant devices.

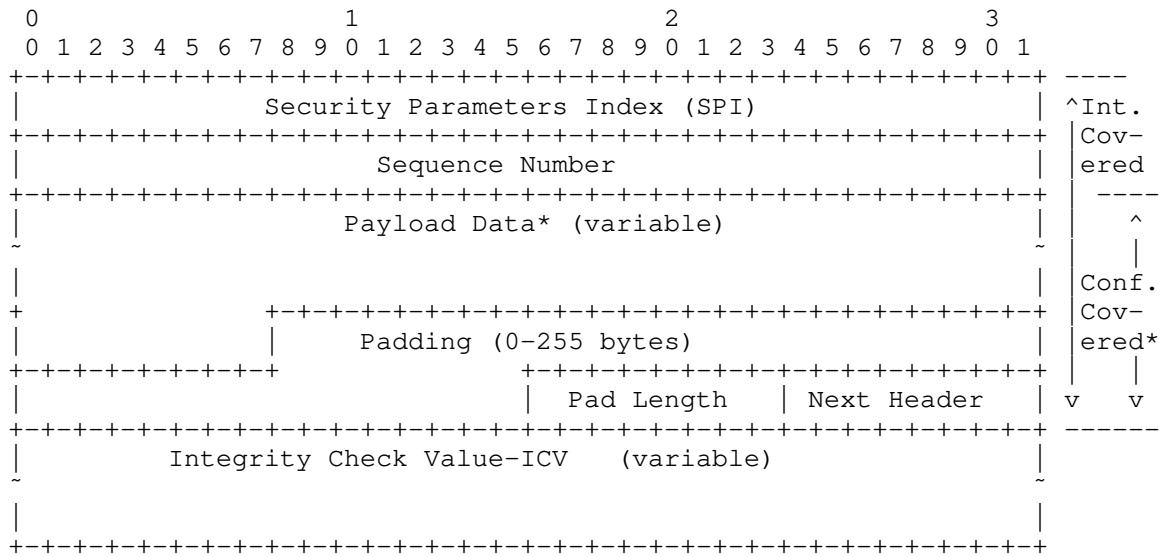


Figure 1: ESP Packet Description

3. Terminology

This document uses the following terminology:

- IoT: Internet of Things
- ESP: ESP like described in [RFC4303].
- ESP-packet: The concatenation of the following fields:
 - ESP-Header: The concatenation of the SPI and SN
 - ESP Payload: The concatenation of the following two fields. The ESP Payload is usually encrypted.
 - Data Payload: The application payload. It MAY include a transport layer header.
 - ESP-Trailer: The Padding concatenated with the Pad Length and Next Header fields.
 - ESP ICV The ICV generated throw the specified algorithm.
- Diet-ESP: Diet version of ESP like described in this document.

- Diet-ESP Context: The Context that describes the Diet-ESP packet format (see Section 5).
- Diet-ESP-packet: The concatenation of the following fields:
 - Diet-ESP-Header: The concatenation of the SPI and SN if they appear in the packet.
 - Diet-ESP Payload: The concatenation of the following two fields. The Diet-ESP Payload is usually encrypted.
 - Data Payload: The application payload. If the transport layer header is present, it MAY be removed.
 - Diet-ESP-Trailer: The Padding concatenated with the Pad Length and Next Header fields if they appear in the packet.
- Diet-ESP ICV: The ICV generated throw the specified algorithm and MAYBE truncated by Diet-ESP.

4. Diet-ESP: Protocol Description

This section describes how each field of the ESP can be compressed.

SPI SIZE: ESP Security Policy Index is 32 bits long. Diet-ESP omits, leaves unchanged, or reduces the SPI to 8, 16 or 24 bits. The length of the SPI 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 that reducing the size of the SPI may expose the system to security flows. See Section 8 for more details.

For those cases where a regular SPI of 32 bits has been negotiated (e.g. via IKEv2 [RFC5996]), the resulting SPI used for Diet-ESP packets corresponds to the high order bits of that 32 bits SPI (see Section 6 for further explanations).

SN SIZE: ESP Sequence Number is 32 bit and extended SN is 64 bit long. Diet-ESP omits, leaves unchanged or reduces SN to 8, 16, 24 bits. The length of the SN should be guided by 1) how the receiving

side handles the SN, 2) the number of packets expected to be sent over Diet-ESP channel, and 3) how the node is willing to use IKEv2 to re-key when SN are expired. SN are used to address replay attacks, thus removing SN may expose the system to security flaws. See Section 8 for more details. If SN is used, a 32 bits value may not be required. Table 1 shows the lifetime of one SA before re-keying is required in case the SN expires.

SN Length	1 packet per second	1 packet per minute	1 packet per hour
8 bit	4min 16sec	4h 16min	10 days 16h
16 bit	18h 12min 16sec	6 weeks 3 days 12h	~7 years 25 weeks
24 bit	~27 weeks 5 days	31 years 47 weeks	~1,915 years
32 bit	~136 years	~8,171 years	~490,293 years

Table 1: Lifetime of one Security Association with different sizes of Sequence Numbers compared to different use cases.

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

NH: Diet-ESP is able to remove the Next Header field from the ESP-Trailer if the underlying protocol can be derived from the Traffic Selector (TS) within the 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. Suppose a device sets an ESP channel with another peer only considering the IP addresses as TS without specifying the transport protocols or (or upper layer protocols). If the device uses this channel for multiple upper layer protocols (like HTTP and tnftp), then the NH cannot be removed as the receiver would not be able to determine whether incoming packets are HTTP or tnftp.

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 to reach 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 pad with 15 modulo 16 bytes, set the 1 byte pad length field to 15 and add the one byte Next Header field. This leads to 15 + 15 + 1 +

1 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. Using AES-CTR with ESP only requires the 32 bit alignment - mostly for OS implementation. In fact if an n byte alignment is required (for encryption or for packet format), data of length $k * n + n - 1$ bytes, k an integer, takes advantage of removing the Next Header and reduces the data to be sent over n bytes. In the case of sensor network it is very likely that data of fixed size $k * n + n - 1$ will be used. Furthermore, if 32 bits alignment is reduced to 8 bits alignment, Next Header is always an additional unnecessary byte being sent.

PAD: With ESP, all packets have a Pad Length field. This field is usually present because ESP requires a 32 bits alignment which is performed with padding. Diet-ESP considers that some devices may use 8 bits alignment, in which case padding is not necessary. Similarly, sensors may send application data that has 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.

ALIGN: Alignment for Padding and Pad Length. ESP is designed for 32 bit alignment. This is mostly an OS implementation and hardware design requirements for regular PC processors. IoT may not have these requirements. Having no alignment requirements or a 16 bits alignment requirement prevents or reduces the number of padding bytes to be sent. As a result Diet-ESP considers alternative alignment (8-bit, 16 bit, 32 bit) so to reduce the number of padding bytes.

Note that when PAD requires the Pad Length field to be present, ALIGN provides the minimum alignment padding considers. More specifically, ALIGN gives more priority to the hardware or OS implementation than to the encryption algorithm used. In fact with AES-CTR padding will be performed based on the value provided by ALIGN. However, AES-CBC padding is performed on the AES block basis (128 bits). This value overwrites the one provided by ALIGN.

IH: With ESP using the tunnel mode, the inner IP Header is sent in every ESP Payload. This extra bytes sent do not carry relevant information over sent packets. As a result Diet-ESP indicates the IP header has been omitted, and MUST be rebuilt by the receiver. These information are negotiated via IKE and are stored in the SA.

TH: With ESP the transport header is transmitted in every packet. This layer may not provide relevant information, especially for UDP transport layer. The port parameters may be negotiated via IKE and stored in the SA. As a result Diet-ESP indicates that the transport protocol header (TH) has been removed from the encrypted ESP Payload. This option can only be used if the header can be restored or if it is unnecessary for the further packet procession. Other protocols than UDP are considered out of scope of this document. TCP, for example, includes information that are not as easy to restore, like options, controls or windows. In order to use other transport layer protocols within specific configuration, additional information may be provided in the future.

ICV: ESP negotiates Authentication protocols. These protocols generate an ICV of a length defined by the authentication protocol negotiated for the SA. These authentication protocols do not provide ways to perform weak authentication, as it only reduces the size of the ICV. IoT is interested in weak authentication as it may send a small amount of bytes, and the trade-off between battery life time and security may be worth. As a result Diet-ESP indicates the number of bytes of the ICV. Diet-ESP considers sending the whole ICV or the first 1 byte resp (2, 4, 8, 12, 16, 32) bytes. Note that Note that reducing the size of the SPI may expose the system to security flows. See Section 8 for more details.

5. Diet-ESP Context: Format Description

This section describes the Diet-ESP Context that contains all necessary parameters for Diet-ESP.

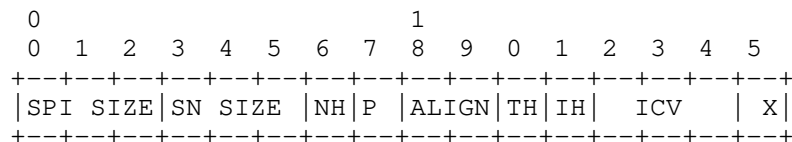


Figure 2: Diet-ESP Context

With the fields defined as below:

- SPI SIZE (3 bits): specifies the size of the SPI field length of the Diet-ESP header in byte. Values can be from 0 to 4. A zero value means the SPI does not appear in the Diet-ESP packet. The size depends on the use case, the connection should be used for.
- 000: indicates a 0 bit SPI. The SPI is removed from the packet.

- 001: indicates an 8 bit SPI in each Diet-ESP-packet.
- 010: indicates a 16 bit SPI in each Diet-ESP-packet.
- 011: indicates a 24 bit SPI in each Diet-ESP-packet.
- 100: indicates a 32 bit SPI in each Diet-ESP-packet. This configuration is according to the RFC 4303 [RFC4303]
- 101: Unassigned
- 110: Unassigned
- 111: Unassigned
- SN SIZE (3 bits): specifies the size of the Sequence Number field within the Diet-ESP header in byte. Values can be from 0 to 4. A zero value means the SN does not appear in the Diet-ESP packet. The size depends on the use case, the connection should be used for.
 - 000: indicates a 0 bit SN. The SN is removed from the packet and anti-replay is disabled on the receiver.
 - 001: indicates an 8 bit SN in each Diet-ESP-packet.
 - 010: indicates a 16 bit SN in each Diet-ESP-packet.
 - 011: indicates a 24 bit SN in each Diet-ESP-packet.
 - 100: indicates a 32 bit SN in each Diet-ESP-packet. This configuration is according to the RFC 4303 [RFC4303]
 - 101: Unassigned
 - 110: Unassigned
 - 111: Unassigned
- NH (1 bit): specifies if the Next Header field appears in the Diet-ESP trailer. NH unset to 0 indicates the Next Header field is present and NH set to 1 indicates the Next Header is omitted.
- P (1 bit): specifies if the Pad Length field appears in the Diet-ESP trailer. P unset to 0 indicates the Pad Length field is present and P set to 1 indicates the Pad Length is omitted.

- ALIGN (2 bits): specifies Padding, Padding Length as follows:
 - 00: indicates an 8 bit alignment. The field Pad Length is omitted and the Diet-ESP packet never has Padding.
 - 01: indicates a 16 bit alignment. The field Pad Length is always present.
 - 10: indicates a 32 bit alignment. The field Pad Length is always present.
 - 11: Unassigned
- TH (1 bit): specifies if the transport layer field appears in the Diet-ESP Payload Data. TH unset to 0 indicates the Transport header field is present and TH set to 1 indicates the transport header is omitted. In this case, the transport protocol MUST be specified in the SA with its associated port. If a non unique port or a non unique transport protocol is specified, this bit MUST be unset to 0. Otherwise, the device will not be able to rebuilt the transport header. This document only considers UDP.
- IH (1 bit): specifies if the inner IP address field appears in the Diet-ESP Payload Data. This bit is only significant for the tunnel mode. With IPsec transport mode, IH SHOULD be set to 0 and ignored. With tunnel mode IH unset to 0 indicates the inner IP header field is present and IH set to 1 indicates the inner IP header is omitted.
- ICV (2 bits): specifies the transmitted number of bytes to authenticate the Diet-ESP packet. Note that ICV is optional so if one chose not to perform authentication, it SHOULD negotiate the authentication algorithm to NULL as defined in [RFC4835]. The minimum length greater than 0 for ICV is 96 bits and can be generated with the following hash functions: HMAC-MD5-96 [RFC2403], HMAC-SHA1-96 [RFC2404], AES-CMAC-96 [RFC4494], AES-XCBC-MAC-96 [RFC3566]. As a result ICV only specifies size lower than 96 bits.
 - 000: ICV is left untouched as it is specified by the authentication algorithm.
 - 001: Diet-ESP ICV consists of the 8 most significant bits of ESP ICV.
 - 010: Diet-ESP ICV consists of the 16 most significant bits of ESP ICV.

- 011: Diet-ESP ICV consists of the 32 most significant bits of ESP ICV.
- 100: Diet-ESP ICV consists of the 64 most significant bits of ESP ICV.
- 101: Unassigned
- 110: Unassigned
- 111: Unassigned
- X (1 bit): Extension bit. When set to 1, this bit indicates an additional byte carry information. In this document, this bit MUST be set to 0.

6. 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 is familiar with these documents and list here the adaptation that MAY be involved by Diet-ESP.

Each device has an internal parameter that defines the minimal kernel alignment that is acceptable. The value HARD-ALIGN defines the minimum alignment allowed by the devices.

Diet-ESP Context with SPI SIZE + SN SIZE that is not a multiple of ALIGN MUST be rejected.

For devices using a single SPI SIZE value (e.g. sensors), the SA will be indexed with the SPI as described in ESP. More specifically, SPI is used as the index in the SAD. The only difference is that it has smaller size.

For devices that allow multiple SPI SIZE, like some IoT generic end points or IoT Security Gateways, SAD lookup has to deal with indexes of different sizes. One way would be to convert SPI of any size to a standard 4 bytes SPI. This means that for inbound packets a conversion from SPI SIZE bytes SPI to 4 bytes SPI is performed before the SAD lookup is performed. Similarly, a 4 bytes SPI to SPI SIZE bytes SPI conversion is performed before inserting the SPI into the Diet-ESP packet. A possible implementation may consist in using the SPI SIZE bytes SPI as the low order bytes of the 4 byte SPI and fill with NULL bytes the remaining high order bytes. This also means that any negotiated SPI must not start with NULL bytes. Another way could consist in adding the SPI SIZE argument in the SAD lookup. This

means that instead of looking at the SPI, implementations will consider looking at the (SPI, SPI SIZE).

The SPI of the SA may be negotiated using IKEv2 [RFC5996]. Regular IKEv2 implementation negotiate a 4 byte SPI. The SPI considered in Diet-ESP consists in the SPI SIZE low power bytes of this SPI. Only the value should be considered in the SAD. How the SPI SIZE SPI is represented in the SAD is another issue addressed above.

When the SPI is omitted, the device must be able to perform a SAD lookup that is not based on a SPI value, but only the IP addresses. This is specific to Diet-ESP, and one way to make Diet-ESP compliant with the IPsec architecture is to generate a SPI from the IP addresses. Most likely, no collision will occur. To avoid all collision cases, one can introduce a check function that proceed to the SPI and IP address match. If SPI matches but the IP address does not match then the hash function can be performed again over the previous SPI, until a match is found. Of course the system should define the maximum number of hashes that should be performed.

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.

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 Payload Data of lPD bytes, BLCK the block size of the encryption algorithm and the device alignment ALIGN. We note $M = \text{MAX}(\text{BLCK}, \text{ALIGN})$. Although not normative the resulting Diet-ESP packet should be and explained below. We consider the Diet-ESP Payload as described in Section 3

- 1: if TH is set to 1, then remove the transport layer of the Payload Data.
- 2: if IH is set to 1, and the IPsec mode is tunnel, then remove the inner IP address of the Payload Data.
- 3: if PAD is set to 0 and NH is set to 0: 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 such that $\text{lPD} + 2 + \text{Pad Length} = 0 \text{ [M]}$.
- 4: if PAD is set to 0 and NH is set to 1: 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 such that $lPD + 1 + \text{Pad Length} = 0 \text{ [M]}$.

- 5: if PAD is set to 1 and NH is set to 0: Diet-ESP considers the Next Header but do not consider the Pad Length field or the Padding Field. This is valid as long as $lPD + 1 = 0 \text{ [M]}$. 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: if PAD is set to 1 and NH is set to 1: Diet-ESP does consider neither the Next Header field nor the Pad Length field nor the Padding Field. This is valid as long as $lAD = 0 \text{ [M]}$. 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.

Decryption is performed the other way around.

After authenticating and encrypting the Diet-ESP payload the original packet is rebuild as follows:

- 1: if PAD is set to 1 and NH is set to 1: 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.
- 2: if PAD is set to 1 and NH is set to 0: 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.
- 3: if PAD is set to 0 and NH is set to 1: 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.
- 4: if PAD is set to 0 and NH is set to 0: 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.

- 5: if IH is set to 1, and the IPsec mode is tunnel and the IP header is reconstructed. The source and destination address and the Next Header field are read from the Traffic Selector. The Payload Length is calculated including the size of the transport header, regardless if it is removed with TH or not. All other IP-header values are set to common defaults or have to be negotiated otherwise which is out of scope of this document.
- 6: if TH is set to 1, the Transport layer header is restored with the information in the Security Association. Section 4 describes some differences between the different protocols. In this document we focus on UDP which can be easily restored with the ports inside the Traffic Selector. The Length field can be calculated and the checksum can be left as 0 according to [RFC0768]

7. IANA Considerations

There are no IANA consideration for this document.

8. Security Considerations

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

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.

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.

Removing the SN prevents protection against replay attack.

9. Acknowledgment

10. References

10.1. Normative 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.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2403] Madson, C. and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH", RFC 2403, November 1998.
- [RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.
- [RFC3566] Frankel, S. and H. Herbert, "The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec", RFC 3566, September 2003.
- [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.
- [RFC4494] Song, JH., Poovendran, R., and J. Lee, "The AES-CMAC-96 Algorithm and Its Use with IPsec", RFC 4494, June 2006.
- [RFC4835] Manral, V., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4835, April 2007.

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

10.2. Informational References

- [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. Comparison

This section compares the proposed Diet ESP with 6LoWPAN ESP [I-D.raza-6lowpan-ipsec] related to IoT use cases. It shows the different ESP packet sent with the two compression methods. In each case the maximum possible compression is used and the underlying UDP header is compressed as much as possible. The big advantage of Diet ESP compression removing the UDP header appears. Furthermore there are no additional compression configuration bytes to be sent in each packet, like done in 6LoWPAN compression, because the configuration is negotiated at the beginning of the during the IKEv2 [RFC5996] negotiation. Diet ESP uses the idea of ROHC[RFC5856] compression removing the disadvantage that the whole packet has to be sent once at the beginning of the connection, because it considers that a lot of information of the Security Association can be reused to decompress the packet.

Both comparisons are using 8 bits alignment. The figures are aligned to 16 bits to improve the readability.

A.1. Transmitting 1 Byte without anti-replay

6LoWPAN does not offer the possibility of removing the Sequence number. Therefore the minimum number of 16 bit has to be sent in each packet, even if it is not going to be used. If a SN of 8 or 24 bit should be used, similar characteristics can be recognized.

6LoWPAN does not offer to reduce the ICV as it is not removed with NULL-authentication. Diet-ESP offers reducing to fair securely 64 bits.

AES-CTR is used for encryption.

Figure 3a and 3b show this comparison. The advantage of Diet ESP for this example is 96 bits.

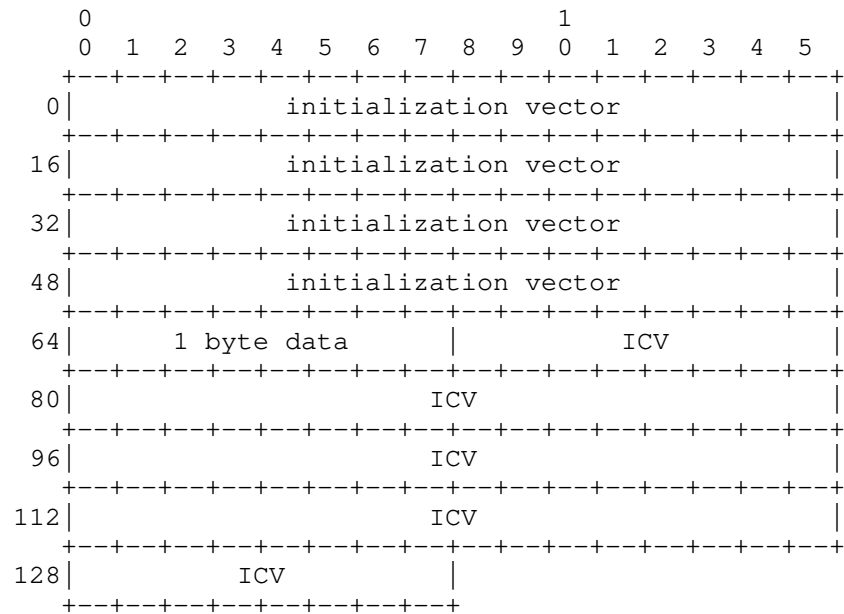


Figure 3a) 1 byte Data Payload with Diet-ESP.
(no SPI, no SN, no PAD, no NH)

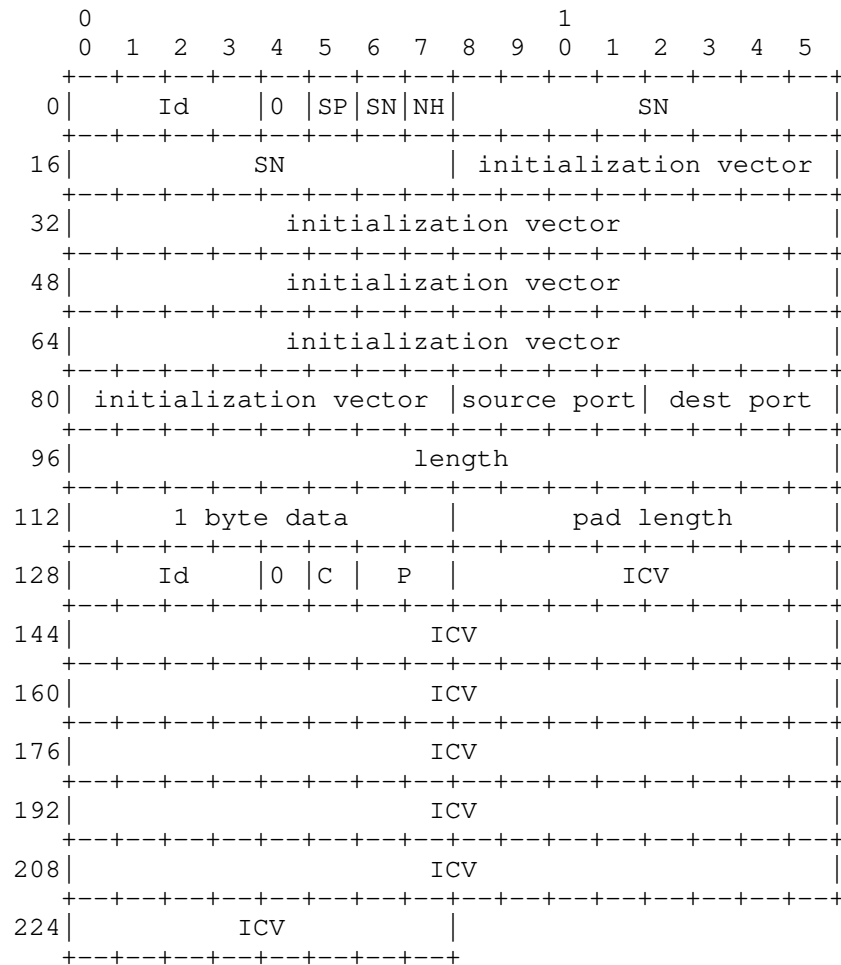


Figure 3b) 1 byte data payload with 6LoWPAN ESP.
 (no SPI, 16 bits SN, 8 bits pad length,
 8 bits 6LoWPAN NH)

A.2. Transmitting 1 Byte to multi directional connections.

Having multiple connections to one host implies the use of the SPI to identify the correct Security Association. Using 6LoWPAN ESP there is no possibility of reducing the SPI, it is only possible to remove it completely. Diet ESP allows the reduction to 8, 16 and 24 bit. In most sensor use cases 254 possible connection are more than enough, whereas the following two pictures show the advantage of Diet ESP against 6LoWPAN ESP for an 8 bit SPI. This advantage remains as long as there is no usage of the whole range of the 32 bit SPI, which

is extraordinary for sensors. Since there is no possibility to remove the SN with 6LoWPAN it has to be at least 16 Bit.

6LoWPAN does not offer the possibility of removing the Sequence number. Therefore the minimum number of 16 bit has to be sent in each packet, even if it is not going to be used. If a SN of 8 or 24 bit should be used, similar characteristics can be recognized.

6LoWPAN does not offer to reduce the ICV as it is not removed with NULL-authentication. Diet-ESP offers reducing to fair securely 64 bits.

AES-CTR is used for encryption.

Figure 4a and 4b show this comparison. In case of an 8 bit SPI the advantage is 120 bits.

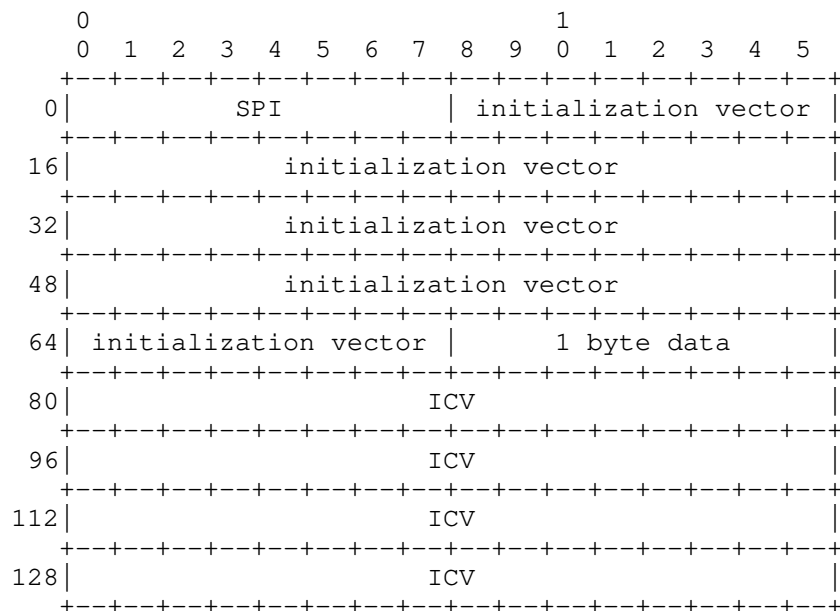


Figure 4a) 1 byte Data Payload with Diet-ESP.
(8 bits SPI, no SN, no PAD, no NH)

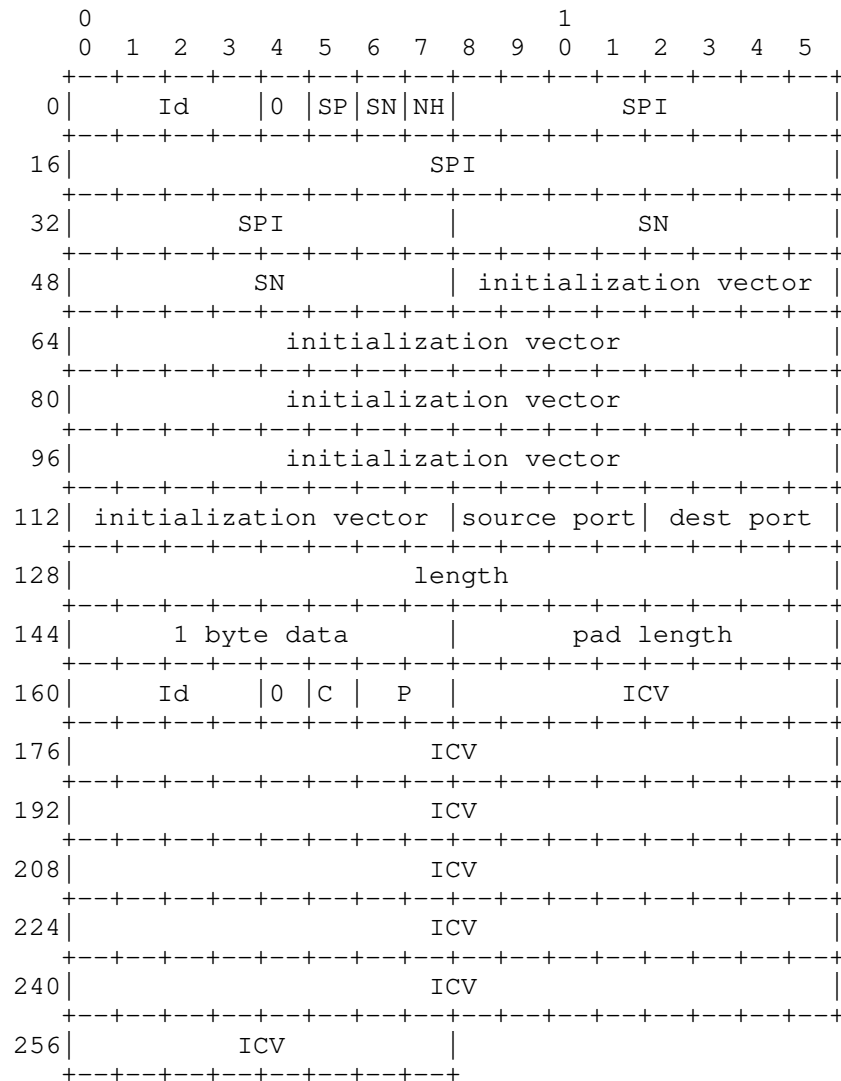


Figure 4b) 1 byte data payload with 6LoWPAN ESP.
 (32 bits SPI, 16 bits SN,
 8 bits pad length, 8 bits 6LoWPAN NH)

Appendix B. Document Change Log

[RFC Editor: This section is to be removed before publication]

-00: First version published.

Authors' Addresses

Daniel Migault
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
Orange / LMU Munich
Am Osteroesch 9
87637 Seeg, Bavaria
Germany

Email: tobias.guggemos@gmail.com

Daniel Palomares
Orange / LIP6 - UMPC
10, Rue du Moulin
92170 Vanves, Ile-de-France
France

Email: daniel.palomares@orange.com

Light-Weight Implementation Guidance (lwig)
Internet-Draft
Intended status: Informational
Expires: April 24, 2019

D. Migault
Ericsson
T. Guggemos
LMU Munich
October 21, 2018

Minimal ESP
draft-mglt-lwig-minimal-esp-07

Abstract

This document describes a minimal implementation of the IP Encapsulation Security Payload (ESP) defined in RFC 4303. Its purpose is to enable implementation of ESP with a minimal set of options to remain compatible with ESP as described in RFC 4303. A minimal version of ESP is not intended to become a replacement of the RFC 4303 ESP, but instead to enable a limited implementation to interoperate with implementations of RFC 4303 ESP.

This document describes what is required from RFC 4303 ESP as well as various ways to optimize compliance with RFC 4303 ESP.

This document does not update or modify RFC 4303, but provides a compact description of how to implement the minimal version of the protocol. If this document and RFC 4303 conflicts then RFC 4303 is the authoritative description.

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 April 24, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

ESP [RFC4303] is part of the IPsec suite protocol [RFC4301]. IPsec is used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity) and limited traffic flow confidentiality.

Figure 1 describes an ESP Packet. Currently ESP is implemented in the kernel of major multi purpose Operating Systems (OS). The ESP and IPsec suite is usually implemented in a complete way to fit multiple purpose usage of these OS. However, completeness of the IPsec suite as well as multi purpose scope of these OS is often performed at the expense of resources, or a lack of performance. As a result, constraint devices are likely to have their own implementation of ESP optimized and adapted to their specificities. With the adoption of IPsec by IoT devices with minimal IKEv2 [RFC7815] and ESP Header Compression (EHC) with [I-D.mglt-ipsecme-diet-esp] or [I-D.mglt-ipsecme-ikev2-diet-esp-extension], it becomes crucial that ESP implementation designed for constraint devices remain interoperable with the standard ESP implementation to avoid a fragmented usage of ESP. This document describes the the minimal properties and ESP implementation needs to meet.

For each field of the ESP packet represented in Figure 1 this document provides recommendations and guidance for minimal implementations. The primary purpose of Minimal ESP is to remain

interoperable with other nodes implementing RFC 4303 ESP, while limiting the standard complexity of the implementation.

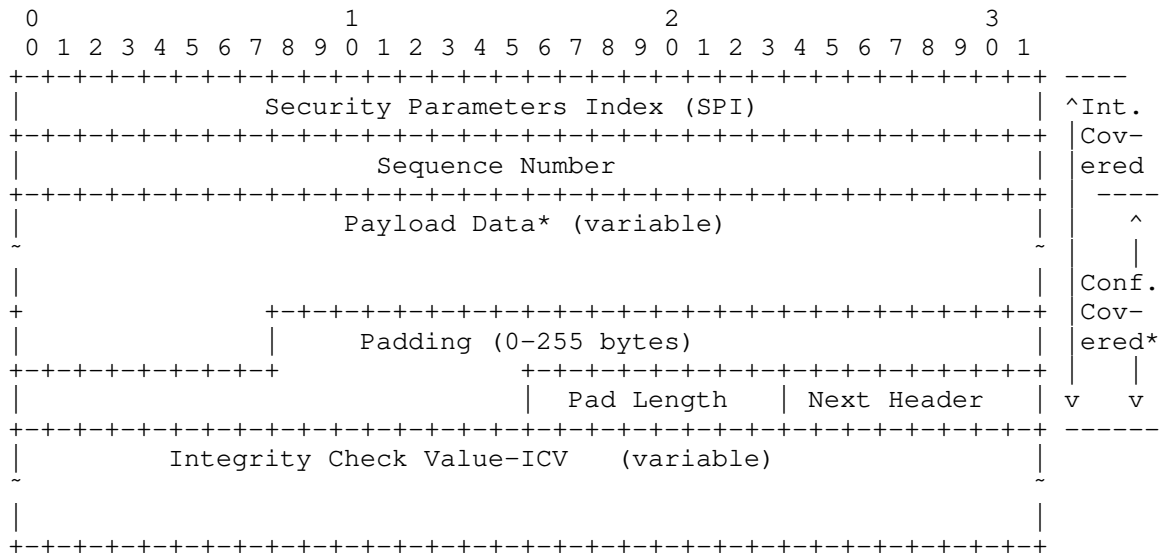


Figure 1: ESP Packet Description

3. Security Parameter Index (SPI) (32 bit)

According to the [RFC4303], the SPI is a mandatory 32 bits field and is not allowed to be removed.

The SPI has a local significance to index the Security Association (SA). From [RFC4301] section 4.1, nodes supporting only unicast communications can index their SA only using the SPI. On the other hand, nodes supporting multicast communications must also use the IP addresses and thus SA lookup needs to be performed using the longest match.

For nodes supporting only unicast communications, it is RECOMMENDED to index SA with the SPI only. Some other local constraints on the node may require a combination of the SPI as well as other parameters to index the SA.

It is RECOMMENDED to randomly generate the SPI indexing each inbound session. A random generation provides a stateless way to generate the SPIs, while keeping the probability of collision between SPIs relatively low. In case of collision, the SPI is simply re-generated.

However, for some constraint nodes, generating a random SPI may consume too much resource, in which case SPI can be generated using predictable functions or even a fix value. In fact, the SPI does not need to be random. Generating non random SPI MAY lead to privacy and security concerns. As a result, this alternative should be considered for devices that would be strongly impacted by the generation of a random SPI and after understanding the privacy and security impact of generating non random SPI.

When a constraint node uses fix value for SPIs, it imposes some limitations on the number of inbound SA. This limitation can be alleviated by how the SA lookup is performed. When fix SPI are used, it is RECOMMENDED the constraint node has as many SPI values as ESP session per host IP address, and that SA lookup includes the IP addresses.

Note that SPI value is used only for inbound traffic, as such the SPI negotiated with IKEv2 [RFC7296] or [RFC7815] by a peer, is the value used by the remote peer when it sends traffic. As SPI are only used for inbound traffic by the peer, this allows each peer to manage the set of SPIs used for its inbound traffic.

The use of fix SPI MUST NOT be considered as a way to avoid strong random generators. Such generator will be required in order to provide strong cryptographic protection and follow the randomness requirements for security described in [RFC4086]. Instead, the use of a fix SPI should only be considered as a way to overcome the resource limitations of the node, when this is feasible.

The use of a limited number of fix SPI or non random SPIs come with security or privacy drawbacks. Typically, a passive attacker may derive information such as the number of constraint devices connecting the remote peer, and in conjunction with data rate, the attacker may eventually determine the application the constraint device is associated to. If the SPI is fixed by a manufacturer or by some software application, the SPI may leak in an obvious way the type of sensor, the application involved or the model of the constraint device. When identification of the application or the hardware is associated to privacy, the SPI MUST be randomly generated. However, one needs to realize that in this case this is likely to be sufficient and a thorough privacy analysis is required. More specifically, traffic pattern MAY leak sufficient information in itself. In other words, privacy leakage is a complex and the use of random SPI is unlikely to be sufficient.

As the general recommendation is to randomly generate the SPI, constraint devices that will use a limited number of fix SPI are expected to be very constrained devices with very limited

capabilities, where the use of randomly generated SPI may prevent them to implement IPsec. In this case the ability to provision non random SPI enables these devices to secure their communications. These devices, due to there limitations, are expected to provide limited information and how the use of non random SPI impacts privacy requires further analysis. Typically temperature sensors, wind sensors, used outdoor do not leak privacy sensitive information. When used indoor, the privacy information is stored in the encrypted data and as such does not leak privacy.

As far as security is concerned, revealing the type of application or model of the constraint device could be used to identify the vulnerabilities the constraint device is subject to. This is especially sensitive for constraint devices where patches or software updates will be challenging to operate. As a result, these devices may remain vulnerable for relatively long period. In addition, predictable SPI enable an attacker to forge packets with a valid SPI. Such packet will not be rejected due to an SPI mismatch, but instead after the signature check which requires more resource and thus make DoS more efficient, especially for devices powered by batteries.

Values 0-255 SHOULD NOT be used. Values 1-255 are reserved and 0 is only allowed to be used internal and it MUST NOT be send on the wire.

[RFC4303] mentions :

"The SPI is an arbitrary 32-bit value that is used by a receiver to identify the SA to which an incoming packet is bound. The SPI field is mandatory. [...]"

"For a unicast SA, the SPI can be used by itself to specify an SA, or it may be used in conjunction with the IPsec protocol type (in this case ESP). Because the SPI value is generated by the receiver for a unicast SA, whether the value is sufficient to identify an SA by itself or whether it must be used in conjunction with the IPsec protocol value is a local matter. This mechanism for mapping inbound traffic to unicast SAs MUST be supported by all ESP implementations."

4. Sequence Number(SN) (32 bit)

According to [RFC4303], the Sequence Number (SN) is a mandatory 32 bits field in the packet.

The SN is set by the sender so the receiver can implement anti-replay protection. The SN is derived from any strictly increasing function that guarantees: if packet B is sent after packet A, then SN of packet B is strictly greater then the SN of packet A.

Some constraint devices may establish communication with specific devices, like a specific gateway, or nodes similar to them. As a result, the sender may know whereas the receiver implements anti-replay protection or not. Even though the sender may know the receiver does not implement anti replay protection, the sender **MUST** implement a always increasing function to generate the SN.

Usually, SN is generated by incrementing a counter for each packet sent. A constraint device may avoid maintaining this context and use another source that is known to always increase. Typically, constraint nodes using 802.15.4 Time Slotted Channel Hopping (TSCH), whose communication is heavily dependent on time, can take advantage of their clock to generate the SN. This would guarantee a strictly increasing function, and avoid storing any additional values or context related to the SN. When the use of a clock is considered, one should take care that packets associated to a given SA are not sent with the same time value.

For inbound traffic, it is **RECOMMENDED** to provide a anti-replay protection, and the size of the window depends on the ability of the network to deliver packet out of order. As a result, in environment where out of order packets is not possible the window size can be set to one. However, while **RECOMMENDED**, there is no requirements to implement an anti replay protection mechanism implemented by IPsec. A node **MAY** drop anti-replay protection provided by IPsec, and instead implement its own internal mechanism.

[RFC4303] mentions :

"This unsigned 32-bit field contains a counter value that increases by one for each packet sent, i.e., a per-SA packet sequence number. For a unicast SA or a single-sender multicast SA, the sender **MUST** increment this field for every transmitted packet. Sharing an SA among multiple senders is permitted, though generally not recommended. [...] The field is mandatory and **MUST** always be present even if the receiver does not elect to enable the anti-replay service for a specific SA."

5. Padding

The purpose of padding is to respect the 32 bit alignment of ESP. ESP **MUST** have at least one padding byte Pad Length that indicates the padding length. ESP padding bytes are generated by a succession of unsigned bytes starting with 1, 2, 3 with the last byte set to Pad Length, where Pad Length designates the length of the padding bytes.

Checking the padding structure is not mandatory, so the constraint device may not proceed to such checks, however, in order to

interoperate with existing ESP implementations, it MUST build the padding bytes as recommended by ESP.

In some situation the padding bytes may take a fix value. This would typically be the case when the Data Payload is of fix size.

[RFC4303] mentions :

"If Padding bytes are needed but the encryption algorithm does not specify the padding contents, then the following default processing MUST be used. The Padding bytes are initialized with a series of (unsigned, 1-byte) integer values. The first padding byte appended to the plaintext is numbered 1, with subsequent padding bytes making up a monotonically increasing sequence: 1, 2, 3, When this padding scheme is employed, the receiver SHOULD inspect the Padding field. (This scheme was selected because of its relative simplicity, ease of implementation in hardware, and because it offers limited protection against certain forms of "cut and paste" attacks in the absence of other integrity measures, if the receiver checks the padding values upon decryption.)"

ESP [RFC4303] also provides Traffic Flow Confidentiality (TFC) as a way to perform padding to hide traffic characteristics, which differs from respecting a 32 bit alignment. TFC is not mandatory and MUST be negotiated with the SA management protocol. TFC has not yet being widely adopted for standard ESP traffic. One possible reason is that it requires to shape the traffic according to one traffic pattern that needs to be maintained. This is likely to require extra processing as well as providing a "well recognized" traffic shape which could end up being counterproductive. As such TFC is not expected to be supported by a minimal ESP implementation.

As a result, TFC cannot not be enabled with minimal, and communication protection that were relying on TFC will be more sensitive to traffic shaping. This could expose the application as well as the devices used to a passive monitoring attacker. Such information could be used by the attacker in case a vulnerability is disclosed on the specific device. In addition, some application use - such as health applications - may also reveal important privacy oriented informations.

Some constraint nodes that have limited battery life time may also prefer avoiding sending extra padding bytes. However the same nodes may also be very specific to an application and device. As a result, they are also likely to be the main target for traffic shaping. In most cases, the payload carried by these nodes is quite small, and the standard padding mechanism may also be used as an alternative to TFC, with a sufficient trade off between the require energy to send

additional payload and the exposure to traffic shaping attacks. In addition, the information leaked by the traffic shaping may also be addressed by the application level. For example, it is preferred to have a sensor sending some information at regular time interval, rather when an specific event is happening. Typically a sensor monitoring the temperature, or a door is expected to send regularly the information - i.e. the temperature of the room or whether the door is closed or open) instead of only sending the information when the temperature has raised or when the door is being opened.

6. Next Header (8 bit)

According to [RFC4303], the Next Header is a mandatory 8 bits field in the packet. Next header is intended to specify the data contained in the payload as well as dummy packet. In addition, the Next Header may also carry an indication on how to process the packet [I-D.nikander-esp-beet-mode].

The ability to generate and receive dummy packet is required by [RFC4303]. For interoperability, it is RECOMMENDED a minimal ESP implementation discards dummy packets. Note that such recommendation only applies for nodes receiving packets, and that nodes designed to only send data may not implement this capability.

As the generation of dummy packets is subject to local management and based on a per-SA basis, a minimal ESP implementation may not generate such dummy packet. More especially, in constraint environment sending dummy packets may have too much impact on the device life time, and so may be avoided. On the other hand, constraint nodes may be dedicated to specific applications, in which case, traffic pattern may expose the application or the type of node. For these nodes, not sending dummy packet may have some privacy implication that needs to be measured. However, for the same reasons exposed in Section 5 traffic shaping at the IPsec layer may also introduce some traffic pattern, and on constraint devices the application is probably the most appropriated layer to limit the risk of leaking information by traffic shaping.

In some cases, devices are dedicated to a single application or a single transport protocol, in which case, the Next Header has a fix value.

Specific processing indications have not been standardized yet [I-D.nikander-esp-beet-mode] and is expected to result from an agreement between the peers. As a result, it is not expected to be part of a minimal implementation of ESP.

[RFC4303] mentions :

"The Next Header is a mandatory, 8-bit field that identifies the type of data contained in the Payload Data field, e.g., an IPv4 or IPv6 packet, or a next layer header and data. [...] the protocol value 59 (which means "no next header") MUST be used to designate a "dummy" packet. A transmitter MUST be capable of generating dummy packets marked with this value in the next protocol field, and a receiver MUST be prepared to discard such packets, without indicating an error."

7. ICV

The ICV depends on the crypto-suite used. Currently recommended [RFC8221] only recommend crypto-suites with an ICV which makes the ICV a mandatory field.

As detailed in Section 8 we recommend to use authentication, the ICV field is expected to be present that is to say with a size different from zero. This makes it a mandatory field which size is defined by the security recommendations only.

[RFC4303] mentions :

"The Integrity Check Value is a variable-length field computed over the ESP header, Payload, and ESP trailer fields. Implicit ESP trailer fields (integrity padding and high-order ESN bits, if applicable) are included in the ICV computation. The ICV field is optional. It is present only if the integrity service is selected and is provided by either a separate integrity algorithm or a combined mode algorithm that uses an ICV. The length of the field is specified by the integrity algorithm selected and associated with the SA. The integrity algorithm specification MUST specify the length of the ICV and the comparison rules and processing steps for validation."

8. Cryptographic Suites

The cryptographic suites implemented are an important component of ESP. The recommended suites to use are expected to evolve over time and implementer SHOULD follow the recommendations provided by [RFC8221] and updates. Recommendations are provided for standard nodes as well as constraint nodes.

This section lists some of the criteria that may be considered. The list is not expected to be exhaustive and may also evolve overtime. As a result, the list is provided as indicative:

1. Security: Security is the criteria that should be considered first for the selection of cipher suites. The security of cipher

suites is expected to evolve over time, and it is of primary importance to follow up-to-date security guidances and recommendations. The chosen cipher suites MUST NOT be known vulnerable or weak (see [RFC8221] for outdated ciphers). ESP can be used to authenticate only or to encrypt the communication. In the later case, authenticated encryption must always be considered [RFC8221].

2. **Interoperability:** Interoperability considers the cipher suites shared with the other nodes. Note that it is not because a cipher suite is widely deployed that is secured. As a result, security SHOULD NOT be weakened for interoperability. [RFC8221] and successors consider the life cycle of cipher suites sufficiently long to provide interoperability. Constraint devices may have limited interoperability requirements which makes possible to reduce the number of cipher suites to implement.
3. **Power Consumption and Cipher Suite Complexity:** Complexity of the cipher suite or the energy associated to it are especially considered when devices have limited resources or are using some batteries, in which case the battery determines the life of the device. The choice of a cryptographic function may consider re-using specific libraries or to take advantage of hardware acceleration provided by the device. For example if the device benefits from AES hardware modules and uses AES-CTR, it may prefer AUTH_AES-XCBC for its authentication. In addition, some devices may also embed radio modules with hardware acceleration for AES-CCM, in which case, this mode may be preferred.
4. **Power Consumption and Bandwidth Consumption:** Similarly to the cipher suite complexity, reducing the payload sent, may significantly reduce the energy consumption of the device. As a result, cipher suites with low overhead may be considered. To reduce the overall payload size one may for example:
 1. Use of counter-based ciphers without fixed block length (e.g. AES-CTR, or ChaCha20-Poly1305).
 2. Use of ciphers with capability of using implicit IVs [I-D.ietf-ipsecme-implicit-iv].
 3. Use of ciphers recommended for IoT [RFC8221].
 4. Avoid Padding by sending payload data which are aligned to the cipher block length - 2 for the ESP trailer.

9. IANA Considerations

There are no IANA consideration for this document.

10. Security Considerations

Security considerations are those of [RFC4303]. In addition, this document provided security recommendations and guidances over the implementation choices for each fields.

11. Acknowledgment

The authors would like to thank Daniel Palomares, Scott Fluhrer, Tero Kivinen, Valery Smyslov, Yoav Nir, Michael Richardson for their valuable comments.

12. References

12.1. Normative References

- [I-D.ietf-ipsecme-implicit-iv]
Migault, D., Guggemos, T., and Y. Nir, "Implicit IV for Counter-based Ciphers in Encapsulating Security Payload (ESP)", draft-ietf-ipsecme-implicit-iv-05 (work in progress), June 2018.
- [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>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [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>>.

- [RFC7815] Kivinen, T., "Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation", RFC 7815, DOI 10.17487/RFC7815, March 2016, <<https://www.rfc-editor.org/info/rfc7815>>.
- [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <<https://www.rfc-editor.org/info/rfc8221>>.

12.2. Informative References

- [I-D.mglt-ipsecme-diet-esp]
Migault, D., Guggemos, T., Bormann, C., and D. Schinazi, "ESP Header Compression and Diet-ESP", draft-mglt-ipsecme-diet-esp-06 (work in progress), May 2018.
- [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) Strategy", draft-mglt-ipsecme-ikev2-diet-esp-extension-01 (work in progress), June 2018.
- [I-D.nikander-esp-beet-mode]
Nikander, P. and J. Melen, "A Bound End-to-End Tunnel (BEET) mode for ESP", draft-nikander-esp-beet-mode-09 (work in progress), August 2008.

Appendix A. Document Change Log

[RFC Editor: This section is to be removed before publication]

-00: First version published.

-01: Clarified description

-02: Clarified description

Authors' Addresses

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

Email: daniel.migault@ericsson.com

Tobias Guggemos
LMU Munich
MNM-Team
Oettingenstr. 67
80538 Munich, Bavaria
Germany

Email: guggemos@mn-m-team.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 28, 2015

Y. Nir
Check Point
November 24, 2014

ChaCha20, Poly1305 and their use in IPsec
draft-nir-ipsecme-chacha20-poly1305-05

Abstract

This document describes the use of the ChaCha20 stream cipher along with the Poly1305 authenticator, combined into an AEAD algorithm for IPsec.

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 May 28, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Conventions Used in This Document	2
2. ESP_Chacha20-Poly1305 for ESP	3
2.1. AAD Construction	4
3. Use in IKEv2	4
4. UI Suite	4
5. Security Considerations	5
6. IANA Considerations	5
7. Acknowledgements	5
8. References	6
8.1. Normative References	6
8.2. Informative References	6
Author's Address	7

1. Introduction

The Advanced Encryption Standard (AES - [FIPS-197]) has become the gold standard in encryption. Its efficient design, wide implementation, and hardware support allow for high performance in many areas, including IPsec VPNs. On most modern platforms, AES is anywhere from 4x to 10x as fast as the previous most-used cipher, 3-key Data Encryption Standard (3DES - [FIPS-46]), which makes it not only the best choice, but the only choice.

The problem is that if future advances in cryptanalysis reveal a weakness in AES, VPN users will be in an unenviable position. With the only other widely supported cipher being the much slower 3DES, it is not feasible to re-configure IPsec installations to use 3DES. [standby-cipher] describes this issue and the need for a standby cipher in greater detail.

This document proposes the ChaCha20 stream cipher as such a standby cipher in an AEAD construction with the Poly1305 authenticator for use with the Encapsulated Security Protocol (ESP - [RFC4303]). We call this ESP_Chacha20-Poly1305. These algorithms are described in a separate document ([chacha_poly]). This document only describes the IPsec-specific things.

1.1. Conventions Used in This Document

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. ESP_ChaCha20-Poly1305 for ESP

ESP_ChaCha20-Poly1305 is a combined mode algorithm, or AEAD. The construction follows the AEAD construction in section 2.7 of [chacha_poly]:

- o The IV is 64-bit, and is used as part of the nonce.
- o A 32-bit sender ID is prepended to the 64-bit IV to form the 96-bit nonce. For regular IPsec, this is set to all zeros. IPsec extensions that allow multiple senders, such as GDOI ([RFC6407]) or [RFC6054] may set this to different values.
- o The encryption key is 256-bit.
- o The Internet Key Exchange protocol (IKE - [RFC7296]) generates a bitstring called KEYMAT that is generated from a PRF. That KEYMAT is divided into keys for encryption, message authentication and whatever else is needed. For the ChaCha20 algorithm, 256 bits are used for the key. TBD: do we want an extra 32 bits as salt for the nonce like in GCM?
- o The ChaCha20 encryption algorithm requires the following parameters: a 256-bit key, a 96-bit nonce, and a 32-bit initial block counter. For ESP we set these as follows:
 - * The key is set to the key mentioned above.
 - * The 96-bit nonce is formed from a concatenation of the 32-bit sender ID and the 64-bit IV, as described above.
 - * The Initial Block Counter is set to one (1). The reason that one is used for the initial counter rather than zero is that zero is reserved for generating the one-time Poly1305 key (see below)
- o As ChaCha20 is not a block cipher, no padding should be necessary. However, in keeping with the specification in RFC 4303, the ESP does have padding, so as to align the buffer to an integral multiple of 4 octets.
- o The same key and nonce, along with a block counter of zero are passed to the ChaCha20 block function, and the top 256 bits of the result are used as the Poly1305 key. The nonce passed to the block function here is the same nonce that is used in ChaCha20, including the 32-bit Sender ID bits, and the key passed is the same as the encryption key.
- o Finally, the Poly1305 function is run on the data to be authenticated, which is, as specified in section 2.7 of [chacha_poly] a concatenation of the following in the below order:
 - * The Authenticated Additional Data (AAD) - see Section 2.1.
 - * The AAD length in bytes as a 32-bit network order quantity.
 - * The ciphertext
 - * The length of the ciphertext as a 32-bit network order quantity.

- o The 128-bit output of Poly1305 is used as the tag. All 16 bytes are included in the packet.

The encryption algorithm transform ID for negotiating this algorithm in IKE is TBA by IANA.

2.1. AAD Construction

The construction of the Additional Authenticated Data (AAD) is similar to the one in [RFC4106]. For security associations (SAs) with 32-bit sequence numbers the AAD is 8 bytes: 4-byte SPI followed by 4-byte sequence number ordered exactly as it is in the packet. For SAs with ESN the AAD is 12 bytes: 4-byte SPI followed by an 8-byte sequence number as a 64-bit network order integer.

3. Use in IKEv2

AEAD algorithms can be used in IKE, as described in [RFC5282]. More specifically, the Encrypted Payload is as described in section 3 of that document, the IV is 64 bits, as described in Section 2, and the AAD is as described in section 5.1 of RFC 5282, so it's 32 bytes (28 for the IKEv2 header + 4 bytes for the encrypted payload header) assuming no unencrypted payloads.

4. UI Suite

This document also defines an RFC 4308-style UI suite for IKE and IPsec (See [RFC4308]). The suite is called "VPN-C". The name was chosen for two reasons:

- o "VPN-A" and "VPN-B" are already defined in RFC 4308.
- o "C" stands for "Civilian", because unlike VPN-A, VPN-B, and the additional UI suites defined in [RFC6379], most of the algorithm in this suite come from civilian researchers, not from government agencies.

The Algorithms:

ESP:

Encryption	ESP_Chacha20-Poly1305
Integrity	NULL

IKEv2:

Encryption	ESP_Chacha20-Poly1305
Integrity	NULL
Pseudo-random function	HMAC-SHA-256 [RFC4868]
Diffie-Hellman group	256-bit random ECP group [RFC5903]

HMAC-SHA-256 is used here because there is no natural way to use either ChaCha20 or Poly1305 as an IKEv2 PRF. See discussion in section 2.7 of [chacha_poly].

TBD: Do we want to define a special PRF function here? Something can be concocted from using ChaCha20 as the PRF function and Poly1305 for shortening keys, but somehow this looks unwieldy.

TBD: Should we replace the Diffie-Hellman group with ED25519 ???

5. Security Considerations

The ChaCha20 cipher is designed to provide 256-bit security.

The Poly1305 authenticator is designed to ensure that forged messages are rejected with a probability of $1-(n/(2^{102}))$ for a $16n$ -byte message, even after sending 2^{64} legitimate messages, so it is SUF-CMA in the terminology of [AE].

The most important security consideration in implementing this draft is the uniqueness of the nonce used in ChaCha20. The nonce should be selected uniquely for a particular key, but unpredictability of the nonce is not required. counters and LFSRs are both acceptable ways of generating unique nonces, as is encrypting a counter using a 64-bit cipher such as DES. Note that it is not acceptable to use a truncation of a counter encrypted with a 128-bit or 256-bit cipher, because such a truncation may repeat after a short time.

Another issue with implementing these algorithms is avoiding side channels. This is trivial for ChaCha20, but requires some care for Poly1305. Considerations for implementations of these algorithms are in the [chacha_poly] document.

6. IANA Considerations

IANA is requested to assign one value from the IKEv2 "Transform Type 1 - Encryption Algorithm Transform IDs" registry, with name ESP_Chacha20-Poly1305, and this document as reference.

IANA is also requested to assign the identifier "VPN-C" with this document as reference from the "Cryptographic Suites for IKEv1, IKEv2, and IPsec" registry.

7. Acknowledgements

All of the algorithms in this document were designed by D. J. Bernstein. The AEAD construction was designed by Adam Langley. The

author would also like to thank Adam for helpful comments, as well as Yaron Sheffer for telling me to write the algorithms draft.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.
- [RFC5282] Black, D. and D. McGrew, "Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol", RFC 5282, August 2008.
- [RFC6054] McGrew, D. and B. Weis, "Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic", RFC 6054, November 2010.
- [RFC7296] Kivinen, T., Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7296, October 2014.
- [chacha_poly] Langley, A. and Y. Nir, "ChaCha20 and Poly1305 for IETF protocols", draft-nir-cfrg-chacha20-poly1305-01 (work in progress), January 2014.

8.2. Informative References

- [AE] Bellare, M. and C. Namprempre, "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm", 2000, <<http://cseweb.ucsd.edu/~mihir/papers/oem.html>>.
- [FIPS-197] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001, <<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>>.
- [FIPS-46] National Institute of Standards and Technology, "Data Encryption Standard", FIPS PUB 46-2, December 1993, <<http://www.itl.nist.gov/fipspubs/fip46-2.htm>>.

- [RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, June 2005.
- [RFC4308] Hoffman, P., "Cryptographic Suites for IPsec", RFC 4308, December 2005.
- [RFC6379] Law, L. and J. Solinas, "Suite B Cryptographic Suites for IPsec", RFC 6379, October 2011.
- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, October 2011.
- [standby-cipher] McGrew, D., Grieco, A., and Y. Sheffer, "Selection of Future Cryptographic Standards", draft-mcgrew-standby-cipher (work in progress), January 2013.

Author's Address

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 6789735
Israel

Email: ynir.ietf@gmail.com

IPSECME
Internet-Draft
Intended status: Informational
Expires: September 6, 2014

D. Palomares (Ed)
D. Migault (Ed)
Orange
March 5, 2014

IKEv2/IPsec Context Definition
draft-plmrs-ipsecme-ipsec-ikev2-context-definition-01

Abstract

IKEv2/IPsec clusters are constituted of multiple nodes accessed via a single address by the end user. The traffic is then split between the nodes via specific IP load balancing policies. Once a session is assigned to a given node, IPsec makes it difficult to assign the session to another node. This makes management operations and transparent high availability for end users difficult to perform within the cluster.

This document describes the IKEv2 and IPsec contexts that MUST be transferred between nodes within a cluster so a session can be restored. This makes possible to transfer an IPsec session between different nodes.

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 September 6, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Requirements notation	2
2. Introduction	2
3. Terminology	3
4. Parameters level definition	3
5. IKEv2 key management	4
6. IKEv2 Session parameters	5
6.1. MANDATORY - IKEv2 Session parameters	5
6.2. OPTIONAL - IKEv2 Session parameters	5
6.3. VENDOR SPECIFIC - IKEv2 Session parameters	5
7. IPsec Session parameters	6
7.1. MANDATORY - IPsec Session parameters	6
7.2. OPTIONAL - IPsec Session parameters	6
7.3. VENDOR SPECIFIC - IPsec Session parameters	7
8. IANA Considerations	7
9. Security Considerations	7
10. Acknowledgment	7
11. References	7
11.1. Normative References	7
11.2. Informative References	7
Appendix A. ANNEX A: Data structure example	8
Appendix B. Document Change Log	9
Authors' Addresses	10

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

Large clusters may take advantage of the multiple nodes to enhance the peer's Quality of Service by performing among others:

- 1) Fail-over with high availability.
- 2) Load balancing among cluster members.

- 3) Scalability for overloaded IPsec platforms.
- 4) Compatibility for IKEv2/IPsec context transfers among different constructors.

This document addresses transfer of an IPsec session between physically or virtually different nodes within an IKEv2/IPsec cluster. More specifically, the document describes the parameters that **MUST** be transmitted between the IPsec/IKEv2 nodes, so that IKEv2 and IPsec session can be restored on the other node.

Currently IPsec based services can hardly benefit from these features as IPsec Security Associations are bound to a single node and cannot be shared among different cluster members.

This draft describes the parameters that **MUST** be transferred in order to keep an IKEv2/IPsec session alive in conformance with the Security Architecture for the Internet Protocol [RFC4301] and the Internet Key Exchange (IKEv2) Protocol [RFC5996].

This includes information such as the cryptographic material, the algorithms and the IP addresses, among others parameters.

Note that IKEv2 and IPsec session do not need to be on the same node as IKEv2 and IPsec context are different. Note also that we do not specify in this document how the IKEv2 or IPsec context are transferred between one node to the other. This can be performed via a simple UDP session that **MAY** be IPsec protected, a SCP session [RFC4251] or using the context transfer protocol [RFC4067].

3. Terminology

This document uses the following terminology:

IKE_SA context: the set of parameters composing a single IKE Security Association. A bidirectional communication will need a pair of **IKE_SAs**, for incoming and outgoing IKE exchanges.

IPsec_SA Context: the set of parameters composing a single IPsec Security Association. A bidirectional communication will need a pair of **IPsec_SAs** for incoming and outgoing traffic.

4. Parameters level definition

Information related to the IKEv2 and IPsec contexts can be defined within three different levels: mandatory, optional or vendor specific. This allows classification of the parameters considering

their relevance and susceptibility to be transferred in order to maintain an IKEv2/IPsec session alive.

- 1) Mandatory (M): Those parameters identified with a Mandatory flag (M) are considered absolutely relevant and necessary in order to maintain an IKE_SA or an IPsec_SA alive. The absence of a parameter with a mandatory flag, results in the loss of the IKE_SA or IPsec_SA.
- 2) Optional (O): Those parameters identified with an optional flag (O) are considered as additional information but are NOT absolutely necessary to maintain an IKEv2/IPsec session alive.
- 3) Vendor Specific (V): Those parameters identified with a vendor's specific flag (V) are considered as the information related to some specific constructor. It ensures enhancement provided by certain proprietary solutions when transmitting IKEv2/IPsec contexts, however, this MUST NOT interfere the interoperability with other IKEv2 and IPsec implementations and standards.

5. IKEv2 key management

Implementations might decide to manage sending cryptographic material (a.k.a. IKEv2/IPsec session keys) in different fashions; especially IKEv2 session keys. This document specifies three different ways to exchange IKEv2 keying information as follows:

- 1) Case 1: The node sends the private Diffie-Hellman key, the peer's KE content and nonces. In this case, the node receiving these information will recalculate all keys from the very beginning as it usually does during any initial IKEv2 exchange. The main drawback for this case is that recalculating keys is computational expensive, especially if thousands of session keys has to be calculated (e.g. during rush hours).
- 2) Case 2: A cluster member sends the SKEYSEED and nonces. In such case, the node receiving the information might not recalculate all the keys since the very beginning, but it still has to compute SK_* (SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, SK_pr).
- 3) Case : The cluster member sends all computed keys (SK_* = SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, SK_pr). In this case, the node receiving the keys wont need to recalculate keys from the beginning. However, this case demands more data to be sent between cluster members. Note that sending SK_pi/SK_pr may be omitted, as these keys are only used during authentication.

6. IKEv2 Session parameters

Considering IKEv2/IPsec sessions as bidirectional, we provide a list of parameters needed to create the IKE_SAs, which are usually stored in the user-land.

6.1. MANDATORY - IKEv2 Session parameters

- 1) Version of IKE: in this draft we only consider version 2.
- 2) The initiator flag and the responder flag for the IKE_SAs.
- 3) Local host address and remote host address (IPv4 or IPv6).
- 4) The IKE_SA's SPI of both initiator and responder.
- 5) The outgoing and incoming Message ID's.
- 7) The cryptographic material for the IKE_SA (see section Section 5 for details).
- 8) The [SA] proposal information: encryption algorithm, length of the encryption key, integrity algorithm, length of the integrity key and the pseudo random function (prf).
- 9) The extensions and condition of the IKE_SA (NAT, EAP, MOBIKE...).
- 10) The IDs of the initiator and responder (ID_IPV4_ADDR, ID_IPV6_ADDR, ID_FQDN, ID_RFC822_ADDR, ID_DER_ASN1_DN, ID_DER_ASN1_GN or ID_KEY_ID).
- 11) Credentials: pre-shared keys or digital certificates.
- 12) The windows bitmap value.

6.2. OPTIONAL - IKEv2 Session parameters

- 1) The IKE lifetime.
- 2) Vendors ID: when a vendors ID payload has been sent during IKE_SA negotiation, it is part of the IKE_SA parameters.

6.3. VENDOR SPECIFIC - IKEv2 Session parameters

For now, there are no vendor specific parameters for IKEv2.

7. IPsec Session parameters

Once the IKE_SAs are established for securing further IKEv2 exchanges, a pair of IPsec_SAs are negotiated in order to secure the traffic flow. The following list includes the parameters needed to build an IPsec_SA:

7.1. MANDATORY - IPsec Session parameters

- 1) Local host and remote host addresses (IPv4 or IPv6).
- 2) The inbound and outbound IPsec_SA Security Parameter Indexes (SPIs).
- 3) The IP compression information: flag for IPcomp. If active, The IPcomp Compression Parameter Index values (CPI IN, CPI OUT) and the the IPcomp algorithm.
- 4) The sequence number values: SN counter and SN overflow flag
- 5) The anti-replay window value.
- 6) IPsec mode: transport or tunnel mode.
- 7) The SA Lifetime: a time interval or byte count after which an SA must be replaced with a new SA (and new SPI).
- 8) Path MTU: maximum size of an IPsec packet that can be transmitted without fragmentation.
- 9) Upperspec: upper-layer protocol to be used.
- 10) Source IP/Destination IP addresses and ports of the protected traffic.
- 11) The IPsec protocol ESP and/or AH, their encryption/integrity algorithms and the key lengths.
- 12) The cryptographic material: KEYMAT (encryption and/or authentication keys).

7.2. OPTIONAL - IPsec Session parameters

For now, there are no optional parameters for IPsec sessions.

7.3. VENDOR SPECIFIC - IPsec Session parameters

- 1) Instance-id or flow-id: helps a node to identify which packet processing unit will process some IPsec traffic or which IPsec instance out of multiple IPsec processing units will process the IPsec traffic.

8. IANA Considerations

There are no IANA consideration for this document.

9. Security Considerations

Transferring an IPsec context between different SG involves sending sensitive information through the network. These pieces of information MUST be sent to an authenticated node via a secure channel.

10. Acknowledgment

IPsec cluster management is a joint work between Orange, Universite Pierre et Marie Curie / LIP6 and Institut Telecom / Telcom SudParis.

We would like to thank Maryline Laurent and Tobias Guggemos for their advises.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

11.2. Informative References

- [RFC4067] Loughney, J., Nakhjiri, M., Perkins, C., and R. Koodli, "Context Transfer Protocol (CXTTP)", RFC 4067, July 2005.

11.3. URIs

- [1] <http://tools.ietf.org/html/draft-plmrs-ipsecme-ipsec-ikev2-context-definition-01>
- [2] <http://tools.ietf.org/html/draft-plmrs-ipsecme-ipsec-ikev2-context-definition-00>

Appendix A. ANNEX A: Data structure example

Example of an IKEv2 data structure:

```
typedef struct _IKEV2CONTEXT
{
    bool *initiator;
    u_int32_t *ike_spi_i;
    u_int32_t *ike_spi_r;
    char *my_host;
    char *other_host;
    u_int16_t *enc_alg_ike;
    u_int16_t *enc_alg_ike_len;
    u_int16_t *int_alg_ike;
    u_int16_t *prf_alg;
    char *nonce_i;
    char *nonce_r;
    char *dh_secret;
    u_int16_t message_id;
    char *cert;
} IKEV2CONTEXT;
```

Example of an IPsec session data structure:

```
typedef struct _IPSECCONTEXT
{
    bool initiator;
    char *my_host;
    char *other_host;
    u_int8_t ipsec_mode;
    u_int16_t encr_alg_child;
    u_int16_t enc_alg_len_child;
    u_int16_t int_alg_child;
    u_int32_t enc_key_i;
    u_int32_t int_key_i;
    u_int32_t enc_key_o;
    u_int32_t int_key_o;
    char *child_seq_i;
    char *child_bit_i;
    char *child_seq_o;
    char *child_bit_o;
    char *child_spi_i;
    char *child_spi_o;
    u_int16_t ts_l_fromport;
    u_int16_t ts_l_toport;
    u_int8_t ts_l_type;
    u_int8_t ts_l_proto;
    char *ts_l_fromaddress;
    char *ts_l_toaddress;
    u_int16_t ts_r_fromport;
    u_int16_t ts_r_toport;
    u_int8_t ts_r_type;
    u_int8_t ts_r_proto;
    char *ts_r_fromaddress;
    char *ts_r_toaddress;
    bool ipcomp_flag;
    u_int32_t ipcom_algo;
    char *ipcomp_cpi_i;
    char *ipcomp_cpi_o;
} IPSECCONTEXT;
```

Appendix B. Document Change Log

[RFC Editor: This section is to be removed before publication]

draft-plmrs-ipsecme-ipsec-ikev2-context-definition-01 [1]
Added missing information as part of the IPsec and IKEv2 contexts
Worked on the text
Include mandatory, optional and vendor specific flags
Added three different ways send keys session keys

draft-plmrs-ipsecme-ipsec-ikev2-context-definition-00 [2]

initial draft.

Authors' Addresses

Daniel Palomares
Orange
38 rue du General Leclerc
92794 Issy-les-Moulineaux Cedex 9
France

Phone: +33 1 45 29 51 16
Email: danielpalomares.ietf@gmail.com

Daniel Migault
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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 8, 2014

Y. Sheffer
Porticor
Y. Nir
Check Point
February 4, 2014

The AutoVPN Architecture
draft-sheffer-autovpn-00

Abstract

This document describes the AutoVPN architecture. AutoVPN allows IPsec security associations to be set up with no prior configuration, using the "leap of faith" paradigm. The document defines a lightweight protocol for negotiating such opportunistic encryption either directly between hosts or between two security gateways on the path.

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 August 8, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terminology	4
3. Architecture and Protocol Overview	4
4. Protocol Exchanges	6
5. Message Format	7
5.1. ICMP Encoding	7
5.2. UDP Encoding	7
5.3. Protocol Payloads	8
5.4. Version Payload	9
5.5. Nonce Payloads	10
5.6. NAT-Detect Payload	10
6. Error Handling and Reliability	10
7. NAT Considerations	11
8. IKE Protocol Considerations	11
8.1. New IKE Payloads	12
8.1.1. AutoVPN Nonce	12
8.1.2. Contact Details	12
8.2. AUTOVPN_SHARED_SECRET Notification	12
9. Security Policy	12
9.1. Certificate States	12
9.2. Certificate Rollover and Permanent Association	14
9.3. Certificate Conflicts	14
9.4. Fallback to Clear	15
10. IANA Considerations	15
11. Security Considerations	15
12. Acknowledgements	15
13. References	15
13.1. Normative References	15
13.2. Informative References	16
Appendix A. Change Log	16
A.1. -00	16
Appendix B. Implementation Considerations	17
B.1. Address Authorization	17
B.2. Multiple Interfaces and Alternative Gateways	17
Authors' Addresses	17

1. Introduction

In the last few years, there have been several attempts to define an opportunistic encryption architecture, where network traffic is confidentiality-protected, even in the absence of proper authentication of the peers. This protection is often accompanied by continuity of identity, i.e. although the identity may not be

authenticated, it is verified to remain unchanged between multiple protocol sessions, and over long periods (days/weeks). This initial protection may be enhanced at a later stage, as peers gain stronger trust in each other's identity. A term which is often used to denote this policy is "leap of faith".

In the IPsec space, these attempts culminated in the BTNS (Better Than Nothing Security) working group's specifications. The BTNS working group produced a number of documents, including [RFC5386] and [RFC5387]. In addition, the earlier [RFC4322] describes Opportunistic Encryption, as implemented in various dialects of Linux (the history of Linux OE is summarized in a long post by Paul Wouters [oe-history]). However these specifications focus on the architectural IPsec implications, and provide insufficient context to implement the behavior described in the current document. "Leap of faith" has never been fully specified in the IPsec context, or when specified, assumes mechanisms that are still not widely deployed.

Similarly to many security architectures, a well designed opportunistic encryption solution requires both a robust protocol, and a user interaction component that allows the user to understand the exact security guarantees available at any time, so that the user may add external inputs about trustworthiness of communication peers while staying away from the "just press OK" mentality.

This document describes the AutoVPN architecture, an opportunistic encryption extension to the Internet Key Exchange v2 (IKEv2 - [RFC5996]) for IPsec VPN.

Some of the requirements behind this protocol are:

- o It should be suitable for business-to-business traffic, and therefore for deployment on the open Internet.
- o It should be robust, efficient and network friendly enough to be enabled by default.
- o It should be deployable on (existing) security gateways, rather than requiring changes to hosts.
- o It should also work on hosts that are not protected by gateways, i.e. hosts that are themselves IPsec endpoints.
- o It should require zero configuration. Some limited level of security should be provided by devices which are not configured.
- o After-the-fact security: the security guarantees can be improved at a later time, possibly using out-of-band means.

- o The protocol should coexist with regular IPsec, with no degradation in security.
- o The protocol should provide the best possible security given the imperfections of today's Internet. In particular, it should not rely on the deployment of DNS Security, anti spoofing mechanisms or routing security.
- o Small gateways, as well as software VPN clients, are often behind NAT. This scenario should be supported.

2. Terminology

We use the term "initiator" for the gateway through which came the original traffic. The gateway may not be the initiator of the new protocol described below. The other gateway is the "responder". Note that these terms correspond to the gateways' behavior with respect to IKE negotiation.

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].

3. Architecture and Protocol Overview

The protocol creates an IPsec tunnel to protect traffic which would otherwise be transmitted in the clear. What follows is a high level description of the sequence of operations.

We use H1 and H2 to denote two hosts (endpoints), and G1 and G2 to denote two IPsec gateways, protecting H1 and H2 respectively. This setup is shown in Figure 1 below. The solution described here is also applicable when one or both hosts is collocated with its respective gateway. Unfortunately it cannot be optimized for these cases, since source IP addresses can always be spoofed.

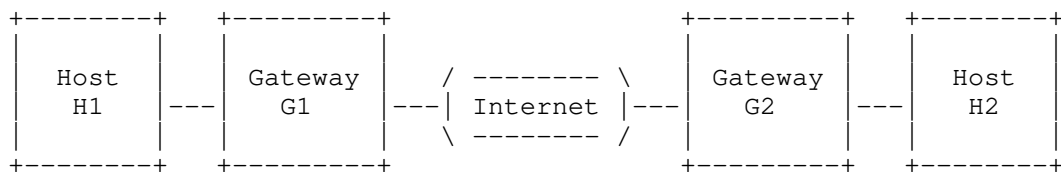


Figure 1: Deployment Architecture

Initially, only H1 knows of H2's address. The protocol below allows both intervening gateways to discover each other, and to gain

assurance that each one is on-path of the opposite host, i.e. it can see traffic addressed to the respective host, and can respond to such traffic.

We assume that both G1 and G2 contain access control functionality, as required by the IPsec architecture, and that both allow some clear traffic between H1 and H2.

The message sequence below is motivated to a great extent by the need to cater to NAT devices in front of G1, the original initiator. It is assumed that correctly implemented NAT devices will perform correct reverse translation of ICMP messages. However we cannot assume that they handle correctly ICMP messages of an unknown type.

Another major consideration is which side should drive the exchange. We have chosen the responder side (G2), since in an Internet where most traffic uses HTTP, the responder side knows best which traffic should be protected.

Lastly, we could have saved one round trip by allowing G2 to spoof H2's address. We believe this would have been ill advised.

The flow of messages is depicted in Figure 2.

- o H1 creates a network connection to H2, for example by sending a TCP SYN packet. H2 replies normally to H1.
- o G2 intercepts the reply packet, but lets it pass through. The connection proceeds normally, possibly including data packets.
- o G2 sends a Probe Request message, addressed to H1.
- o G1 intercepts the Probe Request, does not forward it, and sends a Probe Response, addressed to H2. Note that if H1 is NOT protected by a gateway, it will receive the Probe Request message and therefore the message should be designed to have no effect on innocent receivers.
- o G2 intercepts the Probe Response, does not forward it, and sends a Probe Complete, addressed to G1.
- o G1 now initiates an IKE_SA_INIT exchange to G2. This message includes payloads that can be correlated with the previous messages.
- o G1 and G2 negotiate an IPsec SA, potentially for all traffic between H1 and H2.

- o Both G1 and G2 now move the traffic between H1 and H2 into the new SA.

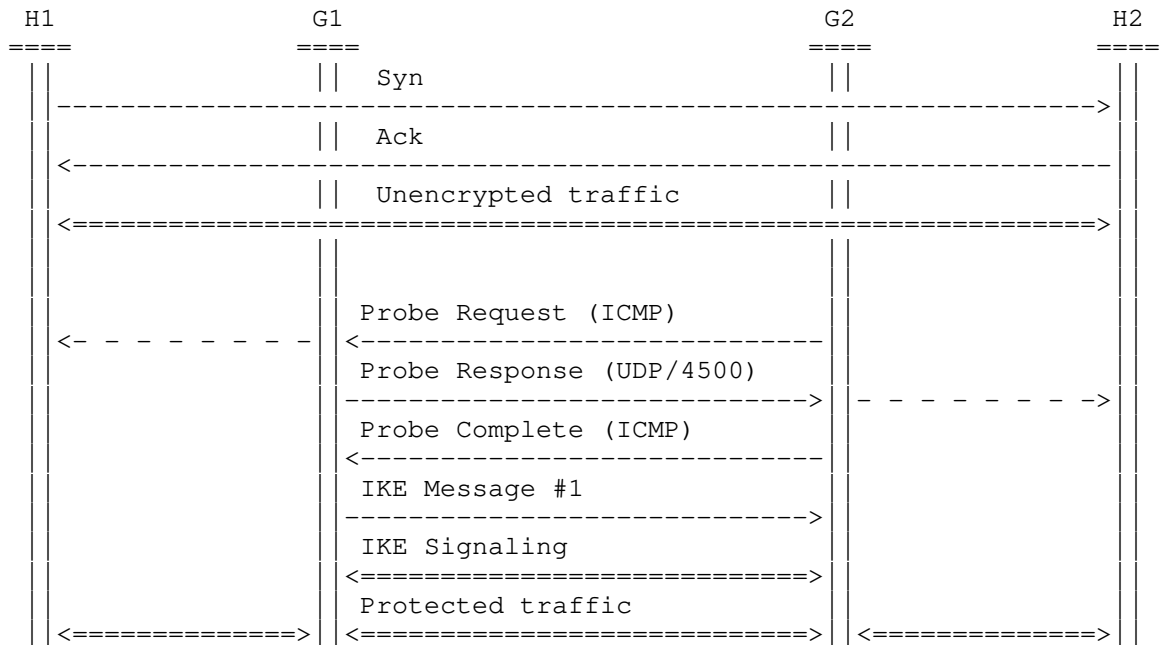


Figure 2: Message Sequence

4. Protocol Exchanges

AutoVPN consists of a 3-way probing protocol, followed by a slightly extended IKEv2 exchange.

The normal execution sequence of the protocol is as follows. G2 generates a fresh, randomly generated value Nonce-R, and sends to H1:

Probe Request: Version, Nonce-R, NAT-Detect

G1 intercepts the received message and does not forward it to H1. G1 MAY verify that the message corresponds to an ongoing connection, using the packet fragment contained in the ICMP envelope. G1 generates a fresh, random Nonce-I and sends to H2:

Probe Response: Version, Nonce-I, Nonce-R

where the content of Nonce-R is copied from the request. G2 intercepts this message, and does not forward it to H2. G2 MUST

verify that Nonce-R is valid, and silently ignore the message otherwise. G2 replies with:

Probe Complete: Version, Nonce-I, Nonce-R

G1 MUST check the validity of Nonce-I and Nonce-R. Finally, G1 sends an IKEv2 IKE_SA_INIT message to G2, containing a copy of the received Nonce-R.

5. Message Format

The AutoVPN protocol messages consist of a sequence of type-length-value (TLV) payloads. The messages are encoded in two different base protocols: ICMP and UDP over port 4500.

The Probe Request and Probe Complete messages MUST be encoded within ICMP. The Probe Response message MUST be encoded within UDP.

5.1. ICMP Encoding

Each payload is encoded as an ICMP Extension Object, as per [RFC4884]. ICMP Error messages contain a copy of (part of) the original packet, and this is used to associate the ICMP message with the original clear traffic. ICMP header fields are populated as follows:

- o Type is "Parameter Problem".
- o Code is the value 1.
- o The Checksum field MUST be computed, as per [RFC0792].
- o The new Length field is defined in [RFC4884].

In the Extension Object Headers, Class-Num is TBD by IANA, and C-Type is the Type value defined for each payload below.

5.2. UDP Encoding

In this encoding, all payloads are simply concatenated following the Preamble. Each payload is preceded by a payload header, as defined in Section 5.3.

The generic Preamble format is described in the next figure.

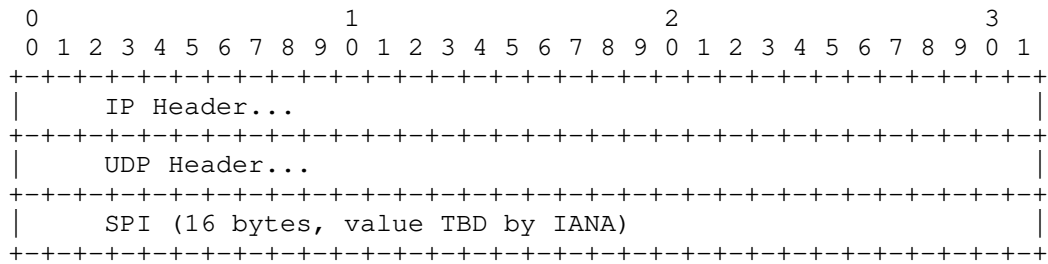


Figure 3: AutoVPN Preamble

The Probe Response protocol message has 4500 as its destination port. The protocol reuses the IKE/IPsec port 4500, however it is neither IKE nor IPsec. All three can coexist, and are distinguished using the SPI value. The specific SPI value will be allocated out of the "reserved" SPI space.

5.3. Protocol Payloads

An AutoVPN payload is encoded as an ICMP Extension Object or within a UDP message. When using UDP, the generic payload header is described in the next figure:

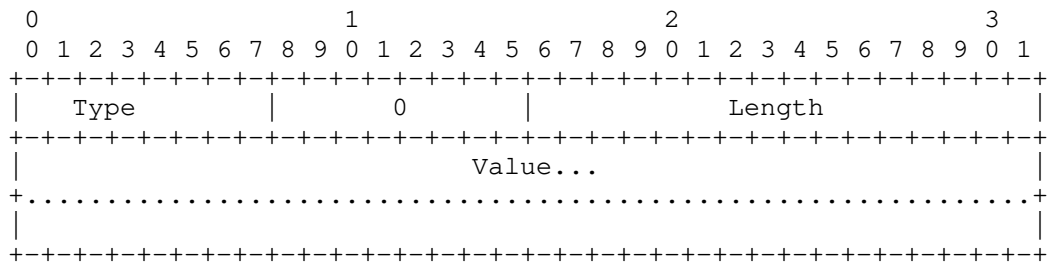


Figure 4: Payload Header

Type:

One of the payload types listed below.

Length:

The payload length in octets, including this header.

The following payload types are defined:

Name	Value	Definition
Unused	0	
Version	1	Generic information about the current message
Nonce-I	2	Initiator's nonce
Nonce-R	3	Responder's nonce
NAT-Detect	4	NAT detection information
	4-127	Reserved to IANA
	128-255	Reserved for private use

5.4. Version Payload

This payload is formatted as follows:

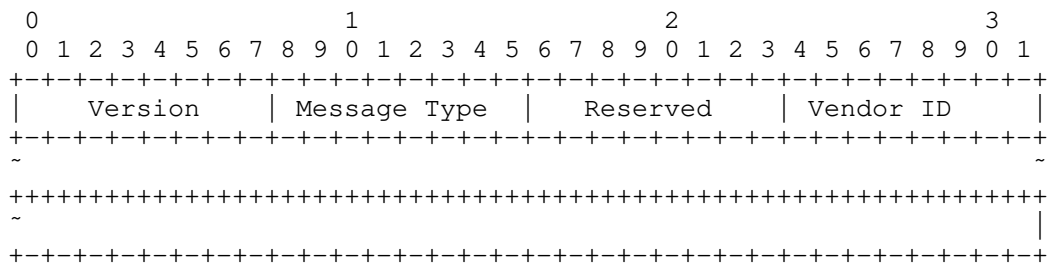


Figure 5: Version Payload

The header contains the following fields:

Version:

MUST be 0x01 for this version of the protocol.

Message Type:

1 for Probe Request, 2 for Probe Response, 3 for Probe Complete.
Other values are reserved to IANA.

Reserved:

MUST be sent as 0, and ignored by the receiver.

Vendor ID:

This field is optional and of variable length, possibly 0. It MAY contain a string uniquely identifying the vendor (e.g. "example.com"), or a binary string that is statistically unique (e.g. the SHA-1 hash of "we support the XX extension").

5.5. Nonce Payloads

Nonces are random or unpredictable values that enable the entity that generated them to recognize them as valid when it receives them again. Nonces MAY be used to encode state, in order to enable stateless implementations of this protocol. The length of each nonce (excluding the payload header) MUST be between 8 and 64 octets, inclusive.

One possible way to construct the nonce is

key-ID || HMAC-SHA256(K, gateway-IP || packet-fragment)

where K is a secret key known only to the sender, and key-ID identifies the key, enabling smooth roll-over of keys. Packet-fragment is the same portion of the packet as returned in an ICMP response, i.e. the IP header and the 8 octets that follow it.

5.6. NAT-Detect Payload

This payload consists of a 4-octet obfuscated IPv4 address, followed by a 2-octet port number. The address is obfuscated by a XOR operation with 0x0F0F0F0F, with the intention of defeating over-eager NAT devices which might try to rewrite the packet. The address and port are the source address/port of the original (clear) packet, as seen by the remote gateway (G2). The IP protocol (e.g. UDP or TCP) is inferred from the packet fragment included in the ICMP message containing this payload.

6. Error Handling and Reliability

The AutoVPN protocol is UDP and ICMP based, and therefore per-message reliability is not guaranteed. Both sides MAY retransmit the ICMP and UDP messages, but MUST NOT do so more than twice (total of 3 messages). The gateway (G2) that sends the first ICMP message MUST NOT retry a particular peer more than once every 24 hours.

The protocol does not include any error messages. If a peer does not accept a particular message for any reason, it MUST silently drop it. For forward compatibility, a receiver SHOULD process incoming

messages even if they contain payloads that it does not understand, and SHOULD ignore these payloads.

7. NAT Considerations

The current version of the protocol allows both sides to detect a NAT being performed between the gateways. Detection takes place during the probing phase. However this scenario raises several issues, which require further investigation before a useful solution can be proposed:

- o If traffic from a host which is behind NAT (H1) is inserted directly into an IPsec tunnel, it will emerge as-is on the other side, and the receiving host might see a non-routable [RFC1918] source address.
- o The initiating gateway may not have enough information to formulate its IKE Traffic Selector payload (TSi).

A solution that can be considered is for G1 to request a Tunnel Inner Address using an IKE Configuration Payload, and to perform NAT on traffic originating from H1 so that it appears to be sourced from that address. One of the issues with this solution is that the initial clear connection will necessarily be broken because of the address change.

We note that the protocol can be implemented correctly on a gateway that performs the NAT function itself.

8. IKE Protocol Considerations

The AutoVPN protocol imposes a few requirements on the IKE peers:

- o Both peers MUST use a certificate to authenticate. In many cases this is expected to be a self-signed certificate. But see also Section 9.2.
- o The peers MUST NOT negotiate any IPsec protocol, other than ESP in tunnel mode.
- o Each peer MUST offer only a single IP address in its negotiated traffic selector. This IP address MUST be identical to the one the gateway has proven authorization for. This MUST also be validated by the opposite peer. Per policy, traffic selectors MAY be even narrower, e.g. referring to specific protocol ports.

8.1. New IKE Payloads

This protocol defines several new IKE payloads.

8.1.1. AutoVPN Nonce

This payload has the payload type TBD by IANA. It MUST only be used in the first message of the IKE_SA_INIT exchange. The payload contains an exact copy of the Nonce-R AutoVPN payload, without the AutoVPN payload header.

8.1.2. Contact Details

This payload has the payload type TBD by IANA. It SHOULD be sent by both peers during the IKE_AUTH exchange. The payload contains a human readable UTF-8 string which is designed to assist the person managing the opposite protocol peer in verifying the sender's true identity. An example string is:

This gateway is operated by Example Inc. To validate our identity, you may wish to obtain our public key's fingerprint from our Web site, at <https://www.example.com/autovpn>. Or you may wish to contact the network administrator at 1-616-555-1212 to get the fingerprint. Please compare this value with the fingerprint value displayed by your gateway.

For obvious security reasons, this string MUST be rendered as plain text, and in particular MUST NOT be rendered as HTML.

8.2. AUTOVPN_SHARED_SECRET Notification

This notification, whose value is TBD by IANA, contains no data. It signifies that an AutoVPN shared secret MUST be created by the two IKE peers. See Section 9.2 for details.

9. Security Policy

This section describes the AutoVPN security policy, and should be viewed as an extension of [RFC4301].

9.1. Certificate States

AutoVPN defines a state for each peer gateway's certificate. A certificate may be in one of the following states (Figure 6):

- o Unknown. This may also be a certificate which had been manually removed, through a manual operation or for a number of reasons listed below.

- o Known but unverified. A DN is associated with a certificate's fingerprint, and additional information may be available ("contact details"). When not used, such certificates may be deleted from the table, after some site-specific timeout. It is RECOMMENDED that this timeout be larger than 7 days.
- o Trusted identity. The administrator can manually mark a certificate as Trusted. Alternatively, the certificate may have been signed by a trusted third party.
- o Untrusted identity. The administrator can manually mark a certificate as Untrusted, if he or she manually checks the certificate's fingerprint and detects a mismatch against an advertised value.
- o Managed. These certificates belong to peers that are part of the same managed VPN. They are not further discussed in this document.

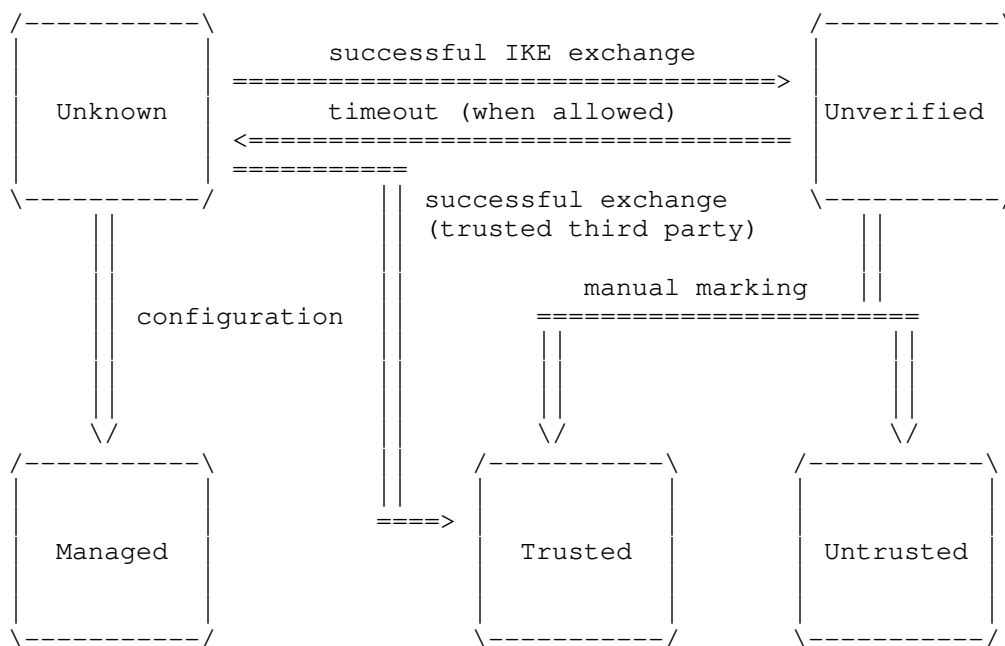


Figure 6: Certificate States

If the gateway detects that a peer's certificate has been explicitly revoked, it MUST delete this certificate from the table.

A PAD entry may exist for certificates in the Unverified, Managed or Trusted states. A PAD entry MUST NOT exist for a certificate in the Untrusted state, and IKE exchanges with peers presenting such certificates MUST be rejected, regardless of who initiated the exchange. When a certificate is deleted (whether manually or automatically) or marked as Untrusted, the associated PAD entry MUST be deleted.

When locating the peer, only the DN should be used. The peer's IP address MUST NOT be used, to allow peers to change their address.

9.2. Certificate Rollover and Permanent Association

In the absence of a certificate rollover mechanism, it would be impossible to distinguish between a legitimate peer presenting a new certificate and a MITM attacker. Therefore, AutoVPN gateways MUST support the shared secret mechanism described here.

As noted above, the information about a peer's certificate will normally time-out and be deleted. However any of the gateways can choose a convenient time to "promote" the association between the gateways, by triggering the creation of a shared secret. This secret never expires, other than through manual deletion on both peers.

The shared secret is associated with the pair of gateway identities, specifically with the IDi, IDr payloads exchanged between the gateways. Once a shared secret is established, both gateways MUST use it with the associated peer, in preference to certificate-based or other forms of authentication. A shared secret MAY be initiated for a peer in Unverified or Trusted state. On each gateway, the shared secret is associated with the peer gateway's certificate, and both MUST NOT be timed out regardless of the certificate's trust state.

The initiating gateway, which may be an IKE initiator or responder, MAY send the AUTOVPN_SHARED_SECRET notification at any time. The shared secret is the value

```
prf+(SK_d, "shared secret for AutoVPN")
```

where SK_d is the derivation key of the current IKE SA. The literal string is represented in ASCII, with no zero terminator.

9.3. Certificate Conflicts

A certificate conflict may be detected during the IKE exchange. This happens when an AutoVPN peer presents a certificate whose DN matches the DN of a known AutoVPN certificate, but which is different from

that certificate. In such cases the new peer MUST be rejected, with the notification AUTHENTICATION_FAILED.

As a result, barring incorrect configuration, the certificate table can never contain multiple certificates with the same DN.

9.4. Fallback to Clear

In some cases it may be desirable to allow fallback to clear traffic in cases where an IKE/IPsec association cannot be established, even when the peer is known. This is left to local policy, and SHOULD be configurable on the gateway. Such configuration MAY take the trust level of the peer gateway into account.

Moreover, some policies may prefer to send traffic unprotected, even when an IKE SA can be established, and then renegotiate an IPsec SA following some manual action. One possible way of doing this is using the mechanism described in [RFC6023].

10. IANA Considerations

TBD.

11. Security Considerations

TBD.

12. Acknowledgements

A proof of concept implementation of this protocol was created by Michael Rogovin at Check Point, and we would like to acknowledge his contribution.

13. References

13.1. Normative References

- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, April 2007.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

13.2. Informative References

- [RFC4322] Richardson, M. and D. Redelmeier, "Opportunistic Encryption using the Internet Key Exchange (IKE)", RFC 4322, December 2005.
- [RFC5386] Williams, N. and M. Richardson, "Better-Than-Nothing Security: An Unauthenticated Mode of IPsec", RFC 5386, November 2008.
- [RFC5387] Touch, J., Black, D., and Y. Wang, "Problem and Applicability Statement for Better-Than-Nothing Security (BTNS)", RFC 5387, November 2008.
- [RFC6023] Nir, Y., Tschofenig, H., Deng, H., and R. Singh, "A Childless Initiation of the Internet Key Exchange Version 2 (IKEv2) Security Association (SA)", RFC 6023, October 2010.
- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, February 2012.
- [oe-history] Wouters, P., "History and implementation status of Opportunistic Encryption for IPsec", September 2013, <<http://nohats.ca/wordpress/blog/2013/09/12/history-and-implementation-status-of-opportunistic-encryption-for-ipsec/>>.

Appendix A. Change Log

A.1. -00

Initial version.

Appendix B. Implementation Considerations

B.1. Address Authorization

Address authorization SHOULD be maintained separately from peer identity. Timing out authorization data (as proposed in [RFC4322]) is risky, since the authorization protocol allows a MITM, and also exposes clear traffic. This issue is TBD, and for now, authorization will only be removed when a peer is deleted.

We should look at alternative ways to prove address ownership. For example, if the gateway G1 can prove its ownership of a certain address range, it might send an RPKI [RFC6480] certificate to that effect, plus proof of possession in IKE_AUTH. The peer gateway G2 might then decide to allow a wider traffic selector including all of G1's addresses, instead of just H1.

Also TBD are the IPsec policy implications, within the framework of [RFC4301], Sec. 4.4.3.

B.2. Multiple Interfaces and Alternative Gateways

We might want to support multiple gateway addresses in the probing protocol, so we can have high quality connectivity without resorting to "fallback to the clear" (i.e. have very long timeouts, measured in days). On the other hand, maybe MOBIKE does the work in IKEv2.

Authors' Addresses

Yaron Sheffer
Porticor

Email: yaronf.ietf@gmail.com

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 6789735
Israel

Email: ynir@checkpoint.com