

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

C. Hopps
LabN Consulting, L.L.C.
March 11, 2019

IP Traffic Flow Security
draft-hopps-ipsecme-iptfs-00

Abstract

This document describes a mechanism to enhance IPsec traffic flow security by adding traffic flow confidentiality to encrypted IP encapsulated traffic. Traffic flow confidentiality is provided by obscuring the size and frequency of IP traffic using a fixed-sized, constant-send-rate IPsec tunnel. The solution allows for congestion control as well.

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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.

Table of Contents

1. Introduction	3
1.1. Terminology & Concepts	3
2. The IP-TFS Tunnel	3
2.1. Tunnel Content	4
2.1.1. IPSec/ESP Payload	4
2.1.2. Data-Blocks	5
2.1.3. No Implicit Padding	5
2.1.4. IP Header Value Mapping	6
2.2. Exclusive SA Use	6
2.3. Initiation of TFS mode	6
2.4. Example of an encapsulated IP packet flow	7
2.5. Modes of operation	7
2.5.1. Non-Congestion Controlled Mode	7
2.5.2. Congestion Controlled Mode	8
3. Congestion Information	9
3.1. ECN Support	9
4. Configuration	9
4.1. Bandwidth	10
4.2. Fixed Packet Size	10
4.3. Congestion Information Configuration	10
5. Packet and Data Formats	11
5.1. IPSec	11
5.1.1. Payload Format	11
5.1.2. Data Blocks	12
5.2. IKEv2	13
5.2.1. IKEv2 Congestion Information Configuration Attribute	13
5.2.2. IKEv2 Congestion Information Notification Data	14
6. IANA Considerations	15
7. Security Considerations	15
8. References	15
8.1. Normative References	15
8.2. Informative References	16
Appendix A. Comparisons of IP-TFS	17
A.1. Comparing Overhead	17
A.1.1. IP-TFS Overhead	17
A.1.2. ESP with Padding Overhead	18
A.2. Overhead Comparison	19
A.3. Comparing Available Bandwidth	19
A.3.1. Ethernet	20
Appendix B. Acknowledgements	22
Appendix C. Contributors	22
Author's Address	22

1. Introduction

Traffic Analysis ([RFC4301], [AppCrypt]) is the act of extracting information about data being sent through a network. While one may directly obscure the data through the use of encryption [RFC4303], the traffic pattern itself exposes information due to variations in it's shape and timing ([I-D.iab-wire-image], [AppCrypt]). Hiding the size and frequency of traffic is referred to as Traffic Flow Confidentiality (TFC) per [RFC4303].

[RFC4303] provides for TFC by allowing padding to be added to encrypted IP packets and allowing for sending all-pad packets (indicated using protocol 59). This method has the major limitation that it can significantly under-utilize the available bandwidth.

The IP-TFS solution provides for full TFC without the aforementioned bandwidth limitation. To do this we use a constant-send-rate IPsec [RFC4303] tunnel with fixed-sized encapsulating packets; however, these fixed-sized packets can contain partial, full or multiple IP packets to maximize the bandwidth of the tunnel.

For a comparison of the overhead of IP-TFS with the RFC4303 prescribed TFC solution see Appendix A.

Additionally, IP-TFS provides for dealing with network congestion [RFC2914]. This is important for when the IP-TFS user is not in full control of the domain through which the IP-TFS tunnel path flows.

1.1. Terminology & Concepts

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

This document assumes familiarity with IP security concepts described in [RFC4301].

2. The IP-TFS Tunnel

As mentioned in Section 1 IP-TFS utilizes an IPsec [RFC4303] tunnel as it's transport. To provide for full TFC we send fixed-sized encapsulating packets at a constant rate on the tunnel.

The primary input to the tunnel algorithm is the requested bandwidth of the tunnel. Two values are then required to provide for this

bandwidth, the fixed size of the encapsulating packets, and rate at which to send them.

The fixed packet size may either be specified manually or can be determined through the use of Path MTU discovery [RFC1191] and [RFC8201].

Given the encapsulating packet size and the requested tunnel bandwidth, the correct packet send rate can be calculated. The packet send rate is the requested bandwidth divided by the payload size of the encapsulating packet.

The egress of the IP-TFS tunnel SHOULD NOT impose any restrictions on tunnel packet size or arrival rate. Packet size and send rate is entirely the function of the ingress (sending) side of the IP-TFS tunnel. Indeed, the ingress (sending) side of the IP-TFS tunnel MUST be allowed by the egress side to vary the size and rate at which it sends encapsulating packets, including sending them larger, smaller, faster or slower than the requested size and rate.

2.1. Tunnel Content

As previously mentioned, one issue with the TFC padding solution in [RFC4303] is the large amount of wasted bandwidth as only one IP packet can be sent per encapsulating packet. In order to maximize bandwidth IP-TFS breaks this one-to-one association.

With IP-TFS we fragment as well as aggregate the inner IP traffic flow into fixed-sized encapsulating IP tunnel packets. We only pad the tunnel packets if there is no data available to be sent at the time of tunnel packet transmission.

In order to do this we create a new payload data type identified with a new IP protocol number IPTFS_PROTOCOL (TBD). A payload of IPTFS_PROTOCOL type is comprised of a 32 bit header followed by either a partial, a full or multiple partial or full data-blocks.

2.1.1. IPSec/ESP Payload

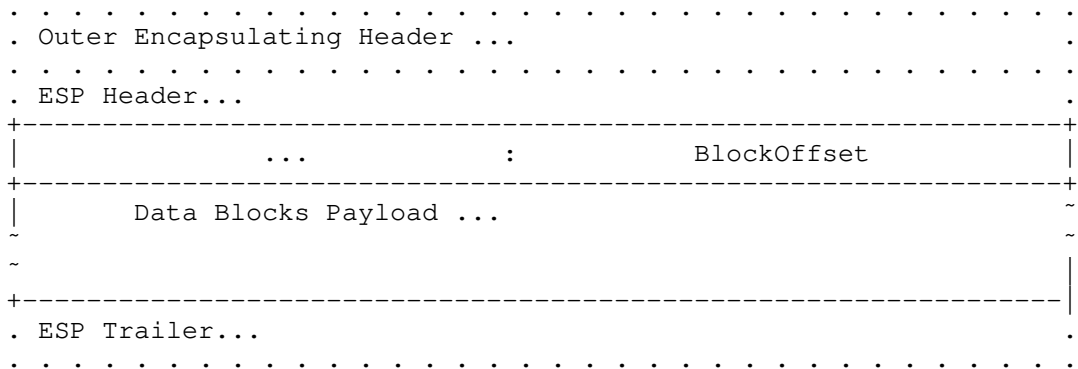


Figure 1: Layout of IP-TFS IPsec Packet

The BlockOffset value is either zero or some offset into or past the end of the data blocks payload data. If the value is zero it means that a new data-block immediately follows the fixed header (i.e., the BlockOffset value). Conversely, if the BlockOffset value is non-zero it points at the start of the next data block. The BlockOffset can point past the end of the data block payload data, this means that the next data-block occurs in a subsequent encapsulating packet. When the BlockOffset is non-zero the data immediately following the header belongs to the previous data-block that is still being re-assembled.

2.1.2. Data-Blocks

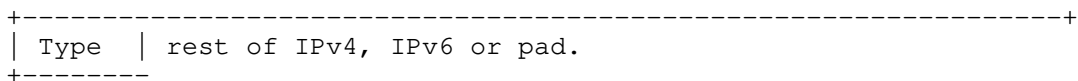


Figure 2: Layout of IP-TFS data block

A data-block is defined by a 4-bit type code followed by the data block data. The type values have been carefully chosen to coincide with the IPv4/IPv6 version field values so that no per-data-block type overhead is required to encapsulate an IP packet. Likewise, the length of the data block is extracted from the encapsulated IPv4 or IPv6 packet's length field.

2.1.3. No Implicit Padding

It's worth noting that there is no need for implicit pads at the end of an encapsulating packet. Even when the start of a data block occurs near the end of a encapsulating packet such that there is no room for the length field of the encapsulated header to be included

in the current encapsulating packet, the fact that the length comes at a known location and as is guaranteed to be present is enough to fetch the length field from the subsequent encapsulating packet payload.

2.1.4. IP Header Value Mapping

[RFC4301] provides some direction on when and how to map various values from an inner IP header to the outer encapsulating header, namely the Don't-Fragment (DF) bit ([RFC0791] and [RFC8200]), the Differentiated Services (DS) field [RFC2474] and the Explicit Congestion Notification (ECN) field [RFC3168]. Unlike [RFC4301] with IP-TFS we may and often will be encapsulating more than 1 IP packet per ESP packet. To deal with this we further restrict these mappings. In particular we never map the inner DF bit as it is unrelated to the IP-TFS tunnel functionality; we never directly fragment the inner packets and the inner packets will not affect the fragmentation of the outer encapsulation packets. Likewise, the ECN value need not be mapped as any congestion related to the constant-send-rate IP-TFS tunnel is unrelated (by design!) to the inner traffic flow. Finally, by default the DS field SHOULD NOT be copied although an implementation MAY choose to allow for configuration to override this behavior. An implementation SHOULD also allow the DS value to be set by configuration.

2.2. Exclusive SA Use

It is not the intention of this specification to allow for mixed use of an IPsec SA. In other words, an SA that is created for IP-TFS is exclusively for IP-TFS use and MUST NOT have non-IP-TFS payloads such as IP (IP protocol 4), TCP transport (IP protocol 6), or ESP pad packets (protocol 59) intermixed with IP-TFS (IP protocol TBD) payloads. While it's possible to envision making the algorithm work in the presence of sequence number skips in the IP-TFS payload stream, the added complexity is not deemed worthwhile. Other IPsec uses can configure and use their own SAs.

2.3. Initiation of TFS mode

While normally a user will configure their IPsec tunnel to operate in IP-TFS mode to start, we also allow IP-TFS mode to be enabled post-SA creation. This may be useful for debugging or other purposes. In this late enabled mode the receiver would switch to IP-TFS mode on receipt of the first ESP payload with the IPTFS_PROTOCOL indicated as the payload type.

2.4. Example of an encapsulated IP packet flow

Below we show an example inner IP packet flow within the encapsulating tunnel packet stream. Notice how encapsulated IP packets can start and end anywhere, and more than one or less than 1 may occur in a single encapsulating packet.

```

Offset: 0      Offset: 100    Offset: 2900    Offset: 1400
[ ESP1  (1500) ][ ESP2  (1500) ][ ESP3  (1500) ][ ESP4  (1500) ]
[--800--][--800--][60][-240-][--4000-----][pad]

```

Figure 3: Inner and Outer Packet Flow

The encapsulated IP packet flow (lengths include IP header and payload) is as follows: an 800 octet packet, an 800 octet packet, a 60 octet packet, a 240 octet packet, a 4000 octet packet.

The BlockOffset values in the 4 IP-TFS payload headers for this packet flow would thus be: 0, 100, 2900, 1400 respectively. The first encapsulating packet ESP1 has a zero BlockOffset which points at the IP data block immediately following the IP-TFS header. The following packet ESP2s BlockOffset points inward 100 octets to the start of the 60 octet data block. The third encapsulating packet ESP3 contains the middle portion of the 4000 octet data block so the offset points past its end and into the forth encapsulating packet. The fourth packet ESP4s offset is 1400 pointing at the padding which follows the completion of the continued 4000 octet packet.

Having the BlockOffset always point at the next available data block allows for quick recovery with minimal inner packet loss in the presence of outer encapsulating packet loss.

2.5. Modes of operation

Just as with normal IPsec tunnels IP-TFS tunnels are unidirectional. Bidirectional functionality is achieved by setting up 2 tunnels, one in either direction.

An IP-TFS tunnel can operate in 2 modes, a non-congestion controlled mode and congestion controlled mode.

2.5.1. Non-Congestion Controlled Mode

In the non-congestion controlled mode IP-TFS sends fixed-sized packets at a constant rate. The packet send rate is constant and is not automatically adjusted regardless of any network congestion (i.e., packet loss).

For similar reasons as given in [RFC7510] the non-congestion controlled mode should only be used where the user has full administrative control over the path the tunnel will take. This is required so the user can guarantee the bandwidth and also be sure as to not be negatively affecting network congestion [RFC2914]. In this case packet loss should be reported to the administrator (e.g., via syslog, YANG notification, SNMP traps, etc) so that any failures due to a lack of bandwidth can be corrected.

2.5.2. Congestion Controlled Mode

With the congestion controlled mode, IP-TFS adapts to network congestion by lowering the packet send rate to accommodate the congestion, as well as raising the rate when congestion subsides.

If congestion were handled in the network on a octet level we might consider lowering the IPsec (encapsulation) packet size to adapt; however, as congestion is normally handled in the network by dropping packets we instead choose to lower the frequency we send our fixed sized packets. This choice also minimizes transport overhead.

The output of a congestion control algorithm SHOULD adjust the frequency that ingress sends packets until the congestion is accommodated. While this document does not standardize the congestion control algorithm, the algorithm used by an implementation SHOULD conform to the guidelines in [RFC2914].

When an implementation is choosing a congestion control algorithm it is worth noting that IP-TFS is not providing for reliable delivery of IP traffic and so per packet ACKs are not required, and are not provided.

It's worth noting that the adjustable rate of sending over the congestion controlled IP-TFS tunnel is being controlled by the network congestion. As long as the encapsulated traffic flow shape and timing are not directly affecting the network congestion, the variations in the tunnel rate will not weaken the provided traffic flow confidentiality.

2.5.2.1. Circuit Breakers

In addition to congestion control, implementations MAY choose to define and implement circuit breakers [RFC8084] as a recovery method of last resort. Enabling circuit breakers is also a reason a user may wish to enable congestion information reports even when using the non-congestion controlled mode of operation. The definition of circuit breakers are outside the scope of this document.

3. Congestion Information

In order to support the congestion control mode, the receiver (egress tunnel endpoint) MUST send regular packet drop reports to the sender (ingress tunnel endpoint). These reports indicate the number of packet drops during a sequence of packets. The sequence or range of packets is identified using the start and end ESP sequence numbers of the packet range.

These congestion information reports MAY also be sent when in the non-congestion controlled mode to allow for reporting from the sending device or to implement Circuit Breakers [RFC8084].

The congestion information is sent using an IKEv2 INFORMATION notifications [RFC7296]. These notifications are sent at a configured interval (which can be configured to 0 to disable the sending of the reports).

3.1. ECN Support

In addition to normal packet loss information IP-TFS supports use of the ECN bits in the encapsulating IP header [RFC3168] for identifying congestion. If ECN use is enabled and a packet arrives at the egress endpoint with the Congestion Experienced (CE) value set, then the receiver records that packet as being dropped, although it does not drop it. When the CE information is used to calculate the packet drop count the receiver also sets the E bit in the congestion information notification data. In order to respond quickly to the congestion indication the receiver MAY immediately send a congestion information notification to the sender upon receiving a packet with the CE indication. This additional immediate send SHOULD only be done once per normal congestion information sending interval though.

As noted in [RFC3168] the ECN bits are not protected by IPsec and thus may constitute a covert channel. For this reason ECN use SHOULD NOT be enabled by default.

4. Configuration

IP-TFS is meant to be deployable with a minimal amount of configuration. All IP-TFS specific configuration (i.e., in addition to the underlying IPsec tunnel configuration) should be able to be specified at the tunnel ingress (sending) side alone (i.e., single-ended provisioning).

4.1. Bandwidth

Bandwidth is a local configuration option. For non-congestion controlled mode the bandwidth SHOULD be configured. For congestion controlled mode one can configure the bandwidth or have no configuration and let congestion control discover the maximum bandwidth available. No standardized configuration method is required.

4.2. Fixed Packet Size

The fixed packet size to be used for the tunnel encapsulation packets can be configured manually or can be automatically determined using Path MTU discovery (see [RFC1191] and [RFC8201]). No standardized configuration method is required.

4.3. Congestion Information Configuration

If congestion control mode is to be used, or if the user wishes to receive congestion information on the sender for circuit breaking or other operational notifications in the non-congestion controlled mode, IP-TFS will need to configure the egress tunnel endpoint to send congestion information periodically.

In order to configure the sending interval of periodic congestion information on the egress tunnel endpoint, we utilize the IKEv2 Configuration Payload (CP) [RFC7296]. Implementations MAY also allow for manual (or default) configuration of this interval; however, implementations of IP-TFS MUST support configuration using the IKEv2 exchange described below.

We utilize a new IKEv2 configuration attribute TFS_INFO_INTERVAL (TBD) to configure the sending interval from the egress endpoint of the tunnel. This value is configured using a CFG_REQUEST payload and is acknowledged by the receiver using a CFG_REPLY payload. This configuration exchange SHOULD be sent during the IKEv2 configuration exchanges occurring as the tunnel is first brought up. The sending interval value MAY also be changed at any time afterwards using a similar CFG_REQUEST/CFG_REPLY payload inside an IKEv2 INFORMATIONAL exchange.

In the absence of a congestion information configuration exchange the sending interval is up to the receiving device configuration.

The sending interval value is given in milliseconds and is 16 bits wide; however, it is not recommended that values below 1/10th of a second are used as this could lead to early exhaustion of the Message

ID field used in the IKEv2 INFORMATIONAL exchange to send the congestion information.

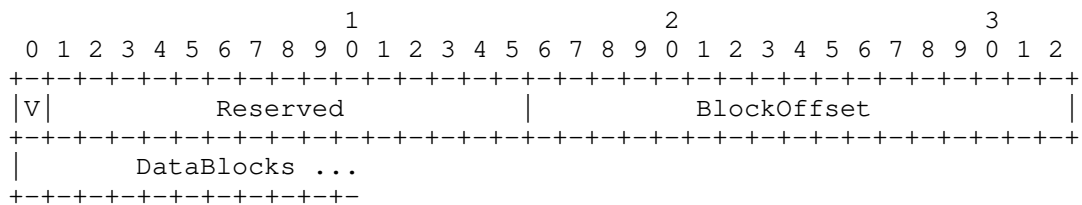
```
{{question: Could we get away with sending the info using the same
message ID each time? We have a timestamp that would allow for
duplicate detection, and the payload will be authenticated by IKEv2.
}}
```

A sending interval value of 0 disables sending of the congestion information.

5. Packet and Data Formats

5.1. IPSec

5.1.1. Payload Format



V:

A 1 bit version field that MUST be set to zero. If received as one the packet MUST be dropped.

Reserved:

A 15 bit field set to 0 and ignored on receipt.

BlockOffset:

A 16 bit unsigned integer counting the number of octets following this 32 bit header before the next data block. It can also point past the end of the containing packet in which case the data entirely belongs to the previous data block. If the offset extends into subsequent packets the subsequent 32 bit IP-TFS headers are not counted by this value.

DataBlocks:

Variable number of octets that constitute the start or continuation of a previous data block.

5.1.2. Data Blocks

```

          1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Type | IPv4, IPv6 or pad...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Type:

A 4 bit field where 0x0 identifies a pad data block, 0x4 indicates an IPv4 data block, and 0x6 indicates an IPv6 data block.

5.1.2.1. IPv4 Data Block

```

          1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x4 | IHL | TypeOfService | TotalLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Rest of the inner packet ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

These values are the actual values within the encapsulated IPv4 header. In other words, the start of this data block is the start of the encapsulated IP packet.

Type:

A 4 bit value of 0x4 indicating IPv4 (i.e., first nibble of the IPv4 packet).

TotalLength:

The 16 bit unsigned integer length field of the IPv4 inner packet.

5.1.2.2. IPv6 Data Block

```

          1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x6 | TrafficClass | FlowLabel |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| TotalLength | Rest of the inner packet ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

These values are the actual values within the encapsulated IPv6 header. In other words, the start of this data block is the start of the encapsulated IP packet.

Type:

A 4 bit value of 0x6 indicating IPv6 (i.e., first nibble of the IPv6 packet).

TotalLength:

The 16 bit unsigned integer length field of the inner IPv6 inner packet.

5.1.2.3. Pad Data Block

```

                                1                2                3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
      +-----+-----+-----+-----+-----+-----+-----+-----+
      | 0x0   | Padding ... |-----+-----+-----+-----+-----+
      +-----+-----+-----+-----+-----+-----+-----+-----+

```

Type:

A 4 bit value of 0x0 indicating a padding data block.

Padding:

extends to end of the encapsulating packet.

5.2. IKEv2

5.2.1. IKEv2 Congestion Information Configuration Attribute

The following defines the configuration attribute structure used in the IKEv2 [RFC7296] configuration exchange to set the congestion information report sending interval.

```

                                1                2                3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
      +-----+-----+-----+-----+-----+-----+-----+-----+
      |R|      Attribute Type      |      Length      |
      +-----+-----+-----+-----+-----+-----+-----+-----+
      |      Interval      |-----+-----+-----+-----+-----+
      +-----+-----+-----+-----+-----+-----+-----+-----+

```

R:

1 bit set to 0.

Attribute Type:

15 bit value set to TFS_INFO_INTERVAL (TBD).

Length:

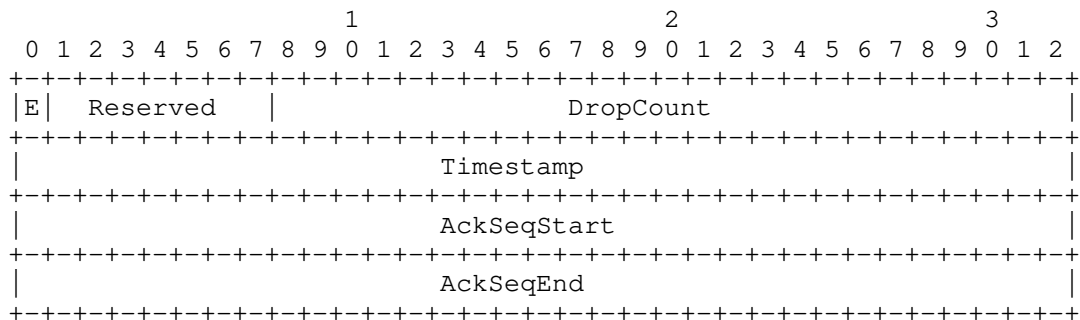
2 octet length set to 2.

SendInterval:

A 2 octet unsigned integer. The sending interval in milliseconds.

5.2.2. IKEv2 Congestion Information Notification Data

We utilize a send only (i.e., no response expected) IKEv2 INFORMATIONAL exchange (37) to transmit the congestion information using a notification payload of type TFS_CONGEST_INFO (TBD). The Response bit should be set to 0. As no response is expected the only payload should be the congestion information in the notification payload. The following diagram defines the notification payload data.



E:

A 1 bit value that if set indicates that packet[s] with Congestion Experienced (CE) ECN bits set were received and used in calculating the DropCount value.

Reserved:

A 7 bit field set to 0 ignored on receipt.

DropCount:

A 24 bit unsigned integer count of the drops that occurred between AckSeqStart and AckSeqEnd. If the drops exceed the resolution of the counter then set to the maximum value (i.e., 0xFFFFFFFF).

AckSeqStart:

A 32 bit unsigned integer containing the first ESP sequence number (as defined in [RFC4303]) of the packet range that this information relates to.

AckSeqEnd:

A 32 bit unsigned integer containing the last ESP sequence number (as defined in [RFC4303]) of the packet range that this information relates to.

Timestamp:

A 32 bit unsigned integer containing the lower 32 bits of a running monotonic millisecond timer of when this notification data

was created/sent. This value is used to determine duplicates and drop counts of this information. Implementations should deal with wrapping of this timer value.

6. IANA Considerations

This document requests a protocol number IPTFS_PROTOCOL be allocated by IANA from "Assigned Internet Protocol Numbers" registry for identifying the IP-TFS ESP payload format.

Type: TBD Description: IP-TFS ESP payload format. Reference: This document

Additionally this document requests an attribute value TFS_INFO_INTERVAL (TBD) be allocated by IANA from "IKEv2 Configuration Payload Attribute Types" registry.

Type: TBD Description: The sending rate of congestion information from egress tunnel endpoint. Reference: This document

Additionally this document requests a notify message status type TFS_CONGEST_INFO (TBD) be allocated by IANA from "IKEv2 Notify Message Types - Status Types" registry.

Type: TBD Description: The sending rate of congestion information from egress tunnel endpoint. Reference: This document

7. Security Considerations

This document describes a mechanism to add Traffic Flow Confidentiality to IP traffic. Use of this mechanism is expected to increase the security of the traffic being transported. Other than the additional security afforded by using this mechanism, IP-TFS utilizes the security protocols [RFC4303] and [RFC7296] and so their security considerations apply to IP-TFS as well.

As noted previously in Section 2.5.2, for TFC to be fully maintained the encapsulated traffic flow should not be affecting network congestion in a predictable way, and if it would be then non-congestion controlled mode use should be considered instead.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

- [AppCrypt] Schneier, B., "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 11 2017.
- [I-D.iab-wire-image] Trammell, B. and M. Kuehlewind, "The Wire Image of a Network Protocol", draft-iab-wire-image-01 (work in progress), November 2018.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.

- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [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>>.
- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <<https://www.rfc-editor.org/info/rfc7510>>.
- [RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <<https://www.rfc-editor.org/info/rfc8084>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

Appendix A. Comparisons of IP-TFS

A.1. Comparing Overhead

A.1.1. IP-TFS Overhead

The overhead of IP-TFS is 40 bytes per outer packet. Therefore the octet overhead per inner packet is 40 divided by the number of outer packets required (fractional allowed). The overhead as a percentage of inner packet size is a constant based on the Outer MTU size.

$$\begin{aligned} \text{OH} &= 40 / \text{Outer Payload Size} / \text{Inner Packet Size} \\ \text{OH \% of Inner Packet Size} &= 100 * \text{OH} / \text{Inner Packet Size} \\ \text{OH \% of Inner Packet Size} &= 4000 / \text{Outer Payload Size} \end{aligned}$$

Type	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000
PSize	536	1460	8960
<hr/>			
40	7.46%	2.74%	0.45%
576	7.46%	2.74%	0.45%
1500	7.46%	2.74%	0.45%
9000	7.46%	2.74%	0.45%

Figure 4: IP-TFS Overhead as Percentage of Inner Packet Size

A.1.2. ESP with Padding Overhead

The overhead per inner packet for constant-send-rate padded ESP (i.e., traditional IPSec TFC) is 36 octets plus any padding, unless fragmentation is required.

When fragmentation of the inner packet is required to fit in the outer IPsec packet, overhead is the number of outer packets required to carry the fragmented inner packet times both the inner IP overhead (20) and the outer packet overhead (36) minus the initial inner IP overhead plus any required tail padding in the last encapsulation packet. The required tail padding is the number of required packets times the difference of the Outer Payload Size and the IP Overhead minus the the Inner Payload Size. So:

Inner Payload Size = IP Packet Size - IP Overhead
 Outer Payload Size = MTU - IPSec Overhead

$$NF0 = \frac{\text{Inner Payload Size}}{\text{Outer Payload Size} - \text{IP Overhead}}$$

$$NF = \text{CEILING}(NF0)$$

$$\begin{aligned} OH &= NF * (\text{IP Overhead} + \text{IPsec Overhead}) \\ &\quad - \text{IP Overhead} \\ &\quad + NF * (\text{Outer Payload Size} - \text{IP Overhead}) \\ &\quad - \text{Inner Payload Size} \end{aligned}$$

$$\begin{aligned} OH &= NF * (\text{IPsec Overhead} + \text{Outer Payload Size}) \\ &\quad - (\text{IP Overhead} + \text{Inner Payload Size}) \end{aligned}$$

$$\begin{aligned} OH &= NF * (\text{IPsec Overhead} + \text{Outer Payload Size}) \\ &\quad - \text{Inner Packet Size} \end{aligned}$$

A.2. Overhead Comparison

The following tables collect the overhead values for some common L3 MTU sizes in order to compare them. The first table is the number of octets of overhead for a given L3 MTU sized packet. The second table is the percentage of overhead in the same MTU sized packet.

Type	ESP+Pad	ESP+Pad	ESP+Pad	IP-TFS	IP-TFS	IP-TFS
L3 MTU	576	1500	9000	576	1500	9000
PSize	540	1464	8964	536	1460	8960
<hr/>						
40	500	1424	8924	3.0	1.1	0.2
128	412	1336	8836	9.6	3.5	0.6
256	284	1208	8708	19.1	7.0	1.1
536	4	928	8428	40.0	14.7	2.4
576	576	888	8388	43.0	15.8	2.6
1460	268	4	7504	109.0	40.0	6.5
1500	228	1500	7464	111.9	41.1	6.7
8960	1408	1540	4	668.7	245.5	40.0
9000	1368	1500	9000	671.6	246.6	40.2

Figure 5: Overhead comparison in octets

Type	ESP+Pad	ESP+Pad	ESP+Pad	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000	576	1500	9000
PSize	540	1464	8964	536	1460	8960
<hr/>						
40	1250.0%	3560.0%	22310.0%	7.46%	2.74%	0.45%
128	321.9%	1043.8%	6903.1%	7.46%	2.74%	0.45%
256	110.9%	471.9%	3401.6%	7.46%	2.74%	0.45%
536	0.7%	173.1%	1572.4%	7.46%	2.74%	0.45%
576	100.0%	154.2%	1456.2%	7.46%	2.74%	0.45%
1460	18.4%	0.3%	514.0%	7.46%	2.74%	0.45%
1500	15.2%	100.0%	497.6%	7.46%	2.74%	0.45%
8960	15.7%	17.2%	0.0%	7.46%	2.74%	0.45%
9000	15.2%	16.7%	100.0%	7.46%	2.74%	0.45%

Figure 6: Overhead as Percentage of Inner Packet Size

A.3. Comparing Available Bandwidth

Another way to compare the two solutions is to look at the amount of available bandwidth each solution provides. The following sections consider and compare the percentage of available bandwidth. For the sake of providing a well understood baseline we will also include normal (unencrypted) Ethernet as well as normal ESP values.

A.3.1. Ethernet

In order to calculate the available bandwidth we first calculate the per packet overhead in bits. The total overhead of Ethernet is 14+4 octets of header and CRC plus and additional 20 octets of framing (preamble, start, and inter-packet gap) for a total of 48 octets. Additionally the minimum payload is 46 octets.

Size	E + P	E + P	E + P	IPTFS	IPTFS	IPTFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	74	74	74	78	78	78	38	74
<hr/>								
40	614	1538	9038	45	42	40	84	114
128	614	1538	9038	146	134	129	166	202
256	614	1538	9038	293	269	258	294	330
536	614	1538	9038	614	564	540	574	610
576	1228	1538	9038	659	606	581	614	650
1460	1842	1538	9038	1672	1538	1472	1498	1534
1500	1842	3076	9038	1718	1580	1513	1538	1574
8960	11052	10766	9038	10263	9438	9038	8998	9034
9000	11052	10766	18076	10309	9480	9078	9038	9074

Figure 7: L2 Octets Per Packet

Size	E + P	E + P	E + P	IPTFS	IPTFS	IPTFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	74	74	74	78	78	78	38	74
<hr/>								
40	2.0M	0.8M	0.1M	27.3M	29.7M	31.0M	14.9M	11.0M
128	2.0M	0.8M	0.1M	8.5M	9.3M	9.7M	7.5M	6.2M
256	2.0M	0.8M	0.1M	4.3M	4.6M	4.8M	4.3M	3.8M
536	2.0M	0.8M	0.1M	2.0M	2.2M	2.3M	2.2M	2.0M
576	1.0M	0.8M	0.1M	1.9M	2.1M	2.2M	2.0M	1.9M
1460	678K	812K	138K	747K	812K	848K	834K	814K
1500	678K	406K	138K	727K	791K	826K	812K	794K
8960	113K	116K	138K	121K	132K	138K	138K	138K
9000	113K	116K	69K	121K	131K	137K	138K	137K

Figure 8: Packets Per Second on 10G Ethernet

Size	E + P 590 74	E + P 1514 74	E + P 9014 74	IP-TFS 590 78	IP-TFS 1514 78	IP-TFS 9014 78	Enet any 38	ESP any 74
40	6.51%	2.60%	0.44%	87.30%	94.93%	99.14%	47.62%	35.09%
128	20.85%	8.32%	1.42%	87.30%	94.93%	99.14%	77.11%	63.37%
256	41.69%	16.64%	2.83%	87.30%	94.93%	99.14%	87.07%	77.58%
536	87.30%	34.85%	5.93%	87.30%	94.93%	99.14%	93.38%	87.87%
576	46.91%	37.45%	6.37%	87.30%	94.93%	99.14%	93.81%	88.62%
1460	79.26%	94.93%	16.15%	87.30%	94.93%	99.14%	97.46%	95.18%
1500	81.43%	48.76%	16.60%	87.30%	94.93%	99.14%	97.53%	95.30%
8960	81.07%	83.22%	99.14%	87.30%	94.93%	99.14%	99.58%	99.18%
9000	81.43%	83.60%	49.79%	87.30%	94.93%	99.14%	99.58%	99.18%

Figure 9: Percentage of Bandwidth on 10G Ethernet

A sometimes unexpected result of using IP-TFS (or any packet aggregating tunnel) is that, for small to medium sized packets, the available bandwidth is actually greater than native Ethernet. This is due to the reduction in Ethernet framing overhead. This increased bandwidth is paid for with an increase in latency. This latency is the time to send the unrelated octets in the outer tunnel frame. The following table illustrates the latency for some common values on a 10G Ethernet link. The table also includes latency introduced by padding if using ESP with padding.

	ESP+Pad 1500	ESP+Pad 9000	IP-TFS 1500	IP-TFS 9000
40	1.14 us	7.14 us	1.17 us	7.17 us
128	1.07 us	7.07 us	1.10 us	7.10 us
256	0.97 us	6.97 us	1.00 us	7.00 us
536	0.74 us	6.74 us	0.77 us	6.77 us
576	0.71 us	6.71 us	0.74 us	6.74 us
1460	0.00 us	6.00 us	0.04 us	6.04 us
1500	1.20 us	5.97 us	0.00 us	6.00 us

Figure 10: Added Latency

Notice that the latency values are very similar between the two solutions; however, whereas IP-TFS provides for constant high bandwidth, in some cases even exceeding native Ethernet, ESP with padding often greatly reduces available bandwidth.

Appendix B. Acknowledgements

We would like to thank Don Fedyk for help in reviewing this work.

Appendix C. Contributors

The following people made significant contributions to this document.

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Author's Address

Christian Hopps
LabN Consulting, L.L.C.

Email: chopps@chopps.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 7, 2019

C. Hopps
LabN Consulting, L.L.C.
June 5, 2019

IP Traffic Flow Security
draft-hopps-ipsecme-iptfs-01

Abstract

This document describes a mechanism to enhance IPsec traffic flow security by adding traffic flow confidentiality to encrypted IP encapsulated traffic. Traffic flow confidentiality is provided by obscuring the size and frequency of IP traffic using a fixed-sized, constant-send-rate IPsec tunnel. The solution allows for congestion control as well.

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 December 7, 2019.

Copyright Notice

Copyright (c) 2019 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.

Table of Contents

1. Introduction	3
1.1. Terminology & Concepts	3
2. The IP-TFS Tunnel	4
2.1. Tunnel Content	4
2.2. IPTFS_PROTOCOL Payload Content	4
2.2.1. Data Blocks	5
2.2.2. No Implicit Padding Required	6
2.2.3. Empty Payload	6
2.2.4. IP Header Value Mapping	6
2.3. Exclusive SA Use	7
2.4. Initiating IP-TFS Operation On The SA.	7
2.5. Modes of Operation	7
2.5.1. Non-Congestion Controlled Mode	7
2.5.2. Congestion Controlled Mode	8
3. Congestion Information	9
3.1. ECN Support	10
4. Configuration	10
4.1. Bandwidth	10
4.2. Fixed Packet Size	10
4.3. Congestion Control	11
5. IKEv2	11
5.1. TFS Type Transform Type	11
5.2. IPTFS_REQUIREMENTS Status Notification	11
6. Packet and Data Formats	12
6.1. ESP IP-TFS Payload	12
6.1.1. Non-Congestion Control IPTFS_PROTOCOL Payload Format	12
6.1.2. Congestion Control IPTFS_PROTOCOL Payload Format	13
6.1.3. Data Blocks	14
7. IANA Considerations	16
7.1. IPTFS_PROTOCOL Type	16
7.2. IKEv2 Transform Type TFS Type	16
7.3. TFS Type Transform IDs Registry	17
7.4. IPTFS_REQUIREMENTS Notify Message Status Type	17
8. Security Considerations	17
9. References	17
9.1. Normative References	17
9.2. Informative References	18
Appendix A. Example Of An Encapsulated IP Packet Flow	19
Appendix B. A Send and Loss Event Rate Calculation	20
Appendix C. Comparisons of IP-TFS	21
C.1. Comparing Overhead	21
C.1.1. IP-TFS Overhead	21
C.1.2. ESP with Padding Overhead	21

C.2. Overhead Comparison	22
C.3. Comparing Available Bandwidth	23
C.3.1. Ethernet	23
Appendix D. Acknowledgements	25
Appendix E. Contributors	25
Author's Address	25

1. Introduction

Traffic Analysis ([RFC4301], [AppCrypt]) is the act of extracting information about data being sent through a network. While one may directly obscure the data through the use of encryption [RFC4303], the traffic pattern itself exposes information due to variations in it's shape and timing ([I-D.iab-wire-image], [AppCrypt]). Hiding the size and frequency of traffic is referred to as Traffic Flow Confidentiality (TFC) per [RFC4303].

[RFC4303] provides for TFC by allowing padding to be added to encrypted IP packets and allowing for transmission of all-pad packets (indicated using protocol 59). This method has the major limitation that it can significantly under-utilize the available bandwidth.

The IP-TFS solution provides for full TFC without the aforementioned bandwidth limitation. To do this, we use a constant-send-rate IPsec [RFC4303] tunnel with fixed-sized encapsulating packets; however, these fixed-sized packets can contain partial, whole or multiple IP packets to maximize the bandwidth of the tunnel.

For a comparison of the overhead of IP-TFS with the RFC4303 prescribed TFC solution see Appendix C.

Additionally, IP-TFS provides for dealing with network congestion [RFC2914]. This is important for when the IP-TFS user is not in full control of the domain through which the IP-TFS tunnel path flows.

1.1. Terminology & Concepts

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

This document assumes familiarity with IP security concepts described in [RFC4301].

2. The IP-TFS Tunnel

As mentioned in Section 1 IP-TFS utilizes an IPsec [RFC4303] tunnel (SA) as it's transport. To provide for full TFC we send fixed-sized encapsulating packets at a constant rate on the tunnel.

The primary input to the tunnel algorithm is the requested bandwidth of the tunnel. Two values are then required to provide for this bandwidth, the fixed size of the encapsulating packets, and rate at which to send them.

The fixed packet size may either be specified manually or can be determined through the use of Path MTU discovery [RFC1191] and [RFC8201].

Given the encapsulating packet size and the requested tunnel bandwidth, the corresponding packet send rate can be calculated. The packet send rate is the requested bandwidth divided by the payload size of the encapsulating packet.

The egress of the IP-TFS tunnel MUST allow for, and expect the ingress (sending) side of the IP-TFS tunnel to vary the size and rate of sent encapsulating packets, unless constrained by other policy.

2.1. Tunnel Content

As previously mentioned, one issue with the TFC padding solution in [RFC4303] is the large amount of wasted bandwidth as only one IP packet can be sent per encapsulating packet. In order to maximize bandwidth IP-TFS breaks this one-to-one association.

With IP-TFS we aggregate as well as fragment the inner IP traffic flow into fixed-sized encapsulating IPsec tunnel packets. We only pad the tunnel packets if there is no data available to be sent at the time of tunnel packet transmission, or if fragmentation has been disabled by the receiver.

In order to do this we use a new Encapsulating Security Payload (ESP, [RFC4303]) payload type which is the new IP protocol number IPTFS_PROTOCOL (TBD1).

2.2. IPTFS_PROTOCOL Payload Content

The IPTFS_PROTOCOL ESP payload is comprised a 4 or 16 octet header followed by either a partial, a full or multiple partial or full data blocks. The following diagram illustrates the IPTFS_PROTOCOL ESP payload within the ESP packet. See Section 6.1 for the exact formats of the IPTFS_PROTOCOL payload.

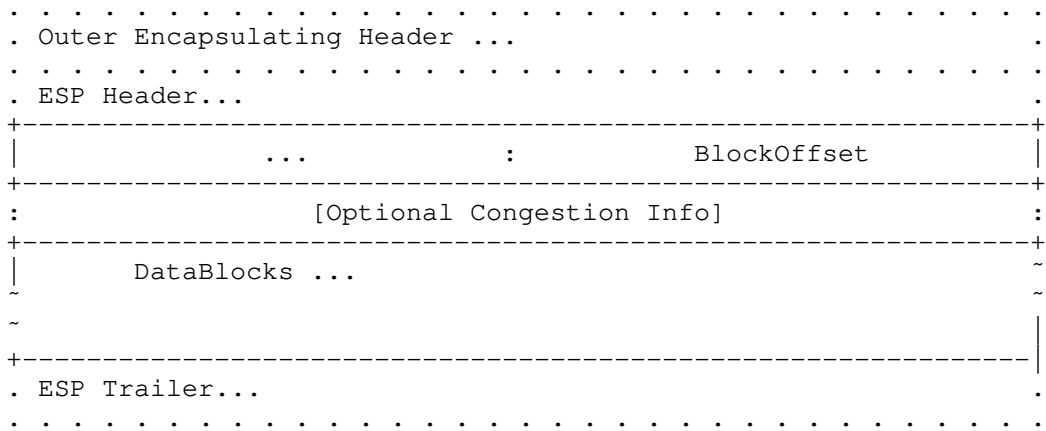


Figure 1: Layout of an IP-TFS IPsec Packet

The "BlockOffset" value is either zero or some offset into or past the end of the "DataBlocks" data.

If the "BlockOffset" value is zero it means that the "DataBlocks" data begins with a new data block.

Conversely, if the "BlockOffset" value is non-zero it points to the start of the new data block, and the initial "DataBlocks" data belongs to a previous data block that is still being re-assembled.

The "BlockOffset" can point past the end of the "DataBlocks" data which indicates that the next data block occurs in a subsequent encapsulating packet.

Having the "BlockOffset" always point at the next available data block allows for quick recovery with minimal inner packet loss in the presence of outer encapsulating packet loss.

An example IP-TFS packet flow can be found in Appendix A.

2.2.1. Data Blocks

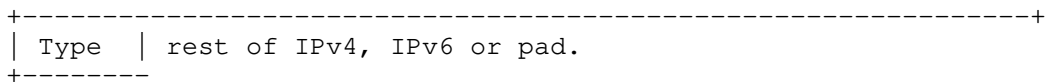


Figure 2: Layout of IP-TFS data block

A data block is defined by a 4-bit type code followed by the data block data. The type values have been carefully chosen to coincide

with the IPv4/IPv6 version field values so that no per-data block type overhead is required to encapsulate an IP packet. Likewise, the length of the data block is extracted from the encapsulated IPv4 or IPv6 packet's length field.

2.2.2. No Implicit Padding Required

It's worth noting that there is never a need for an implicit pad at the end of an encapsulating packet. Even when the start of a data block occurs near the end of a encapsulating packet such that there is no room for the length field of the encapsulated header to be included in the current encapsulating packet, the fact that the length comes at a known location and is guaranteed to be present is enough to fetch the length field from the subsequent encapsulating packet payload. Only when there is no data to encapsulate is padding required, and then an explicit "Pad Data Block" would be used to identify the padding.

2.2.3. Empty Payload

In order to support reporting of congestion control information (described later) on a non-IP-TFS enabled SA, IP-TFS allows for the sending of an IP-TFS payload with no data blocks (i.e., the ESP payload length is equal to the IP-TFS header length). This special payload is called an empty payload.

2.2.4. IP Header Value Mapping

[RFC4301] provides some direction on when and how to map various values from an inner IP header to the outer encapsulating header, namely the Don't-Fragment (DF) bit ([RFC0791] and [RFC8200]), the Differentiated Services (DS) field [RFC2474] and the Explicit Congestion Notification (ECN) field [RFC3168]. Unlike [RFC4301] with IP-TFS we may and often will be encapsulating more than 1 IP packet per ESP packet. To deal with this we further restrict these mappings. In particular we never map the inner DF bit as it is unrelated to the IP-TFS tunnel functionality; we never IP fragment the inner packets and the inner packets will not affect the fragmentation of the outer encapsulation packets. Likewise, the ECN value need not be mapped as any congestion related to the constant-send-rate IP-TFS tunnel is unrelated (by design!) to the inner traffic flow. Finally, by default the DS field SHOULD NOT be copied although an implementation MAY choose to allow for configuration to override this behavior. An implementation SHOULD also allow the DS value to be set by configuration.

2.3. Exclusive SA Use

It is not the intention of this specification to allow for mixed use of an IP-TFS enabled SA. In other words, an SA that has IP-TFS enabled is exclusively for IP-TFS use and MUST NOT have non-IP-TFS payloads such as IP (IP protocol 4), TCP transport (IP protocol 6), or ESP pad packets (protocol 59) intermixed with non-empty IP-TFS (IP protocol TBD1) payloads. While it's possible to envision making the algorithm work in the presence of sequence number skips in the IP-TFS payload stream, the added complexity is not deemed worthwhile. Other IPsec uses can configure and use their own SAs.

2.4. Initiating IP-TFS Operation On The SA.

While a user will normally configure their IPsec tunnel (SA) to operate using IP-TFS to start, we also allow IP-TFS operation to be enabled post-SA creation and use. This late-enabling may be useful for debugging or other purposes. To support this late-enabled operation the receiver switches to IP-TFS operation on receipt of the first ESP payload with the IPTFS_PROTOCOL indicated as the payload type which also contains a data block (i.e., a non-empty IP-TFS payload). The receipt of an empty IPTFS_PROTOCOL payload (i.e., one without any data blocks) is used to communicate congestion control information from the receiver back to the sender on a non-IP-TFS enabled SA, and MUST NOT cause IP-TFS to be enabled on that SA.

2.5. Modes of Operation

Just as with normal IPsec/ESP tunnels, IP-TFS tunnels are unidirectional. Bidirectional IP-TFS functionality is achieved by setting up 2 IP-TFS tunnels, one in either direction.

An IP-TFS tunnel can operate in 2 modes, a non-congestion controlled mode and congestion controlled mode.

2.5.1. Non-Congestion Controlled Mode

In the non-congestion controlled mode IP-TFS sends fixed-sized packets at a constant rate. The packet send rate is constant and is not automatically adjusted regardless of any network congestion (e.g., packet loss).

For similar reasons as given in [RFC7510] the non-congestion controlled mode should only be used where the user has full administrative control over the path the tunnel will take. This is required so the user can guarantee the bandwidth and also be sure as to not be negatively affecting network congestion [RFC2914]. In this case packet loss should be reported to the administrator (e.g., via

syslog, YANG notification, SNMP traps, etc) so that any failures due to a lack of bandwidth can be corrected.

2.5.2. Congestion Controlled Mode

With the congestion controlled mode, IP-TFS adapts to network congestion by lowering the packet send rate to accommodate the congestion, as well as raising the rate when congestion subsides. Since overhead is per packet, by allowing for maximal fixed-size packets and varying the send rate we minimize transport overhead.

The output of the congestion control algorithm will adjust the rate at which the ingress sends packets. While this document does not require a specific congestion control algorithm, best current practice RECOMMENDS that the algorithm conform to [RFC5348]. Congestion control principles are documented in [RFC2914] as well. An example of an implementation of the [RFC5348] algorithm which matches the requirements of IP-TFS (i.e., designed for fixed-size packet and send rate varied based on congestion) is documented in [RFC4342].

The required inputs for the TCP friendly rate control algorithm described in [RFC5348] are the receivers loss event rate and the senders estimated round-trip time (RTT). These values are provided by IP-TFS using the congestion information header fields described in Section 3. In particular these values are sufficient to implement the algorithm described in [RFC5348].

At a minimum, the congestion information must be sent, from the receiver as well as from the sender, at least once per RTT. Prior to establishing an RTT the information SHOULD be sent constantly from the sender and the receiver so that an RTT estimate can be established. The lack of receiving this information over multiple consecutive RTT intervals should be considered a congestion event that causes the sender to adjust it's sending rate lower. For example, [RFC4342] calls this the "no feedback timeout" and it is equal to 4 RTT intervals. When a "no feedback timeout" has occurred [RFC4342] halves the sending rate.

An implementation could choose to always include the congestion information in it's IP-TFS payload header if sending on an IP-TFS enabled SA. Since IP-TFS normally will operate with a large packet size, the congestion information should represent a small portion of the available tunnel bandwidth.

When an implementation is choosing a congestion control algorithm (or a selection of algorithms) one should remember that IP-TFS is not

providing for reliable delivery of IP traffic, and so per packet ACKs are not required and are not provided.

It's worth noting that the variable send-rate of a congestion controlled IP-TFS tunnel, is not private; however, this send-rate is being driven by network congestion, and as long as the encapsulated (inner) traffic flow shape and timing are not directly affecting the (outer) network congestion, the variations in the tunnel rate will not weaken the provided inner traffic flow confidentiality.

2.5.2.1. Circuit Breakers

In addition to congestion control, implementations MAY choose to define and implement circuit breakers [RFC8084] as a recovery method of last resort. Enabling circuit breakers is also a reason a user may wish to enable congestion information reports even when using the non-congestion controlled mode of operation. The definition of circuit breakers are outside the scope of this document.

3. Congestion Information

In order to support the congestion control mode, the sender needs to know the loss event rate and also be able to approximate the RTT ([RFC5348]). In order to obtain these values the receiver sends congestion control information on it's SA back to the sender. Thus, in order to support congestion control the receiver must have a paired SA back to the sender (this is always the case when the tunnel was created using IKEv2). If the SA back to the sender is a non-IP-TFS enabled SA then an IPTFS_PROTOCOL empty payload (i.e., header only) is used to convey the information.

In order to calculate a loss event rate compatible with [RFC5348], the receiver needs to have a round-trip time estimate. Thus the sender communicates this estimate in the "RTT" header field. On startup this value will be zero as no RTT estimate is yet known.

In order to allow the sender to calculate the "RTT" value, the receiver communicates the last sequence number it has seen to the sender in the "LastSeqNum" header field. In addition to the "LastSeqNum" value, the receiver sends an estimate of the amount of time between receiving the "LastSeqNum" packet and transmitting the "LastSeqNum" value back to the sender in the congestion information. It places this time estimate in the "Delay" header field along with the "LastSeqNum".

The receiver also calculates, and communicates in the "LossEventRate" header field, the loss event rate for use by the sender. This is slightly different from [RFC4342] which periodically sends all the

loss interval data back to the sender so that it can do the calculation. See Appendix B for a suggested way to calculate the loss event rate value. Initially this value will be zero (indicating no loss) until enough data has been collected by the receiver to update it.

3.1. ECN Support

In addition to normal packet loss information IP-TFS supports use of the ECN bits in the encapsulating IP header [RFC3168] for identifying congestion. If ECN use is enabled and a packet arrives at the egress endpoint with the Congestion Experienced (CE) value set, then the receiver considers that packet as being dropped, although it does not drop it. The receiver MUST set the E bit in any IPTFS_PROTOCOL payload header containing a "LossEventRate" value derived from a CE value being considered.

As noted in [RFC3168] the ECN bits are not protected by IPsec and thus may constitute a covert channel. For this reason ECN use SHOULD NOT be enabled by default.

4. Configuration

IP-TFS is meant to be deployable with a minimal amount of configuration. All IP-TFS specific configuration should be able to be specified at the unidirectional tunnel ingress (sending) side. It is intended that non-IKEv2 operation is supported, at least, with local static configuration.

4.1. Bandwidth

Bandwidth is a local configuration option. For non-congestion controlled mode the bandwidth SHOULD be configured. For congestion controlled mode one can configure the bandwidth or have no configuration and let congestion control discover the maximum bandwidth available. No standardized configuration method is required.

4.2. Fixed Packet Size

The fixed packet size to be used for the tunnel encapsulation packets can be configured manually or can be automatically determined using Path MTU discovery (see [RFC1191] and [RFC8201]). No standardized configuration method is required.

4.3. Congestion Control

Congestion control is a local configuration option. No standardized configuration method is required.

5. IKEv2

5.1. TFS Type Transform Type

When IP-TFS is used with IKEv2 a new "TFS Type" Transform Type (TBD2) is used to negotiate (as defined in [RFC7296]) the possible operation of IP-TFS on a child SA pair. This document defines 3 "TFS Type" Transform IDs for the new "TFS Type" Transform Type: None (0), TFS_IPTFS_CC (1) for congestion-controlled IP-TFS mode or TFS_IPTFS_NOCC (2) for non-congestion controlled IP-TFS mode. The selection of a proposal with a "TFS Type" Transform ID TFS_IPTFS_CC or TFS_IPTFS_NOCC does not mandate the use of IP-TFS, rather it indicates a willingness or intent to use IP-TFS on the SA pair. In addition, a new Notify Message Status Type IPTFS_REQUIREMENTS (TBD3) MAY be used by the initiator as well as the responder to further refine any operational requirements.

Additional "TFS Type" Transform IDs may be defined in the future, and so readers are referred to [IKEV2IANA] for the most up to date list.

5.2. IPTFS_REQUIREMENTS Status Notification

As mentioned in the previous section, a new Notify Message Status Type IPTFS_REQUIREMENTS (TBD3) MAY be sent by the initiator and/or the responder to further refine what will be supported. This notification is sent during IKE_AUTH and new CREATE_CHILD_SA exchanges; however, it MUST NOT be sent, and MUST be ignored, during a CREATE_CHILD_SA rekeying exchange as the requirements are not allowed to change during rekeying.

The IPTFS_REQUIREMENTS notification contains a 1 octet payload of flags that specify any extra requirements from the sender of the message. The flag values (currently a single flag) are defined below. If the IPTFS_REQUIREMENTS notification is not sent then it implies that all the flag bits are clear.

```

+-----+
|0|0|0|0|0|0|0|D|
+-----+

```

0:

MUST be zero on send and MUST be ignored on receive.

D:

Don't Fragment bit, if set indicates the sender of the notify message does not support receiving packet fragments (i.e., inner packets MUST be sent using a single "Data Block"). This value only applies to what the sender is capable of receiving; the sender MAY still send packet fragments unless similarly restricted by the receiver in it's IPTFS_REQUIREMENTS notification.

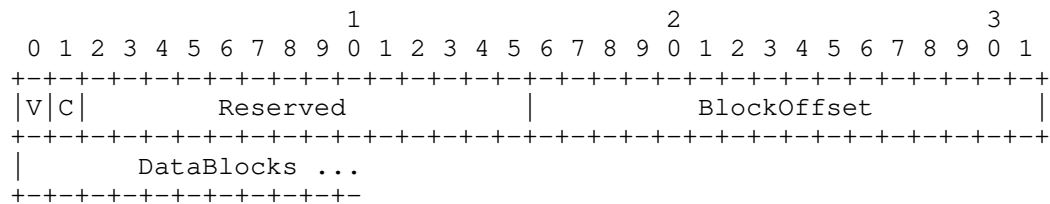
6. Packet and Data Formats

6.1. ESP IP-TFS Payload

An ESP IP-TFS payload is identified by the IP protocol number IPTFS_PROTOCOL (TBD1). This payload begins with a fixed 4 or 16 octet header followed by a variable amount of "DataBlocks" data. The exact payload format and fields are defined in the following sections.

6.1.1. Non-Congestion Control IPTFS_PROTOCOL Payload Format

The non-congestion control IPTFS_PROTOCOL payload is comprised of a 4 octet header followed by a variable amount of "DataBlocks" data as shown below.



V:

A 1 bit version field that MUST be set to zero. If received as one the packet MUST be dropped.

C:

A 1 bit value that MUST be set to 0 to indicate no congestion control information is present.

Reserved:

A 14 bit field set to 0 and ignored on receipt.

BlockOffset:

A 16 bit unsigned integer counting the number of octets of "DataBlocks" data before the start of a new data block. "BlockOffset" can count past the end of the "DataBlocks" data in which case all the "DataBlocks" data belongs to the previous data

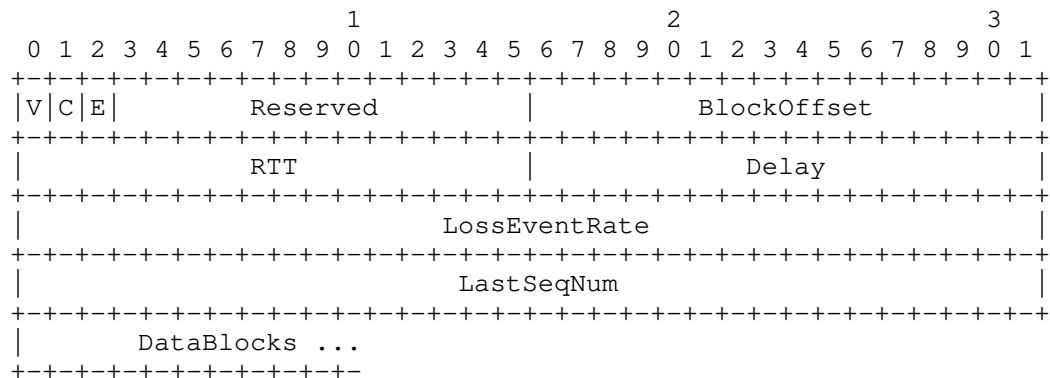
block being re-assembled. If the "BlockOffset" extends into subsequent packets it continues to only count subsequent "DataBlocks" data (i.e., it does not count subsequent packets non-"DataBlocks" octets).

DataBlocks:

Variable number of octets that begins with the start of a data block, or the continuation of a previous data block, followed by zero or more additional data blocks.

6.1.2. Congestion Control IPTFS_PROTOCOL Payload Format

The congestion control IPTFS_PROTOCOL payload is comprised of a 16 octet header followed by a variable amount of "DataBlocks" data as shown below.



V:

A 1 bit version field that MUST be set to zero. If received as one the packet MUST be dropped.

C:

A 1 bit value that MUST be set to 1 which indicates the presence of the congestion information header fields "RTT", "Delay", "LossEventRate" and "LastSeqNum".

E:

A 1 bit value if set indicates that Congestion Experienced (CE) ECN bits were received and used in deriving the reported "LossEventRate".

Reserved:

A 13 bit field set to 0 and ignored on receipt.

BlockOffset:

The same value as the non-congestion controlled payload format value.

RTT:

A 16 bit value specifying the sender's current round-trip time estimate in milliseconds. The value MAY be zero prior to the sender having calculated a round-trip time estimate. The value SHOULD be set to zero on non-IP-TFS enabled SAs.

Delay:

A 16 bit value specifying the delay in milliseconds incurred between the receiver receiving the "LastSeqNum" packet and the sending of this acknowledgement of it.

LossEventRate:

A 32 bit value specifying the inverse of the current loss event rate as calculated by the receiver. A value of zero indicates no loss. Otherwise the loss event rate is "1/LossEventRate".

LastSeqNum:

A 32 bit value containing the lower 32 bits of the largest sequence number last received. This is the latest in the sequence not necessarily the most recent (in the case of re-ordering of packets it may be less recent). When determining largest and 64 bit extended sequence numbers are in use, the upper 32 bits should be used during the comparison.

DataBlocks:

Variable number of octets that begins with the start of a data block, or the continuation of a previous data block, followed by zero or more additional data blocks. For the special case of sending congestion control information on an non-IP-TFS enabled SA this value MUST be empty (i.e., be zero octets long).

6.1.3. Data Blocks

```

          1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Type | IPv4, IPv6 or pad...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Type:

A 4 bit field where 0x0 identifies a pad data block, 0x4 indicates an IPv4 data block, and 0x6 indicates an IPv6 data block.

6.1.3.1. IPv4 Data Block

```

      1                               2                               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| 0x4 | IHL | TypeOfService | TotalLength |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Rest of the inner packet ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

These values are the actual values within the encapsulated IPv4 header. In other words, the start of this data block is the start of the encapsulated IP packet.

Type:

A 4 bit value of 0x4 indicating IPv4 (i.e., first nibble of the IPv4 packet).

TotalLength:

The 16 bit unsigned integer length field of the IPv4 inner packet.

6.1.3.2. IPv6 Data Block

										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
0x6										TrafficClass										FlowLabel																			
										TotalLength										Rest of the inner packet ...																			

These values are the actual values within the encapsulated IPv6 header. In other words, the start of this data block is the start of the encapsulated IP packet.

Type:

A 4 bit value of 0x6 indicating IPv6 (i.e., first nibble of the IPv6 packet).

TotalLength:

The 16 bit unsigned integer length field of the inner IPv6 inner packet.

6.1.3.3. Pad Data Block

```

          1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|  0x0  | Padding ...
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

Type:

A 4 bit value of 0x0 indicating a padding data block.

Padding:

extends to end of the encapsulating packet.

7. IANA Considerations**7.1. IPTFS_PROTOCOL Type**

This document requests a protocol number IPTFS_PROTOCOL be allocated by IANA from "Assigned Internet Protocol Numbers" registry for identifying the IP-TFS ESP payload format.

Type:

TBD1

Description:

IP-TFS ESP payload format.

Reference:

This document

7.2. IKEv2 Transform Type TFS Type

This document requests an IKEv2 Transform Type "TFS Type" be allocated by IANA from the "Transform Type Values" registry.

Type:

TBD2

Description:

TFS Type

Used In:

(optional in ESP)

Reference:

This document

7.3. TFS Type Transform IDs Registry

This document requests a "Transform Type TBD3 - TFS Type Transform IDs" registry be created. The registration procedure is Expert Review. The initial values are as follows:

Number	Name	Reference
0	NONE	This document
1	TFS_IPTFS_CC	This document
2	TFS_IPTFS_NOCC	This document
3-65535	Reserved	This document

7.4. IPTFS_REQUIREMENTS Notify Message Status Type

This document requests a status type IPTFS_REQUIREMENTS be allocated from the "IKEv2 Notify Message Types - Status Types" registry.

Value:

TBD3

Name:

IPTFS_REQUIREMENTS

Reference:

This document

8. Security Considerations

This document describes a mechanism to add Traffic Flow Confidentiality to IP traffic. Use of this mechanism is expected to increase the security of the traffic being transported. Other than the additional security afforded by using this mechanism, IP-TFS utilizes the security protocols [RFC4303] and [RFC7296] and so their security considerations apply to IP-TFS as well.

As noted previously in Section 2.5.2, for TFC to be fully maintained the encapsulated traffic flow should not be affecting network congestion in a predictable way, and if it would be then non-congestion controlled mode use should be considered instead.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [AppCrypt] Schneier, B., "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 11 2017.
- [I-D.iab-wire-image] Trammell, B. and M. Kuehlewind, "The Wire Image of a Network Protocol", draft-iab-wire-image-01 (work in progress), November 2018.
- [IKEV2IANA] IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <<http://www.iana.org/assignments/ikev2-parameters/>>.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.

- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [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>>.
- [RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, DOI 10.17487/RFC4342, March 2006, <<https://www.rfc-editor.org/info/rfc4342>>.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <<https://www.rfc-editor.org/info/rfc5348>>.
- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <<https://www.rfc-editor.org/info/rfc7510>>.
- [RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <<https://www.rfc-editor.org/info/rfc8084>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

Appendix A. Example Of An Encapsulated IP Packet Flow

Below we show an example inner IP packet flow within the encapsulating tunnel packet stream. Notice how encapsulated IP

packets can start and end anywhere, and more than one or less than 1 may occur in a single encapsulating packet.

```

Offset: 0           Offset: 100       Offset: 2900       Offset: 1400
[ ESP1  (1500) ][ ESP2  (1500) ][ ESP3  (1500) ][ ESP4  (1500) ]
[--800--][--800--][60][-240-][--4000-----][pad]

```

Figure 3: Inner and Outer Packet Flow

The encapsulated IP packet flow (lengths include IP header and payload) is as follows: an 800 octet packet, an 800 octet packet, a 60 octet packet, a 240 octet packet, a 4000 octet packet.

The "BlockOffset" values in the 4 IP-TFS payload headers for this packet flow would thus be: 0, 100, 2900, 1400 respectively. The first encapsulating packet ESP1 has a zero "BlockOffset" which points at the IP data block immediately following the IP-TFS header. The following packet ESP2s "BlockOffset" points inward 100 octets to the start of the 60 octet data block. The third encapsulating packet ESP3 contains the middle portion of the 4000 octet data block so the offset points past its end and into the forth encapsulating packet. The fourth packet ESP4s offset is 1400 pointing at the padding which follows the completion of the continued 4000 octet packet.

Appendix B. A Send and Loss Event Rate Calculation

The current best practice indicates that congestion control should be done in a TCP friendly way. A TCP friendly congestion control algorithm is described in [RFC5348]. For our use case (as with [RFC4342]) we consider our (fixed) packet size the segment size for the algorithm. The formula for the send rate is then as follows:

$$X_Pps = \frac{1}{R * (\sqrt{2*p/3}) + 12*\sqrt{3*p/8}*p*(1+32*p^2)}$$

Where "X_Pps" is the send rate in packets per second, "R" is the round trip time estimate and "p" is the loss event rate (the inverse of which is provided by the receiver).

The IP-TFS receiver, having the RTT estimate from the sender MAY use the same method as described in [RFC4342] to collect the loss intervals and calculate the loss event rate value using the weighted average as indicated. The receiver communicates the inverse of this value back to the sender in the IPTFS_PROTOCOL payload header field "LossEventRate".

The IP-TFS sender now has both the "R" and "p" values and can calculate the correct sending rate ("X_Pps"). If following [RFC5348] the sender SHOULD also use the slow start mechanism described therein when the IP-TFS SA is first established.

Appendix C. Comparisons of IP-TFS

C.1. Comparing Overhead

C.1.1. IP-TFS Overhead

The overhead of IP-TFS is 40 bytes per outer packet. Therefore the octet overhead per inner packet is 40 divided by the number of outer packets required (fractional allowed). The overhead as a percentage of inner packet size is a constant based on the Outer MTU size.

OH = 40 / Outer Payload Size / Inner Packet Size
 OH % of Inner Packet Size = 100 * OH / Inner Packet Size
 OH % of Inner Packet Size = 4000 / Outer Payload Size

Type	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000
PSize	536	1460	8960

40	7.46%	2.74%	0.45%
576	7.46%	2.74%	0.45%
1500	7.46%	2.74%	0.45%
9000	7.46%	2.74%	0.45%

Figure 4: IP-TFS Overhead as Percentage of Inner Packet Size

C.1.2. ESP with Padding Overhead

The overhead per inner packet for constant-send-rate padded ESP (i.e., traditional IPsec TFC) is 36 octets plus any padding, unless fragmentation is required.

When fragmentation of the inner packet is required to fit in the outer IPsec packet, overhead is the number of outer packets required to carry the fragmented inner packet times both the inner IP overhead (20) and the outer packet overhead (36) minus the initial inner IP overhead plus any required tail padding in the last encapsulation packet. The required tail padding is the number of required packets times the difference of the Outer Payload Size and the IP Overhead minus the Inner Payload Size. So:

Inner Payload Size = IP Packet Size - IP Overhead
 Outer Payload Size = MTU - IPsec Overhead

$$NF0 = \frac{\text{Inner Payload Size}}{\text{Outer Payload Size} - \text{IP Overhead}}$$

NF = CEILING(NF0)

$$OH = NF * (\text{IP Overhead} + \text{IPsec Overhead}) \\ - \text{IP Overhead} \\ + NF * (\text{Outer Payload Size} - \text{IP Overhead}) \\ - \text{Inner Payload Size}$$

$$OH = NF * (\text{IPsec Overhead} + \text{Outer Payload Size}) \\ - (\text{IP Overhead} + \text{Inner Payload Size})$$

$$OH = NF * (\text{IPsec Overhead} + \text{Outer Payload Size}) \\ - \text{Inner Packet Size}$$

C.2. Overhead Comparison

The following tables collect the overhead values for some common L3 MTU sizes in order to compare them. The first table is the number of octets of overhead for a given L3 MTU sized packet. The second table is the percentage of overhead in the same MTU sized packet.

Type	ESP+Pad	ESP+Pad	ESP+Pad	IP-TFS	IP-TFS	IP-TFS
L3 MTU	576	1500	9000	576	1500	9000
PSize	540	1464	8964	536	1460	8960
<hr/>						
40	500	1424	8924	3.0	1.1	0.2
128	412	1336	8836	9.6	3.5	0.6
256	284	1208	8708	19.1	7.0	1.1
536	4	928	8428	40.0	14.7	2.4
576	576	888	8388	43.0	15.8	2.6
1460	268	4	7504	109.0	40.0	6.5
1500	228	1500	7464	111.9	41.1	6.7
8960	1408	1540	4	668.7	245.5	40.0
9000	1368	1500	9000	671.6	246.6	40.2

Figure 5: Overhead comparison in octets

Type	ESP+Pad	ESP+Pad	ESP+Pad	IP-TFS	IP-TFS	IP-TFS
MTU	576	1500	9000	576	1500	9000
PSize	540	1464	8964	536	1460	8960
<hr/>						
40	1250.0%	3560.0%	22310.0%	7.46%	2.74%	0.45%
128	321.9%	1043.8%	6903.1%	7.46%	2.74%	0.45%
256	110.9%	471.9%	3401.6%	7.46%	2.74%	0.45%
536	0.7%	173.1%	1572.4%	7.46%	2.74%	0.45%
576	100.0%	154.2%	1456.2%	7.46%	2.74%	0.45%
1460	18.4%	0.3%	514.0%	7.46%	2.74%	0.45%
1500	15.2%	100.0%	497.6%	7.46%	2.74%	0.45%
8960	15.7%	17.2%	0.0%	7.46%	2.74%	0.45%
9000	15.2%	16.7%	100.0%	7.46%	2.74%	0.45%

Figure 6: Overhead as Percentage of Inner Packet Size

C.3. Comparing Available Bandwidth

Another way to compare the two solutions is to look at the amount of available bandwidth each solution provides. The following sections consider and compare the percentage of available bandwidth. For the sake of providing a well understood baseline we will also include normal (unencrypted) Ethernet as well as normal ESP values.

C.3.1. Ethernet

In order to calculate the available bandwidth we first calculate the per packet overhead in bits. The total overhead of Ethernet is 14+4 octets of header and CRC plus and additional 20 octets of framing (preamble, start, and inter-packet gap) for a total of 48 octets. Additionally the minimum payload is 46 octets.

Size	E + P	E + P	E + P	IP-TFS	IP-TFS	IP-TFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	74	74	74	78	78	78	38	74
<hr/>								
40	614	1538	9038	45	42	40	84	114
128	614	1538	9038	146	134	129	166	202
256	614	1538	9038	293	269	258	294	330
536	614	1538	9038	614	564	540	574	610
576	1228	1538	9038	659	606	581	614	650
1460	1842	1538	9038	1672	1538	1472	1498	1534
1500	1842	3076	9038	1718	1580	1513	1538	1574
8960	11052	10766	9038	10263	9438	9038	8998	9034
9000	11052	10766	18076	10309	9480	9078	9038	9074

Figure 7: L2 Octets Per Packet

Size	E + P	E + P	E + P	IPTFS	IPTFS	IPTFS	Enet	ESP
MTU	590	1514	9014	590	1514	9014	any	any
OH	74	74	74	78	78	78	38	74
40	2.0M	0.8M	0.1M	27.3M	29.7M	31.0M	14.9M	11.0M
128	2.0M	0.8M	0.1M	8.5M	9.3M	9.7M	7.5M	6.2M
256	2.0M	0.8M	0.1M	4.3M	4.6M	4.8M	4.3M	3.8M
536	2.0M	0.8M	0.1M	2.0M	2.2M	2.3M	2.2M	2.0M
576	1.0M	0.8M	0.1M	1.9M	2.1M	2.2M	2.0M	1.9M
1460	678K	812K	138K	747K	812K	848K	834K	814K
1500	678K	406K	138K	727K	791K	826K	812K	794K
8960	113K	116K	138K	121K	132K	138K	138K	138K
9000	113K	116K	69K	121K	131K	137K	138K	137K

Figure 8: Packets Per Second on 10G Ethernet

Size	E + P	E + P	E + P	IPTFS	IPTFS	IPTFS	Enet	ESP
	590	1514	9014	590	1514	9014	any	any
	74	74	74	78	78	78	38	74
40	6.51%	2.60%	0.44%	87.30%	94.93%	99.14%	47.62%	35.09%
128	20.85%	8.32%	1.42%	87.30%	94.93%	99.14%	77.11%	63.37%
256	41.69%	16.64%	2.83%	87.30%	94.93%	99.14%	87.07%	77.58%
536	87.30%	34.85%	5.93%	87.30%	94.93%	99.14%	93.38%	87.87%
576	46.91%	37.45%	6.37%	87.30%	94.93%	99.14%	93.81%	88.62%
1460	79.26%	94.93%	16.15%	87.30%	94.93%	99.14%	97.46%	95.18%
1500	81.43%	48.76%	16.60%	87.30%	94.93%	99.14%	97.53%	95.30%
8960	81.07%	83.22%	99.14%	87.30%	94.93%	99.14%	99.58%	99.18%
9000	81.43%	83.60%	49.79%	87.30%	94.93%	99.14%	99.58%	99.18%

Figure 9: Percentage of Bandwidth on 10G Ethernet

A sometimes unexpected result of using IP-TFS (or any packet aggregating tunnel) is that, for small to medium sized packets, the available bandwidth is actually greater than native Ethernet. This is due to the reduction in Ethernet framing overhead. This increased bandwidth is paid for with an increase in latency. This latency is the time to send the unrelated octets in the outer tunnel frame. The following table illustrates the latency for some common values on a 10G Ethernet link. The table also includes latency introduced by padding if using ESP with padding.

	ESP+Pad 1500	ESP+Pad 9000	IP-TFS 1500	IP-TFS 9000

40	1.14 us	7.14 us	1.17 us	7.17 us
128	1.07 us	7.07 us	1.10 us	7.10 us
256	0.97 us	6.97 us	1.00 us	7.00 us
536	0.74 us	6.74 us	0.77 us	6.77 us
576	0.71 us	6.71 us	0.74 us	6.74 us
1460	0.00 us	6.00 us	0.04 us	6.04 us
1500	1.20 us	5.97 us	0.00 us	6.00 us

Figure 10: Added Latency

Notice that the latency values are very similar between the two solutions; however, whereas IP-TFS provides for constant high bandwidth, in some cases even exceeding native Ethernet, ESP with padding often greatly reduces available bandwidth.

Appendix D. Acknowledgements

We would like to thank Don Fedyk for help in reviewing this work.

Appendix E. Contributors

The following people made significant contributions to this document.

Lou Berger
LabN Consulting, L.L.C.

Email: lberger@labn.net

Author's Address

Christian Hopps
LabN Consulting, L.L.C.

Email: chopps@chopps.org

IPSECME
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2020

D. Migault
Ericsson
T. Guggemos
LMU Munich
Y. Nir
Dell EMC
October 21, 2019

Implicit IV for Counter-based Ciphers in Encapsulating Security Payload
(ESP)
draft-ietf-ipsecme-implicit-iv-11

Abstract

Encapsulating Security Payload (ESP) sends an initialization vector (IV) in each packet. The size of IV depends on the applied transform, being usually 8 or 16 octets for the transforms defined by the time this document is written. Some algorithms such as AES-GCM, AES-CCM and ChaCha20-Poly1305 when used with IPsec, take the IV to generate a nonce that is used as an input parameter for encrypting and decrypting. This IV must be unique but can be predictable. As a result, the value provided in the ESP Sequence Number (SN) can be used instead to generate the nonce. This avoids sending the IV itself, and saves in the case of AES-GCM, AES-CCM and ChaCha20-Poly1305 8 octets per packet. This document describes how to do this.

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 23, 2020.

Copyright Notice

Copyright (c) 2019 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.

Table of Contents

1. Requirements notation	2
2. Introduction	2
3. Terminology	3
4. Implicit IV	3
5. IKEv2 Initiator Behavior	4
6. IKEv2 Responder Behavior	5
7. Security Considerations	5
8. IANA Considerations	5
9. Acknowledgements	6
10. References	6
10.1. Normative References	6
10.2. Informational References	8
Authors' Addresses	8

1. Requirements notation

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

2. Introduction

Counter-based AES modes of operation such as AES-CCM ([RFC4309]), and AES-GCM ([RFC4106]) require the specification of a nonce for each ESP packet. The same applies for ChaCha20-Poly1305 ([RFC7634]). Currently this nonce is generated thanks to the Initialization Vector (IV) provided in each ESP packet ([RFC4303]). This practice is designated in this document as "explicit IV".

In some contexts, such as IoT, it may be preferable to avoid carrying the extra bytes associated to the IV and instead generate it locally on each peer. The local generation of the IV is designated in this document as "implicit IV".

The size of this IV depends on the specific algorithm, but all of the algorithms mentioned above take an 8-octet IV.

This document defines how to compute the IV locally when it is implicit. It also specifies how peers agree with the Internet Key Exchange version 2 (IKEv2 - [RFC7296]) on using an implicit IV versus an explicit IV.

This document limits its scope to the algorithms mentioned above. Other algorithms with similar properties may later be defined to use similar mechanisms.

This document does not consider AES-CBC ([RFC3602]) as AES-CBC requires the IV to be unpredictable. Deriving it directly from the packet counter as described below is insecure as mentioned in Security Consideration of [RFC3602] and has led to real world chosen plain-text attack such as BEAST [BEAST].

This document does not consider AES-CTR [RFC3686] as it focuses on the recommended AEAD suites provided in [RFC8221].

3. Terminology

- o IoT: Internet of Things.
- o IV: Initialization Vector.
- o IIV: Implicit Initialization Vector.
- o Nonce: a fixed-size octet string used only once. In our case, the nonce takes the IV as input and is provided as an input parameter for encryption/decryption.

4. Implicit IV

With the algorithms listed in Section 2, the 8-byte IV MUST NOT repeat for a given key. The binding between an ESP packet and its IV is provided using the Sequence Number or the Extended Sequence Number. Figure 1 and Figure 2 represent the IV with a regular 4-byte Sequence Number and with an 8-byte Extended Sequence Number respectively.

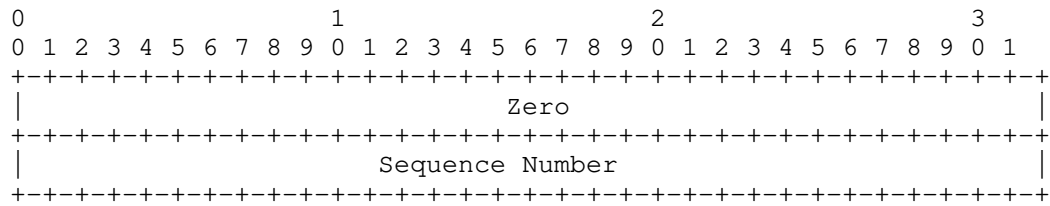


Figure 1: Implicit IV with a 4 byte Sequence Number

- o Sequence Number: the 4 byte Sequence Number carried in the ESP packet.
- o Zero: a 4 byte array with all bits set to zero.

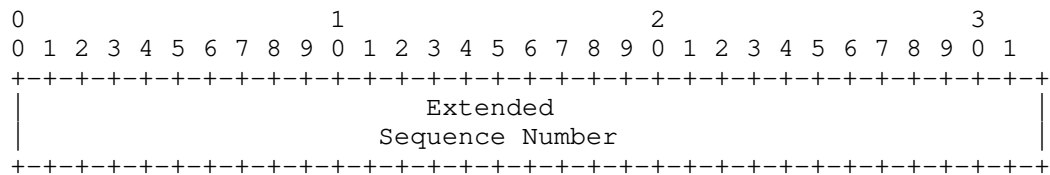


Figure 2: Implicit IV with an 8-byte Extended Sequence Number

- o Extended Sequence Number: the 8-byte Extended Sequence Number of the Security Association. The 4 byte low order bytes are carried in the ESP packet.

This document solely defines the IV generation of the algorithms defined in [RFC4106] for AES-GCM, [RFC4309] for AES-CCM and [RFC7634] for ChaCha20-Poly1305. All other aspects and parameters of those algorithms are unchanged, and are used as defined in their respective specifications.

5. IKEv2 Initiator Behavior

An initiator supporting this feature SHOULD propose implicit IV (IIV) algorithms in the Transform Type 1 (Encryption Algorithm) Substructure of the Proposal Substructure inside the Security Association Payload (SA Payload) in the IKEv2 Exchange. To facilitate backward compatibility with non-supporting peers the initiator SHOULD also include those same algorithms with explicit IV as separate transforms.

6. IKEv2 Responder Behavior

The rules of SA Payload processing require that responder picks its algorithms from the proposal sent by the initiator, thus this will ensure that the responder will never send an SA payload containing the IIV transform to an initiator that did not propose it.

7. Security Considerations

Nonce generation for these algorithms has not been explicitly defined. It has been left to the implementation as long as certain security requirements are met. Typically, for AES-GCM, AES-CCM and ChaCha20-Poly1305, the IV is not allowed to be repeated for one particular key. This document provides an explicit and normative way to generate IVs. The mechanism described in this document meets the IV security requirements of all relevant algorithms.

As the IV must not repeat for one SA when Counter-Mode ciphers are used, implicit IV as described in this document **MUST NOT** be used in setups with the chance that the Sequence Number overlaps for one SA. The sender's counter and the receiver's counter **MUST** be reset (by establishing a new SA and thus a new key) prior to the transmission of the 2³²nd packet for an SA that uses a non extended Sequence Number (respectively the 2⁶⁴nd packet for an SA that uses an Extended Sequence Number). This prevents sequence number overlaps for the mundane point-to-point case. Multicast as described in [RFC5374], [RFC6407] and [I-D.yeung-g-ikev2] is a prominent example, where many senders share one secret and thus one SA. As such, Implicit IV may only be used with Multicast if some mechanisms are employed that prevent Sequence Number to overlap for one SA, otherwise Implicit IV **MUST NOT** be used with Multicast.

This document defines three new encryption transforms that use implicit IV. Unlike most encryption transforms defined to date, which can be used for both ESP and IKEv2, these transforms are defined for ESP only and cannot be used in IKEv2. The reason is that IKEv2 messages don't contain a unique per-message value that can be used for IV generation. The Message-ID field in IKEv2 header is similar to the SN field in ESP header, but recent IKEv2 extensions ([RFC6311], [RFC7383]) do allow it to repeat, so there is not an easy way to derive unique IV from IKEv2 header fields.

8. IANA Considerations

The IANA has updated the "Internet Key Exchange Version 2 (IKEv2) Parameters" [RFC7296] by adding new code points to the "Transform Type Values"/"Transform Type 1 - Encryption Algorithm Transform IDs" registry [IANA]:

- ENCR_AES_CCM_8_IIV: 29
- ENCR_AES_GCM_16_IIV: 30
- ENCR_CHACHA20_POLY1305_IIV: 31

These algorithms should be added with this document as ESP Reference and "Not Allowed" for IKEv2 Reference.

9. Acknowledgements

We would like to thank Valery Smyslov, Eric Vyncke, Alexey Melnikov, Adam Roach, Magnus Nystrom (security directorate), as well as our three Security ADs Eric Rescorla, Benjamin Kaduk and Roman Danyliw for their valuable comments. We also would like to thank David Schinazi for its implementation, as well as the ipsecme chairs Tero Kivinen and David Waltermire for moving this work forward.

NOTE TO THE EDITOR Eric has a accent on E and Magnus has double points on o.

10. References

10.1. Normative References

- [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>>.
- [RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, DOI 10.17487/RFC3602, September 2003, <<https://www.rfc-editor.org/info/rfc3602>>.
- [RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, DOI 10.17487/RFC3686, January 2004, <<https://www.rfc-editor.org/info/rfc3686>>.
- [RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <<https://www.rfc-editor.org/info/rfc4106>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.

- [RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <<https://www.rfc-editor.org/info/rfc4309>>.
- [RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, DOI 10.17487/RFC5374, November 2008, <<https://www.rfc-editor.org/info/rfc5374>>.
- [RFC6311] Singh, R., Ed., Kalyani, G., Nir, Y., Sheffer, Y., and D. Zhang, "Protocol Support for High Availability of IKEv2/IPsec", RFC 6311, DOI 10.17487/RFC6311, July 2011, <<https://www.rfc-editor.org/info/rfc6311>>.
- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <<https://www.rfc-editor.org/info/rfc6407>>.
- [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>>.
- [RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <<https://www.rfc-editor.org/info/rfc7383>>.
- [RFC7634] Nir, Y., "ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec", RFC 7634, DOI 10.17487/RFC7634, August 2015, <<https://www.rfc-editor.org/info/rfc7634>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 8221, DOI 10.17487/RFC8221, October 2017, <<https://www.rfc-editor.org/info/rfc8221>>.

10.2. Informational References

- [BEAST] Thai, T. and J. Juliano, "Here Come The xor Ninjas", , May 2011, <https://www.researchgate.net/publication/266529975_Here_Come_The_Ninjas>.
- [I-D.yeung-g-ikev2] Weis, B. and V. Smyslov, "Group Key Management using IKEv2", draft-yeung-g-ikev2-16 (work in progress), July 2019.
- [IANA] "IANA IKEv2 Parameter - Type 1 - Encryption Algorithm Transform IDs", <<https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-5>>.

Authors' Addresses

Daniel Migault
Ericsson
8275 Trans Canada Route
Saint Laurent, QC H4S 0B6
Canada

Email: daniel.migault@ericsson.com

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

Email: guggemos@mn-team.org
URI: <http://mn-team.org/~guggemos>

Yoav Nir
Dell EMC
9 Andrei Sakharov St
Haifa 3190500
Israel

Email: ynir.ietf@gmail.com

ipsecme
Internet-Draft
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: June 20, 2021

M. Boucadair
Orange
December 17, 2020

IKEv2 Notification Status Types for IPv4/IPv6 Coexistence
draft-ietf-ipsecme-ipv6-ipv4-codes-06

Abstract

This document specifies new IKEv2 notification status types to better manage IPv4 and IPv6 co-existence by allowing the responder to signal to the initiator which address families are allowed.

This document updates RFC7296.

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 June 20, 2021.

Copyright Notice

Copyright (c) 2020 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.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Why Not INTERNAL_ADDRESS_FAILURE?	3
4. IP6_ALLOWED and IP4_ALLOWED Status Types	4
5. An Update to RFC7296	4
6. Security Considerations	6
7. IANA Considerations	6
8. Acknowledgements	6
9. References	6
9.1. Normative References	6
9.2. Informative References	7
Author's Address	7

1. Introduction

As described in [RFC7849], if the subscription data or network configuration allows only one IP address family (IPv4 or IPv6), the cellular host must not request a second PDP-Context (Section 3.2 of [RFC6459]) to the same Access Point Name (APN) for the other IP address family (AF). The Third Generation Partnership Project (3GPP) network informs the cellular host about allowed Packet Data Protocol (PDP) types by means of Session Management (SM) cause codes. In particular, the following cause codes can be returned:

- o cause #50 "PDP type IPv4 only allowed": This cause code is used by the network to indicate that only PDP type IPv4 is allowed for the requested Public Data Network (PDN) connectivity.
- o cause #51 "PDP type IPv6 only allowed": This cause code is used by the network to indicate that only PDP type IPv6 is allowed for the requested PDN connectivity.
- o cause #52 "single address bearers only allowed": This cause code is used by the network to indicate that the requested PDN connectivity is accepted with the restriction that only single IP version bearers are allowed.

If the requested IPv4v6 PDP-Context is not supported by the network but IPv4 and IPv6 PDP types are allowed, then the cellular host will be configured with an IPv4 address or an IPv6 prefix by the network. It must initiate another PDP-Context activation of the other address family in addition to the one already activated for a given APN. The purpose of initiating a second PDP-Context is to achieve dual-stack

connectivity (that is, IPv4 and IPv6 connectivity) by means of two PDP-Contexts.

When the User Equipment (UE) attaches to the 3GPP network using a non-3GPP access network (e.g., Wireless Local Area Network (WLAN)), there are no equivalent Internet Key Exchange Protocol Version 2 (IKEv2) capabilities [RFC7296] notification codes for the 3GPP network to inform the UE why an IP address family is not assigned or whether that UE should retry with another address family.

This document fills that void by introducing new IKEv2 notification status types for the sake of deterministic UE behaviors (Section 4).

These notification status types are not specific to 3GPP architectures, but can be used in other deployment contexts. Cellular networks are provided as an illustration example.

2. Terminology

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

This document makes use of the terms defined in [RFC7296]. In particular, readers should be familiar with "initiator" and "responder" terms used in that document.

3. Why Not INTERNAL_ADDRESS_FAILURE?

The following address assignment failures may be encountered when an initiator requests assignment of IP addresses/prefixes:

- o An initiator asks for IPv_x, but IPv_x address assignment is not supported by the responder.
- o An initiator requests both IPv4 and IPv6 addresses, but only IPv4 address assignment is supported by the responder.
- o An initiator requests both IPv4 and IPv6 addresses, but only IPv6 prefix assignment is supported by the responder.
- o An initiator asks for both IPv4 and IPv6 addresses, but only one address family can be assigned by the responder for policy reasons.

Section 3.15.4 of [RFC7296] defines a generic notification error type (INTERNAL_ADDRESS_FAILURE) that is related to a failure to handle an address assignment request. The responder sends INTERNAL_ADDRESS_FAILURE only if no addresses can be assigned. This behavior does not explicitly allow an initiator to determine why a given address family is not assigned, nor whether it should try using another address family. INTERNAL_ADDRESS_FAILURE is a catch-all error type when an address-related issue is encountered by an IKEv2 responder.

INTERNAL_ADDRESS_FAILURE does not provide sufficient hints to the IKEv2 initiator to adjust its behavior.

4. IP6_ALLOWED and IP4_ALLOWED Status Types

IP6_ALLOWED and IP4_ALLOWED notification status types (see Section 7) are defined to inform the initiator about the responder's address family assignment support capabilities, and to report to the initiator the reason why an address assignment failed. These notification status types are used by the initiator to adjust its behavior accordingly (Section 5).

No data is associated with these notifications.

5. An Update to RFC7296

If the initiator is dual-stack (i.e., supports both IPv4 and IPv6), it MUST include both address families configuration attributes in its configuration request (absent explicit policy/configuration otherwise). More details about IPv4 and IPv6 configuration attributes are provided in Section 3.15 of [RFC7296]. These attributes are used to infer the requested/assigned AFs listed in Table 1.

The responder MUST include IP6_ALLOWED and/or IP4_ALLOWED notification status type in a response to an address assignment request as indicated in Table 1.

Requested AF(s) (Initiator)	Supported AF(s) (Responder)	Assigned AF(s) (Responder)	Returned Notification Status Type(s) (Responder)
IPv4	IPv6	None	IP6_ALLOWED
IPv4	IPv4	IPv4	IP4_ALLOWED
IPv4	IPv4 and IPv6	IPv4	IP4_ALLOWED, IP6_ALLOWED
IPv6	IPv6	IPv6	IP6_ALLOWED
IPv6	IPv4	None	IP4_ALLOWED
IPv6	IPv4 and IPv6	IPv6	IP4_ALLOWED, IP6_ALLOWED
IPv4 and IPv6	IPv4	IPv4	IP4_ALLOWED
IPv4 and IPv6	IPv6	IPv6	IP6_ALLOWED
IPv4 and IPv6	IPv4 and IPv6	IPv4 and IPv6	IP4_ALLOWED, IP6_ALLOWED
IPv4 and IPv6	IPv4 or IPv6 (Policy-based)	IPv4 or IPv6	IP4_ALLOWED, IP6_ALLOWED

Table 1: Returned Notification Status Types

If the initiator only receives one single notification IP4_ALLOWED or IP6_ALLOWED from the responder, the initiator MUST NOT send a subsequent request for an alternate address family not supported by the responder.

If a dual-stack initiator requests only an IPv6 prefix (or an IPv4 address) but only receives IP4_ALLOWED (or IP6_ALLOWED) notification status type from the responder, the initiator MUST send a request for IPv4 address(es) (or IPv6 prefix(es)).

If a dual-stack initiator requests both an IPv6 prefix and an IPv4 address but receives an IPv6 prefix (or an IPv4 address) only with both IP4_ALLOWED and IP6_ALLOWED notification status types from the responder, the initiator MAY send a request for the other AF (i.e., IPv4 address (or IPv6 prefix)). In such case, the initiator MUST create a new IKE Security Association (SA) and request that another address family using the new IKE SA.

For other address-related error cases that have not been covered by the aforementioned notification status types, the responder/initiator MUST follow the procedure defined in Section 3.15.4 of [RFC7296].

6. Security Considerations

Since the IPv4/IPv6 capabilities of a node are readily determined from the traffic it generates, this document does not introduce any new security considerations compared to the ones described in [RFC7296], which continue to apply.

7. IANA Considerations

This document requests IANA to update the "IKEv2 Notify Message Types - Status Types" registry available at: <https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml> with the following status types:

Value	NOTIFY MESSAGES - STATUS TYPES	Reference
TBD	IP4_ALLOWED	[This-Document]
TBD	IP6_ALLOWED	[This-Document]

8. Acknowledgements

Many thanks to Christian Jacquenet for the review.

Thanks to Paul Wouters, Yaov Nir, Valery Smyslov, Daniel Migault, Tero Kivinen, and Michael Richardson for the comments and review.

Thanks to Benjamin Kaduk for the AD review.

Thanks to Murray Kucherawy, Eric Vyncke, and Robert Wilton for the IESG review.

9. References

9.1. Normative References

- [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>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [RFC6459] Korhonen, J., Ed., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, DOI 10.17487/RFC6459, January 2012, <<https://www.rfc-editor.org/info/rfc6459>>.
- [RFC7849] Binet, D., Boucadair, M., Vizdal, A., Chen, G., Heatley, N., Chandler, R., Michaud, D., Lopez, D., and W. Haeffner, "An IPv6 Profile for 3GPP Mobile Devices", RFC 7849, DOI 10.17487/RFC7849, May 2016, <<https://www.rfc-editor.org/info/rfc7849>>.

Author's Address

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Network
Internet-Draft
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: 25 September 2022

P. Wouters
Aiven
S. Prasad
Red Hat
24 March 2022

Labeled IPsec Traffic Selector support for IKEv2
draft-ietf-ipsecme-labeled-ipsec-07

Abstract

This document defines a new Traffic Selector (TS) Type for Internet Key Exchange version 2 to add support for negotiating Mandatory Access Control (MAC) security labels as a traffic selector of the Security Policy Database (SPD). Security Labels for IPsec are also known as "Labeled IPsec". The new TS type is TS_SECLABEL, which consists of a variable length opaque field specifying the security label. This document updates the IKEv2 TS negotiation specified in RFC 7296 Section 2.9.

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 25 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
1.2. Traffic Selector clarification	3
1.3. Traffic Selector update	4
2. TS_SECLABEL Traffic Selector Type	4
2.1. TS_SECLABEL payload format	4
2.2. TS_SECLABEL properties	4
3. Traffic Selector negotiation	5
3.1. Example TS negotiation	6
3.2. Considerations for using multiple TS_TYPES in a TS	6
4. Security Considerations	7
5. IANA Considerations	7
6. Implementation Status	7
6.1. Libreswan	8
7. Acknowledgements	9
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Authors' Addresses	10

1. Introduction

In computer security, Mandatory Access Control usually refers to systems in which all subjects and objects are assigned a security label. A security label is comprised of a set of security attributes. The security labels along with a system authorization policy determine access. Rules within the system authorization policy determine whether the access will be granted based on the security attributes of the subject and object.

Traditionally, security labels used by Multilevel Systems (MLS) are comprised of a sensitivity level (or classification) field and a compartment (or category) field, as defined in [FIPS188] and [RFC5570]. As MAC systems evolved, other MAC models gained in popularity. For example, SELinux, a Flux Advanced Security Kernel (FLASK) implementation, has security labels represented as colon-separated ASCII strings composed of values for identity, role, and type. The security labels are often referred to as security contexts.

Traffic Selector (TS) payloads specify the selection criteria for packets that will be forwarded over the newly set up IPsec SA as enforced by the Security Policy Database (SPD, see [RFC4301]). This document updates the Traffic Selector negotiation specified in Section 2.9 of [RFC7296].

This document specifies a new Traffic Selector Type TS_SECLABEL for IKEv2 that can be used to negotiate security labels as additional selectors for the Security Policy Database (SPD) to further restrict the type of traffic allowed to be sent and received over the IPsec SA.

1.1. Requirements Language

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

1.2. Traffic Selector clarification

The negotiation of Traffic Selectors is specified in Section 2.9 of [RFC7296] where it defines two TS Types (TS_IPV4_ADDR_RANGE and TS_IPV6_ADDR_RANGE). The Traffic Selector payload format is specified in Section 3.13 of [RFC7296]. However, the term Traffic Selector is used to denote the traffic selector payloads and individual traffic selectors of that payload. Sometimes the exact meaning can only be learned from context or if the item is written in plural ("Traffic Selectors" or "TSs"). This section clarifies these terms as follows:

A Traffic Selector (no acronym) is one selector for traffic of a specific Traffic Selector Type (TS_TYPE). For example a Traffic Selector of TS_TYPE TS_IPV4_ADDR_RANGE for UDP traffic in the IP network 198.51.100.0/24 covering all ports, is denoted as (17, 0, 198.51.100.0-198.51.100.255)

A Traffic Selector payload (TS) is a set of one or more Traffic Selectors of the same or different TS_TYPES, but MUST include at least one TS_TYPE of TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE. For example, the above Traffic Selector by itself in a TS payload is denoted as TS((17, 0, 198.51.100.0-198.51.100.255))

1.3. Traffic Selector update

The negotiation of Traffic Selectors is specified in Section 2.9 of [RFC7296] and states that the TSi/TSr payloads MUST contain at least one Traffic Selector type. This document updates the text to mean that the TSi/TSr payloads MUST contain at least one Traffic Selector of type TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE, as other Traffic Selector types can be defined that are complimentary to these Traffic Selector Types and cannot be selected on their own without TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE. The below defined TS_SECLABEL Traffic Selector Type is an example of this.

2. TS_SECLABEL Traffic Selector Type

This document defines a new TS Type, TS_SECLABEL that contains a single new opaque Security Label.

2.1. TS_SECLABEL payload format

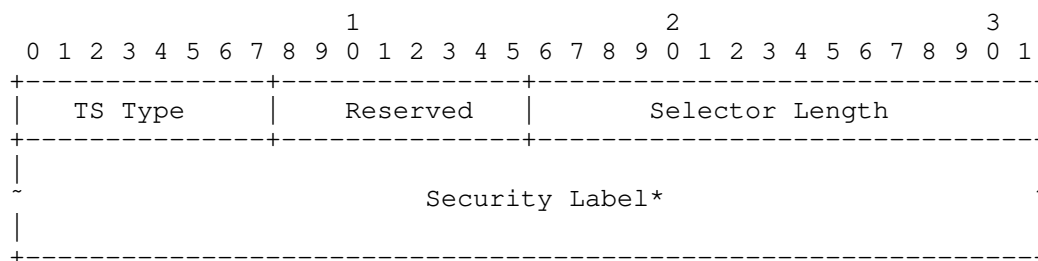


Figure 1: Labeled IPsec Traffic Selector

*Note: All fields other than TS Type and Selector Length depend on the TS Type. The fields shown is for TS Type TS_SECLABEL, the selector this document defines.

- * TS Type (one octet) - Set to 10 for TS_SECLABEL,
- * Selector Length (2 octets, unsigned integer) - Specifies the length of this Traffic Selector substructure including the header.
- * Security Label - An opaque byte stream of at least one octet.

2.2. TS_SECLABEL properties

The TS_SECLABEL Traffic Selector Type does not support narrowing or wildcards. It MUST be used as an exact match value.

The Security Label contents are opaque to the IKE implementation. That is, the IKE implementation might not have any knowledge of the meaning of this selector, other than as a type and opaque value to pass to the SPD.

A zero length Security Label MUST NOT be used. If a received TS payload contains a TS_TYPE of TS_SECLABEL with a zero length Security Label, that specific Traffic Selector MUST be ignored. If no other Traffic Selector of TS_TYPE TS_SECLABEL can be selected, a TS_UNACCEPTABLE Error Notify message MUST be returned. A zero length Security Label MUST NOT be interpreted as a wildcard security label.

If multiple Security Labels are allowed for a given IP protocol, start and end address/port match, the initiator includes all of the acceptable TS_SECLABEL's and the responder MUST select one of them.

If the Security Label traffic selector is optional from a configuration point of view, the initiator will have to choose which TS payload to attempt first. If it includes the Security Label and receives a TS_UNACCEPTABLE, it can attempt a new Child SA negotiation without that Security Label.

A responder that selected a TS with TS_SECLABEL MUST use the Security Label for all selector operations on the resulting TS. It MUST NOT select a TS_SECLABEL without using the specified Security Label, even if it deems the Security Label optional, as the initiator has indicated (and expects) that Security Label will be set for all traffic matching the negotiated TS.

3. Traffic Selector negotiation

This document updates the [RFC7296] specification as follows:

Each TS payload (TSi and TSr) MUST contain at least one TS_TYPE of TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE.

Each TS payload (TSi or TSr) MAY contain one or more other TS_TYPES, such as TS_SECLABEL.

A responder MUST create each TS response by creating one of more (narrowed or not) TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE entries, plus one of each further TS_TYPE present in the offered TS by the initiator. If this is not possible, it MUST return a TS_UNACCEPTABLE Error Notify payload.

If a specific TS_TYPE (other than TS_IPV4_ADDR_RANGE or TS_IPV6_ADDR_RANGE which are mandatory) is deemed optional, the initiator SHOULD first try to negotiate the Child SA with the TS

payload including the optional TS_TYPE. Upon receiving TS_UNACCEPTABLE, it SHOULD attempt a new Child SA negotiation using the same TS but without the optional TS_TYPE.

3.1. Example TS negotiation

An initiator could send:

```
TSi = ((17,24233,198.51.12-198.51.12),
      (17,0,192.0.2.0-192.0.2.255),
      (0,0,198.51.0-198.51.255),
      TS_SECLABEL1, TS_SECLABEL2)

TSr = ((17,53,203.0.113.1-203.0.113.1),
      (17,0,203.0.113.0-203.0.113.255),
      (0,0,203.0.113.0-203.0.113.255),
      TS_SECLABEL1, TS_SECLABEL2)
```

Figure 2: initiator TS payloads example

The responder could answer with the following example:

```
TSi = ((0,0,198.51.0-198.51.255),
      TS_SECLABEL1)

TSr = (((0,0,203.0.113.0-203.0.113.255),
      TS_SECLABEL1)
```

Figure 3: responder TS payloads example

3.2. Considerations for using multiple TS_TYPES in a TS

It would be unlikely that the traffic for TSi and TSr would have a different Security Label, but this specification does allow this to be specified. If the initiator does not support this, and wants to prevent the responder from picking different labels for the TSi / TSr payloads, it should attempt a Child SA negotiation with only the first Security Label first, and upon failure retry a new Child SA negotiation with only the second Security Label.

If different IP ranges can only use different specific Security Labels, than these should be negotiated in two different Child SA negotiations. If in the example above, the initiator only allows 192.0.2.0/24 with TS_SECLABEL1, and 198.51.0/24 with TS_SECLABEL2, than it MUST NOT combine these two ranges and security labels into one Child SA negotiation.

The mechanism of narrowing of Traffic Selectors with TS_IPV4_ADDR_RANGE and TS_IPV6_ADDR_RANGE does not apply to TS_SECLABEL as the Security Label itself is not interpreted and cannot be narrowed. It MUST be matched exactly. Since a rekey MUST NOT narrow down the Traffic Selectors narrower than the scope currently in use, the only valid choice of TS_SECLABEL for a rekey is the identical TS_SECLABEL that is in use by the Child SA being rekeyed. If the TS_LABEL is missing from the TS during the rekey negotiation, the negotiation MUST fail with TS_UNACCEPTABLE.

4. Security Considerations

It is assumed that the Security Label can be matched by the IKE implementation to its own configured value, even if the IKE implementation itself cannot interpret the Security Label value.

A packet that matches an SPD entry for all components except the Security Label would be treated as "not matching". If no other SPD entries match, the (mis-labeled) traffic might end up being transmitted in the clear. It is presumed that other Mandatory Access Control methods are in place to prevent mis-labeled traffic from reaching the IPsec subsystem, or that the IPsec subsystem itself would install a REJECT/DISCARD rule in the SPD to prevent unlabeled traffic otherwise matching a labeled security SPD rule from being transmitted without IPsec protection.

5. IANA Considerations

This document defines one new entry in the IKEv2 Traffic Selector Types registry:

[Note to RFC Editor (please remove before publication): This value has already been added via Early Allocation.]

Value	TS Type	Reference
-----	-----	-----
10	TS_SECLABEL	[this document]

Figure 4

6. Implementation Status

[Note to RFC Editor: Please remove this section and the reference to [RFC7942] before publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942].

The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Authors are requested to add a note to the RFC Editor at the top of this section, advising the Editor to remove the entire section before publication, as well as the reference to [RFC7942].

6.1. Libreswan

Organization: The Libreswan Project

Name: <https://lists.libreswan.org/mailman/listinfo/swan-dev/>

Description: Implementation has been released as part of libreswan version 4.4.

Level of maturity: beta

Coverage: Implements the entire draft using SELinux based labels

Licensing: GPLv2

Implementation experience: No interop testing has been done yet.
The code works as proof of concept, but is not yet production ready when using multiple different labels with on-demand kernel ACQUIRES.

Contact: Libreswan Development: swan-dev@libreswan.org

7. Acknowledgements

A large part of the introduction text was taken verbatim from [draft-jml-ipsec-ikev2-security-label] whose authors are J Latten, D. Quigley and J. Lu. Valery Smyslov provided valuable input regarding IKEv2 Traffic Selector semantics.

8. References

8.1. Normative References

- [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>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

- [draft-jml-ipsec-ikev2-security-label] Latten, J., Quigley, D., and J. Lu, "Security Label Extension to IKE", 28 January 2011.
- [FIPS188] NIST, "National Institute of Standards and Technology, "Standard Security Label for Information Transfer"", Federal Information Processing Standard (FIPS) Publication 188, September 1994, <<https://csrc.nist.gov/publications/detail/fips/188/archive/1994-09-06>>.
- [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>>.
- [RFC5570] StJohns, M., Atkinson, R., and G. Thomas, "Common Architecture Label IPv6 Security Option (CALIPSO)", RFC 5570, DOI 10.17487/RFC5570, July 2009, <<https://www.rfc-editor.org/info/rfc5570>>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <<https://www.rfc-editor.org/info/rfc7942>>.

Authors' Addresses

Paul Wouters
Aiven
Email: paul.wouters@aiven.io

Sahana Prasad
Red Hat
Email: sahana@redhat.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: July 17, 2020

S. Fluhrer
P. Kampanakis
D. McGrew
Cisco Systems
V. Smyslov
ELVIS-PLUS
January 14, 2020

Mixing Preshared Keys in IKEv2 for Post-quantum Security
draft-ietf-ipsecme-qr-ikev2-11

Abstract

The possibility of quantum computers poses a serious challenge to cryptographic algorithms deployed widely today. IKEv2 is one example of a cryptosystem that could be broken; someone storing VPN communications today could decrypt them at a later time when a quantum computer is available. It is anticipated that IKEv2 will be extended to support quantum-secure key exchange algorithms; however that is not likely to happen in the near term. To address this problem before then, this document describes an extension of IKEv2 to allow it to be resistant to a quantum computer, by using preshared keys.

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 July 17, 2020.

Copyright Notice

Copyright (c) 2020 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.

Table of Contents

1. Introduction	2
1.1. Changes	3
1.2. Requirements Language	6
2. Assumptions	6
3. Exchanges	6
4. Upgrade procedure	11
5. PPK	12
5.1. PPK_ID format	12
5.2. Operational Considerations	13
5.2.1. PPK Distribution	13
5.2.2. Group PPK	13
5.2.3. PPK-only Authentication	14
6. Security Considerations	14
7. IANA Considerations	16
8. References	17
8.1. Normative References	17
8.2. Informational References	18
Appendix A. Discussion and Rationale	19
Appendix B. Acknowledgements	20
Authors' Addresses	20

1. Introduction

Recent achievements in developing quantum computers demonstrate that it is probably feasible to build a cryptographically significant one. If such a computer is implemented, many of the cryptographic algorithms and protocols currently in use would be insecure. A quantum computer would be able to solve DH and ECDH problems in polynomial time [I-D.hoffman-c2pq], and this would imply that the security of existing IKEv2 [RFC7296] systems would be compromised. IKEv1 [RFC2409], when used with strong preshared keys, is not vulnerable to quantum attacks, because those keys are one of the inputs to the key derivation function. If the preshared key has sufficient entropy and the PRF, encryption and authentication transforms are quantum-secure, then the resulting system is believed

to be quantum-secure, that is, secure against classical attackers of today or future attackers with a quantum computer.

This document describes a way to extend IKEv2 to have a similar property; assuming that the two end systems share a long secret key, then the resulting exchange is quantum-secure. By bringing post-quantum security to IKEv2, this document removes the need to use an obsolete version of the Internet Key Exchange in order to achieve that security goal.

The general idea is that we add an additional secret that is shared between the initiator and the responder; this secret is in addition to the authentication method that is already provided within IKEv2. We stir this secret into the SK_d value, which is used to generate the key material (KEYMAT) and the SKEYSEED for the child SAs; this secret provides quantum resistance to the IPsec SAs (and any child IKE SAs). We also stir the secret into the SK_pi, SK_pr values; this allows both sides to detect a secret mismatch cleanly.

It was considered important to minimize the changes to IKEv2. The existing mechanisms to do authentication and key exchange remain in place (that is, we continue to do (EC)DH, and potentially PKI authentication if configured). This document does not replace the authentication checks that the protocol does; instead, they are strengthened by using an additional secret key.

1.1. Changes

RFC EDITOR PLEASE DELETE THIS SECTION.

Changes in this draft in each version iterations.

draft-ietf-ipsecme-qr-ikev2-11

- o Updates the IANA section based on Eric V.'s IESG Review.
- o Updates based on IESG Reviews (Alissa, Adam, Barry, Alexey, Mijra, Roman, Martin).

draft-ietf-ipsecme-qr-ikev2-10

- o Addresses issues raised during IETF LC.

draft-ietf-ipsecme-qr-ikev2-09

- o Addresses issues raised in AD review.

draft-ietf-ipsecme-qr-ikev2-08

- o Editorial changes.

draft-ietf-ipsecme-qr-ikev2-07

- o Editorial changes.

draft-ietf-ipsecme-qr-ikev2-06

- o Editorial changes.

draft-ietf-ipsecme-qr-ikev2-05

- o Addressed comments received during WGLC.

draft-ietf-ipsecme-qr-ikev2-04

- o Using Group PPK is clarified based on comment from Quynh Dang.

draft-ietf-ipsecme-qr-ikev2-03

- o Editorial changes and minor text nit fixes.

- o Integrated Tommy P. text suggestions.

draft-ietf-ipsecme-qr-ikev2-02

- o Added note that the PPK is stirred in the initial IKE SA setup only.

- o Added note about the initiator ignoring any content in the PPK_IDENTITY notification from the responder.

- o fixed Tero's suggestions from 2/6/1028

- o Added IANA assigned message types where necessary.

- o fixed minor text nits

draft-ietf-ipsecme-qr-ikev2-01

- o Nits and minor fixes.

- o prf is replaced with prf+ for the SK_d and SK_pi/r calculations.

- o Clarified using PPK in case of EAP authentication.

- o PPK_SUPPORT notification is changed to USE_PPK to better reflect its purpose.

draft-ietf-ipsecme-qr-ikev2-00

- o Migrated from draft-fluhrer-qr-ikev2-05 to draft-ietf-ipsecme-qr-ikev2-00 that is a WG item.

draft-fluhrer-qr-ikev2-05

- o Nits and editorial fixes.
- o Made PPK_ID format and PPK Distributions subsection of the PPK section. Also added an Operational Considerations section.
- o Added comment about Child SA rekey in the Security Considerations section.
- o Added NO_PPK_AUTH to solve the cases where a PPK_ID is not configured for a responder.
- o Various text changes and clarifications.
- o Expanded Security Considerations section to describe some security concerns and how they should be addressed.

draft-fluhrer-qr-ikev2-03

- o Modified how we stir the PPK into the IKEv2 secret state.
- o Modified how the use of PPKs is negotiated.

draft-fluhrer-qr-ikev2-02

- o Simplified the protocol by stirring in the preshared key into the child SAs; this avoids the problem of having the responder decide which preshared key to use (as it knows the initiator identity at that point); it does mean that someone with a quantum computer can recover the initial IKE negotiation.
- o Removed positive endorsements of various algorithms. Retained warnings about algorithms known to be weak against a quantum computer.

draft-fluhrer-qr-ikev2-01

- o Added explicit guidance as to what IKE and IPsec algorithms are quantum resistant.

draft-fluhrer-qr-ikev2-00

- o We switched from using vendor ID's to transmit the additional data to notifications.
- o We added a mandatory cookie exchange to allow the server to communicate to the client before the initial exchange.
- o We added algorithm agility by having the server tell the client what algorithm to use in the cookie exchange.
- o We have the server specify the PPK Indicator Input, which allows the server to make a trade-off between the efficiency for the search of the clients PPK, and the anonymity of the client.
- o We now use the negotiated PRF (rather than a fixed HMAC-SHA256) to transform the nonces during the KDF.

1.2. Requirements Language

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

2. Assumptions

We assume that each IKE peer has a list of Post-quantum Preshared Keys (PPK) along with their identifiers (PPK_ID), and any potential IKE initiator selects which PPK to use with any specific responder. In addition, implementations have a configurable flag that determines whether this post-quantum preshared key is mandatory. This PPK is independent of the preshared key (if any) that the IKEv2 protocol uses to perform authentication (because the preshared key in IKEv2 is not used for any key derivation, and thus doesn't protect against quantum computers). The PPK specific configuration that is assumed to be on each node consists of the following tuple:

Peer, PPK, PPK_ID, mandatory_or_not

3. Exchanges

If the initiator is configured to use a post-quantum preshared key with the responder (whether or not the use of the PPK is mandatory), then it MUST include a notification USE_PPK in the IKE_SA_INIT request message as follows:

HDR, SAi1, KEi, Ni, N(USE_PPK) --->

```
<--- HDR, SAr1, KEr, Nr, [CERTREQ,] N(USE_PPK)
```

```

SKEYSEED = prf(Ni | Nr, g^ir)
{SK_d' | SK_ai | SK_ar | SK_ei | SK_er | SK_pi' | SK_pr' }
      = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr }

SK_d  = prf+ (PPK, SK_d')
SK_pi = prf+ (PPK, SK_pi')
SK_pr = prf+ (PPK, SK_pr')

```

That is, we use the standard IKEv2 key derivation process except that the three resulting subkeys SK_d, SK_pi, SK_pr (marked with primes in the formula above) are then run through the prf+ again, this time using the PPK as the key. The result is the unprimed versions of these keys which are then used as inputs to subsequent steps of the IKEv2 exchange.

Using a prf+ construction ensures that it is always possible to get the resulting keys of the same size as the initial ones, even if the underlying PRF has output size different from its key size. Note, that at the time of this writing, all PRFs defined for use in IKEv2 [IKEV2-IANA-PRFS] had output size equal to the (preferred) key size. For such PRFs only the first iteration of prf+ is needed:

```

SK_d  = prf (PPK, SK_d' | 0x01)
SK_pi = prf (PPK, SK_pi' | 0x01)
SK_pr = prf (PPK, SK_pr' | 0x01)

```

Note that the PPK is used in SK_d, SK_pi and SK_pr calculation only during the initial IKE SA setup. It MUST NOT be used when these subkeys are calculated as result of IKE SA rekey, resumption or other similar operation.

The initiator then sends the IKE_AUTH request message, including the PPK_ID value as follows:

Initiator	Responder

HDR, SK {IDi, [CERT,] [CERTREQ,]	
[IDr,] AUTH, SAI2,	
TSi, TSr, N(PPK_IDENTITY, PPK_ID), [N(NO_PPK_AUTH)]} --->	

PPK_IDENTITY is a status notification with the type 16436; it has a protocol ID of 0, no SPI and a notification data that consists of the identifier PPK_ID.

A situation may happen when the responder has some PPKs, but doesn't have a PPK with the PPK_ID received from the initiator. In this case the responder cannot continue with PPK (in particular, it cannot authenticate the initiator), but the responder could be able to

continue with normal IKEv2 protocol if the initiator provided its authentication data computed as in normal IKEv2, without using PPKs. For this purpose, if using PPKs for communication with this responder is optional for the initiator (based on the mandatory_or_not flag), then the initiator MUST include a NO_PPK_AUTH notification in the above message. This notification informs the responder that PPK is optional and allows for authenticating the initiator without using PPK.

NO_PPK_AUTH is a status notification with the type 16437; it has a protocol ID of 0 and no SPI. The Notification Data field contains the initiator's authentication data computed using SK_pi', which has been computed without using PPKs. This is the same data that would normally be placed in the Authentication Data field of an AUTH payload. Since the Auth Method field is not present in the notification, the authentication method used for computing the authentication data MUST be the same as method indicated in the AUTH payload. Note that if the initiator decides to include the NO_PPK_AUTH notification, the initiator needs to perform authentication data computation twice, which may consume computation power (e.g., if digital signatures are involved).

When the responder receives this encrypted exchange, it first computes the values:

$$\begin{aligned} \text{SKEYSEED} &= \text{prf}(\text{Ni} \parallel \text{Nr}, g^{\text{air}}) \\ \{ \text{SK_d'} \parallel \text{SK_ai} \parallel \text{SK_ar} \parallel \text{SK_ei} \parallel \text{SK_er} \parallel \text{SK_pi'} \parallel \text{SK_pr'} \} \\ &= \text{prf+}(\text{SKEYSEED}, \text{Ni} \parallel \text{Nr} \parallel \text{SPIi} \parallel \text{SPIr}) \end{aligned}$$

The responder then uses the SK_ei/SK_ai values to decrypt/check the message and then scans through the payloads for the PPK_ID attached to the PPK_IDENTITY notification. If no PPK_IDENTITY notification is found and the peers successfully exchanged USE_PPK notifications in the IKE_SA_INIT exchange, then the responder MUST send back AUTHENTICATION_FAILED notification and then fail the negotiation.

If the PPK_IDENTITY notification contains a PPK_ID that is not known to the responder or is not configured for use for the identity from IDi payload, then the responder checks whether using PPKs for this initiator is mandatory and whether the initiator included NO_PPK_AUTH notification in the message. If using PPKs is mandatory or no NO_PPK_AUTH notification is found, then the responder MUST send back AUTHENTICATION_FAILED notification and then fail the negotiation. Otherwise (when PPK is optional and the initiator included NO_PPK_AUTH notification) the responder MAY continue regular IKEv2 protocol, except that it uses the data from the NO_PPK_AUTH notification as the authentication data (which usually resides in the AUTH payload), for the purpose of the initiator authentication.

Note, that Authentication Method is still indicated in the AUTH payload.

This table summarizes the above logic for the responder:

Received USE_PPK	Received NO_PPK_AUTH	Configured with PPK	PPK is Mandatory	Action
No	*	No	*	Standard IKEv2 protocol
No	*	Yes	No	Standard IKEv2 protocol
No	*	Yes	Yes	Abort negotiation
Yes	No	No	*	Abort negotiation
Yes	Yes	No	Yes	Abort negotiation
Yes	Yes	No	No	Standard IKEv2 protocol
Yes	*	Yes	*	Use PPK

If PPK is in use, then the responder extracts the corresponding PPK and computes the following values:

```
SK_d  = prf+ (PPK, SK_d')
SK_pi = prf+ (PPK, SK_pi')
SK_pr = prf+ (PPK, SK_pr')
```

The responder then continues with the IKE_AUTH exchange (validating the AUTH payload that the initiator included) as usual and sends back a response, which includes the PPK_IDENTITY notification with no data to indicate that the PPK is used in the exchange:

Initiator	Responder
	<pre><-- HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr, N(PPK_IDENTITY)}</pre>

When the initiator receives the response, then it checks for the presence of the PPK_IDENTITY notification. If it receives one, it marks the SA as using the configured PPK to generate SK_d, SK_pi, SK_pr (as shown above); the content of the received PPK_IDENTITY (if any) MUST be ignored. If the initiator does not receive the PPK_IDENTITY, it MUST either fail the IKE SA negotiation sending the AUTHENTICATION_FAILED notification in the Informational exchange (if the PPK was configured as mandatory), or continue without using the PPK (if the PPK was not configured as mandatory and the initiator included the NO_PPK_AUTH notification in the request).

If EAP is used in the IKE_AUTH exchange, then the initiator doesn't include AUTH payload in the first request message, however the responder sends back AUTH payload in the first reply. The peers then

exchange AUTH payloads after EAP is successfully completed. As a result, the responder sends AUTH payload twice - in the first IKE_AUTH reply message and in the last one, while the initiator sends AUTH payload only in the last IKE_AUTH request. See more details about EAP authentication in IKEv2 in Section 2.16 of [RFC7296].

The general rule for using PPK in the IKE_AUTH exchange, which covers EAP authentication case too, is that the initiator includes PPK_IDENTITY (and optionally NO_PPK_AUTH) notification in the request message containing AUTH payload. Therefore, in case of EAP the responder always computes the AUTH payload in the first IKE_AUTH reply message without using PPK (by means of SK_{pr'}), since PPK_ID is not yet known to the responder. Once the IKE_AUTH request message containing the PPK_IDENTITY notification is received, the responder follows the rules described above for the non-EAP authentication case.

Initiator	Responder

HDR, SK {IDi, [CERTREQ, [IDr,] SAi2, TSi, TSr]} -->	<-- HDR, SK {IDr, [CERT,] AUTH, EAP}
HDR, SK {EAP} -->	<-- HDR, SK {EAP (success)}
HDR, SK {AUTH, N(PPK_IDENTITY, PPK_ID) [, N(NO_PPK_AUTH)]} -->	<-- HDR, SK {AUTH, SAr2, TSi, TSr [, N(PPK_IDENTITY)]}

Note that the diagram above shows both the cases when the responder uses PPK and when it chooses not to use it (provided the initiator has included NO_PPK_AUTH notification), and thus the responder's PPK_IDENTITY notification is marked as optional. Also, note that the IKE_SA_INIT exchange in case of PPK is as described above (including exchange of the USE_PPK notifications), regardless whether EAP is employed in the IKE_AUTH or not.

4. Upgrade procedure

This algorithm was designed so that someone can introduce PPKs into an existing IKE network without causing network disruption.

In the initial phase of the network upgrade, the network administrator would visit each IKE node, and configure:

- o The set of PPKs (and corresponding PPK_IDs) that this node would need to know.
- o For each peer that this node would initiate to, which PPK will be used.
- o That the use of PPK is currently not mandatory.

With this configuration, the node will continue to operate with nodes that have not yet been upgraded. This is due to the USE_PPK notification and the NO_PPK_AUTH notification; if the initiator has not been upgraded, it will not send the USE_PPK notification (and so the responder will know that the peers will not use a PPK). If the responder has not been upgraded, it will not send the USE_PPK notification (and so the initiator will know to not use a PPK). If both peers have been upgraded, but the responder isn't yet configured with the PPK for the initiator, then the responder could do standard IKEv2 protocol if the initiator sent NO_PPK_AUTH notification. If both the responder and initiator have been upgraded and properly configured, they will both realize it, and the Child SAs will be quantum-secure.

As an optional second step, after all nodes have been upgraded, then the administrator should then go back through the nodes, and mark the use of PPK as mandatory. This will not affect the strength against a passive attacker, but it would mean that an active attacker with a quantum computer (which is sufficiently fast to be able to break the (EC)DH in real-time) would not be able to perform a downgrade attack.

5. PPK

5.1. PPK_ID format

This standard requires that both the initiator and the responder have a secret PPK value, with the responder selecting the PPK based on the PPK_ID that the initiator sends. In this standard, both the initiator and the responder are configured with fixed PPK and PPK_ID values, and do the look up based on PPK_ID value. It is anticipated that later specifications will extend this technique to allow dynamically changing PPK values. To facilitate such an extension, we specify that the PPK_ID the initiator sends will have its first octet be the PPK_ID Type value. This document defines two values for PPK_ID Type:

- o PPK_ID_OPAQUE (1) - for this type the format of the PPK_ID (and the PPK itself) is not specified by this document; it is assumed to be mutually intelligible by both by initiator and the

responder. This PPK_ID type is intended for those implementations that choose not to disclose the type of PPK to active attackers.

- o PPK_ID_FIXED (2) - in this case the format of the PPK_ID and the PPK are fixed octet strings; the remaining bytes of the PPK_ID are a configured value. We assume that there is a fixed mapping between PPK_ID and PPK, which is configured locally to both the initiator and the responder. The responder can use the PPK_ID to look up the corresponding PPK value. Not all implementations are able to configure arbitrary octet strings; to improve the potential interoperability, it is recommended that, in the PPK_ID_FIXED case, both the PPK and the PPK_ID strings be limited to the Base64 character set [RFC4648].

5.2. Operational Considerations

The need to maintain several independent sets of security credentials can significantly complicate a security administrator's job, and can potentially slow down widespread adoption of this specification. It is anticipated, that administrators will try to simplify their job by decreasing the number of credentials they need to maintain. This section describes some of the considerations for PPK management.

5.2.1. PPK Distribution

PPK_IDs of the type PPK_ID_FIXED (and the corresponding PPKs) are assumed to be configured within the IKE device in an out-of-band fashion. While the method of distribution is a local matter and out of scope of this document or IKEv2, [RFC6030] describes a format for the transport and provisioning of symmetric keys. That format could be reused using the PIN profile (defined in Section 10.2 of [RFC6030]) with the "Id" attribute of the <Key> element being the PPK_ID (without the PPK_ID Type octet for a PPK_ID_FIXED) and the <Secret> element containing the PPK.

5.2.2. Group PPK

This document doesn't explicitly require that PPK is unique for each pair of peers. If it is the case, then this solution provides full peer authentication, but it also means that each host must have as many independent PPKs as the peers it is going to communicate with. As the number of peers grows the PPKs will not scale.

It is possible to use a single PPK for a group of users. Since each peer uses classical public key cryptography in addition to PPK for key exchange and authentication, members of the group can neither impersonate each other nor read other's traffic, unless they use quantum computers to break public key operations. However group

members can record any traffic they have access to that comes from other group members and decrypt it later, when they get access to a quantum computer.

In addition, the fact that the PPK is known to a (potentially large) group of users makes it more susceptible to theft. When an attacker equipped with a quantum computer gets access to a group PPK, all communications inside the group are revealed.

For these reasons using group PPK is NOT RECOMMENDED.

5.2.3. PPK-only Authentication

If quantum computers become a reality, classical public key cryptography will provide little security, so administrators may find it attractive not to use it at all for authentication. This will reduce the number of credentials they need to maintain to PPKs only. Combining group PPK and PPK-only authentication is NOT RECOMMENDED, since in this case any member of the group can impersonate any other member even without help of quantum computers.

PPK-only authentication can be achieved in IKEv2 if the NULL Authentication method [RFC7619] is employed. Without PPK the NULL Authentication method provides no authentication of the peers, however since a PPK is stirred into the SK_pi and the SK_pr, the peers become authenticated if a PPK is in use. Using PPKs MUST be mandatory for the peers if they advertise support for PPK in IKE_SA_INIT and use NULL Authentication. Additionally, since the peers are authenticated via PPK, the ID Type in the IDi/IDr payloads SHOULD NOT be ID_NULL, despite using the NULL Authentication method.

6. Security Considerations

Quantum computers are able to perform Grover's algorithm [GROVER]; that effectively halves the size of a symmetric key. Because of this, the user SHOULD ensure that the post-quantum preshared key used has at least 256 bits of entropy, in order to provide 128 bits of post-quantum security. That provides security equivalent to Level 5 as defined in the NIST PQ Project Call For Proposals [NISTPQCFP].

With this protocol, the computed SK_d is a function of the PPK. Assuming that the PPK has sufficient entropy (for example, at least 2^{256} possible values), then even if an attacker was able to recover the rest of the inputs to the PRF function, it would be infeasible to use Grover's algorithm with a quantum computer to recover the SK_d value. Similarly, all keys that are a function of SK_d, which include all Child SAs keys and all keys for subsequent IKE SAs (created when the initial IKE SA is rekeyed), are also quantum-secure

(assuming that the PPK was of high enough entropy, and that all the subkeys are sufficiently long).

An attacker with a quantum computer that can decrypt the initial IKE SA has access to all the information exchanged over it, such as identities of the peers, configuration parameters and all negotiated IPsec SAs information (including traffic selectors), with the exception of the cryptographic keys used by the IPsec SAs which are protected by the PPK.

Deployments that treat this information as sensitive or that send other sensitive data (like cryptographic keys) over IKE SA MUST rekey the IKE SA before the sensitive information is sent to ensure this information is protected by the PPK. It is possible to create a childless IKE SA as specified in [RFC6023]. This prevents Child SA configuration information from being transmitted in the original IKE SA that is not protected by a PPK. Some information related to IKE SA, that is sent in the IKE_AUTH exchange, such as peer identities, feature notifications, Vendor ID's etc. cannot be hidden from the attack described above, even if the additional IKE SA rekey is performed.

In addition, the policy SHOULD be set to negotiate only quantum-secure symmetric algorithms; while this RFC doesn't claim to give advice as to what algorithms are secure (as that may change based on future cryptographical results), below is a list of defined IKEv2 and IPsec algorithms that should not be used, as they are known to provide less than 128 bits of post-quantum security

- o Any IKEv2 Encryption algorithm, PRF or Integrity algorithm with key size less than 256 bits.
- o Any ESP Transform with key size less than 256 bits.
- o PRF_AES128_XCBC and PRF_AES128_CBC; even though they are defined to be able to use an arbitrary key size, they convert it into a 128-bit key internally.

Section 3 requires the initiator to abort the initial exchange if using PPKs is mandatory for it, but the responder does not include the USE_PPK notification in the response. In this situation, when the initiator aborts negotiation it leaves a half-open IKE SA on the responder (because IKE_SA_INIT completes successfully from the responder's point of view). This half-open SA will eventually expire and be deleted, but if the initiator continues its attempts to create IKE SA with a high enough rate, then the responder may consider it as a Denial-of-Service (DoS) attack and take protection measures (see [RFC8019] for more detail). In this situation, it is RECOMMENDED

that the initiator caches the negative result of the negotiation and doesn't make attempts to create it again for some time. This period of time may vary, but it is believed that waiting for at least few minutes will not cause the responder to treat it as DoS attack. Note, that this situation would most likely be a result of misconfiguration and some re-configuration of the peers would probably be needed.

If using PPKs is optional for both peers and they authenticate themselves using digital signatures, then an attacker in between, equipped with a quantum computer capable of breaking public key operations in real time, is able to mount downgrade attack by removing USE_PPK notification from the IKE_SA_INIT and forging digital signatures in the subsequent exchange. If using PPKs is mandatory for at least one of the peers or PSK is used for authentication, then the attack will be detected and the SA won't be created.

If using PPKs is mandatory for the initiator, then an attacker able to eavesdrop and to inject packets into the network can prevent creating an IKE SA by mounting the following attack. The attacker intercepts the initial request containing the USE_PPK notification and injects a forged response containing no USE_PPK. If the attacker manages to inject this packet before the responder sends a genuine response, then the initiator would abort the exchange. To thwart this kind of attack it is RECOMMENDED, that if using PPKs is mandatory for the initiator and the received response doesn't contain the USE_PPK notification, then the initiator doesn't abort the exchange immediately. Instead it waits for more response messages retransmitting the request as if no responses were received at all, until either the received message contains the USE_PPK or the exchange times out (see section 2.4 of [RFC7296] for more details about retransmission timers in IKEv2). If neither of the received responses contains USE_PPK, then the exchange is aborted.

If using PPK is optional for both peers, then in case of misconfiguration (e.g., mismatched PPK_ID) the IKE SA will be created without protection against quantum computers. It is advised that if PPK was configured, but was not used for a particular IKE SA, then implementations SHOULD audit this event.

7. IANA Considerations

This document defines three new Notify Message Types in the "Notify Message Types - Status Types" registry (<https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-16>):

16435	USE_PPK	[THIS RFC]
16436	PPK_IDENTITY	[THIS RFC]
16437	NO_PPK_AUTH	[THIS RFC]

This document also creates a new IANA registry "IKEv2 Post-quantum Preshared Key ID Types" in IKEv2 IANA registry (<https://www.iana.org/assignments/ikev2-parameters/>) for the PPK_ID types used in the PPK_IDENTITY notification defined in this specification. The initial values of the new registry are:

PPK_ID Type	Value	Reference
-----	-----	-----
Reserved	0	[THIS RFC]
PPK_ID_OPAQUE	1	[THIS RFC]
PPK_ID_FIXED	2	[THIS RFC]
Unassigned	3-127	[THIS RFC]
Private Use	128-255	[THIS RFC]

The PPK_ID type value 0 is reserved; values 3-127 are to be assigned by IANA; values 128-255 are for private use among mutually consenting parties. To register new PPK_IDs in the unassigned range, a Type name, a Value between 3 and 127 and a Reference specification need to be defined. Changes and additions to the unassigned range of this registry are by the Expert Review Policy [RFC8126]. Changes and additions to the private use range of this registry are by the Private Use Policy [RFC8126].

8. References

8.1. Normative References

- [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>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informational References

- [GROVER] Grover, L., "A Fast Quantum Mechanical Algorithm for Database Search", Proc. of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996), 1996.
- [I-D.hoffman-c2pq] Hoffman, P., "The Transition from Classical to Post-Quantum Cryptography", draft-hoffman-c2pq-06 (work in progress), November 2019.
- [IKEV2-IANA-PRFS] "Internet Key Exchange Version 2 (IKEv2) Parameters, Transform Type 2 - Pseudorandom Function Transform IDs", <<https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-6>>.
- [NISTPQCFP] NIST, "NIST Post-Quantum Cryptography Call for Proposals", 2016, <<https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>>.
- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <<https://www.rfc-editor.org/info/rfc2409>>.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <<https://www.rfc-editor.org/info/rfc4648>>.
- [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, DOI 10.17487/RFC6023, October 2010, <<https://www.rfc-editor.org/info/rfc6023>>.
- [RFC6030] Hoyer, P., Pei, M., and S. Machani, "Portable Symmetric Key Container (PSKC)", RFC 6030, DOI 10.17487/RFC6030, October 2010, <<https://www.rfc-editor.org/info/rfc6030>>.
- [RFC7619] Smyslov, V. and P. Wouters, "The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7619, DOI 10.17487/RFC7619, August 2015, <<https://www.rfc-editor.org/info/rfc7619>>.

- [RFC8019] Nir, Y. and V. Smyslov, "Protecting Internet Key Exchange Protocol Version 2 (IKEv2) Implementations from Distributed Denial-of-Service Attacks", RFC 8019, DOI 10.17487/RFC8019, November 2016, <<https://www.rfc-editor.org/info/rfc8019>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

Appendix A. Discussion and Rationale

The idea behind this document is that while a quantum computer can easily reconstruct the shared secret of an (EC)DH exchange, they cannot as easily recover a secret from a symmetric exchange. This document makes the SK_d, and hence the IPsec KEYMAT and any child SA's SKEYSEED, depend on both the symmetric PPK, and also the Diffie-Hellman exchange. If we assume that the attacker knows everything except the PPK during the key exchange, and there are 2ⁿ plausible PPKs, then a quantum computer (using Grover's algorithm) would take O(2^{n/2}) time to recover the PPK. So, even if the (EC)DH can be trivially solved, the attacker still can't recover any key material (except for the SK_{ei}, SK_{er}, SK_{ai} and SK_{ar} values for the initial IKE exchange) unless they can find the PPK, which is too difficult if the PPK has enough entropy (for example, 256 bits). Note that we do allow an attacker with a quantum computer to rederive the keying material for the initial IKE SA; this was a compromise to allow the responder to select the correct PPK quickly.

Another goal of this protocol is to minimize the number of changes within the IKEv2 protocol, and in particular, within the cryptography of IKEv2. By limiting our changes to notifications, and only adjusting the SK_d, SK_{pi}, SK_{pr}, it is hoped that this would be implementable, even on systems that perform most of the IKEv2 processing in hardware.

A third goal was to be friendly to incremental deployment in operational networks, for which we might not want to have a global shared key, or quantum-secure IKEv2 is rolled out incrementally. This is why we specifically try to allow the PPK to be dependent on the peer, and why we allow the PPK to be configured as optional.

A fourth goal was to avoid violating any of the security properties provided by IKEv2.

Appendix B. Acknowledgements

We would like to thank Tero Kivinen, Paul Wouters, Graham Bartlett, Tommy Pauly, Quynh Dang and the rest of the IPSecME Working Group for their feedback and suggestions for the scheme.

Authors' Addresses

Scott Fluhner
Cisco Systems

Email: sfluhner@cisco.com

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

David McGrew
Cisco Systems

Email: mcgrew@cisco.com

Valery Smyslov
ELVIS-PLUS

Phone: +7 495 276 0211
Email: svan@elvis.ru

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

T. Pauly
Apple Inc.
P. Wouters
Red Hat
March 11, 2019

Split DNS Configuration for IKEv2
draft-ietf-ipsecme-split-dns-17

Abstract

This document defines two Configuration Payload Attribute Types (INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA) for the Internet Key Exchange Protocol Version 2 (IKEv2). These payloads add support for private (internal-only) DNS domains. These domains are intended to be resolved using non-public DNS servers that are only reachable through the IPsec connection. DNS resolution for other domains remains unchanged. These Configuration Payloads only apply to split tunnel configurations.

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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.

Table of Contents

1. Introduction	2
1.1. Requirements Language	3
2. Applicability	3
3. Protocol Exchange	5
3.1. Configuration Request	5
3.2. Configuration Reply	6
3.3. Mapping DNS Servers to Domains	6
3.4. Example Exchanges	6
3.4.1. Simple Case	6
3.4.2. Requesting Domains and DNSSEC trust anchors	7
4. Payload Formats	8
4.1. INTERNAL_DNS_DOMAIN Configuration Attribute Type Request and Reply	8
4.2. INTERNAL_DNSSEC_TA Configuration Attribute	9
5. INTERNAL_DNS_DOMAIN Usage Guidelines	10
6. INTERNAL_DNSSEC_TA Usage Guidelines	11
7. Security Considerations	12
8. IANA Considerations	14
9. References	14
9.1. Normative References	14
9.2. Informative References	15
Authors' Addresses	15

1. Introduction

Split tunnel Virtual Private Network ("VPN") configurations only send packets with a specific destination IP range, usually chosen from [RFC1918], via the VPN. All other traffic is not sent via the VPN. This allows an enterprise deployment to offer Remote Access VPN services without needing to accept and forward all the non-enterprise related network traffic generated by their remote users. Resources within the enterprise can be accessed by the user via the VPN, while all other traffic generated by the user is not sent over the VPN.

These internal resources tend to only have internal-only DNS names and require the use of special internal-only DNS servers to get resolved. Split DNS [RFC2775] is a common configuration that is part of split tunnel VPN configurations to support configuring Remote Access users to use these special internal-only domain names.

The IKEv2 protocol [RFC7296] negotiates configuration parameters using Configuration Payload Attribute Types. This document defines two Configuration Payload Attribute Types that add support for trusted Split DNS domains.

The INTERNAL_DNS_DOMAIN attribute type is used to convey that the specified DNS domain MUST be resolved using the provided DNS nameserver IP addresses as specified in the INTERNAL_IP4_DNS and INTERNAL_IP6_DNS Configuration Payloads, causing these requests to use the IPsec connection.

The INTERNAL_DNSSEC_TA attribute type is used to convey a DNSSEC trust anchor for such a domain. This is required if the external view uses DNSSEC that would prove the internal view does not exist or would expect a different DNSSEC key on the different versions (internal and external) of the enterprise domain.

If an INTERNAL_DNS_DOMAIN is sent by the responder, the responder MUST also include one or more INTERNAL_IP4_DNS or INTERNAL_IP6_DNS attributes that contain the IPv4 or IPv6 address of the internal DNS server.

For the purposes of this document, DNS resolution servers accessible through an IPsec connection will be referred to as "internal DNS servers", and other DNS servers will be referred to as "external DNS servers".

Other tunnel-establishment protocols already support the assignment of Split DNS domains. For example, there are proprietary extensions to IKEv1 that allow a server to assign Split DNS domains to a client. However, the IKEv2 standard does not include a method to configure this option. This document defines a standard way to negotiate this option for IKEv2.

1.1. Requirements Language

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

2. Applicability

If the negotiated IPsec connection is not a split tunnel configuration, the INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA Configuration Payloads MUST be ignored. This prevents generic (non-

enterprise) VPN services from overriding the public DNS hierarchy, which could lead to malicious overrides of DNS and DNSSEC.

Such configurations SHOULD instead use only the INTERNAL_IP4_DNS and INTERNAL_IP6_DNS Configuration Payloads to ensure all of the user's DNS traffic is sent through the IPsec connection and does not leak unencrypted onto the local network, as the local network is often explicitly exempted from IPsec encryption.

For split tunnel configurations, an enterprise can require one or more DNS domains to be resolved via internal DNS servers. This can be a special domain, such as "corp.example.com" for an enterprise that is publicly known to use "example.com". In this case, the remote user needs to be informed what the internal-only domain names are and what the IP addresses of the internal DNS servers are. An enterprise can also run a different version of its public domain on its internal network. In that case, the VPN client is instructed to send DNS queries for the enterprise public domain (eg "example.com") to the internal DNS servers. A configuration for this deployment scenario is referred to as a Split DNS configuration.

Split DNS configurations are often preferable to sending all DNS queries to the enterprise. This allows the remote user to only send DNS queries for the enterprise to the internal DNS servers. The enterprise remains unaware of all non-enterprise (DNS) activity of the user. It also allows the enterprise DNS servers to only be configured for the enterprise DNS domains which removes the legal and technical responsibility of the enterprise to resolve every DNS domain potentially asked for by the remote user.

A client using these configuration payloads will be able to request and receive Split DNS configurations using the INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA configuration attributes. These attributes MUST be accompanied by one or more INTERNAL_IP4_DNS or INTERNAL_IP6_DNS configuration attributes. The client device can then use the internal DNS server(s) for any DNS queries within the assigned domains. DNS queries for other domains SHOULD be sent to the regular DNS service of the client unless it prefers to use the IPsec tunnel for all its DNS queries. For example, the client could trust the IPsec provided DNS servers more than the locally provided DNS servers especially in the case of connecting to unknown or untrusted networks (eg coffee shops or hotel networks). Or the client could prefer the IPsec based DNS servers because those provide additional features over the local DNS servers.

3. Protocol Exchange

In order to negotiate which domains are considered internal to an IKEv2 tunnel, initiators indicate support for Split DNS in their CFG_REQUEST payloads, and responders assign internal domains (and DNSSEC trust anchors) in their CFG_REPLY payloads. When Split DNS has been negotiated, the INTERNAL_IP4_DNS and INTERNAL_IP6_DNS DNS server configuration attributes will be interpreted as internal DNS servers that can resolve hostnames within the internal domains.

3.1. Configuration Request

To indicate support for Split DNS, an initiator includes one or more INTERNAL_DNS_DOMAIN attributes as defined in Section 4 as part of the CFG_REQUEST payload. If an INTERNAL_DNS_DOMAIN attribute is included in the CFG_REQUEST, the initiator MUST also include one or more INTERNAL_IP4_DNS or INTERNAL_IP6_DNS attributes in the CFG_REQUEST.

The INTERNAL_DNS_DOMAIN attribute sent by the initiator is usually empty but MAY contain a suggested domain name.

The absence of INTERNAL_DNS_DOMAIN attributes in the CFG_REQUEST payload indicates that the initiator does not support or is unwilling to accept Split DNS configuration.

To indicate support for receiving DNSSEC trust anchors for Split DNS domains, an initiator includes one or more INTERNAL_DNSSEC_TA attributes as defined in Section 4 as part of the CFG_REQUEST payload. If an INTERNAL_DNSSEC_TA attribute is included in the CFG_REQUEST, the initiator MUST also include one or more INTERNAL_DNS_DOMAIN attributes in the CFG_REQUEST. If the initiator includes an INTERNAL_DNSSEC_TA attribute, but does not include an INTERNAL_DNS_DOMAIN attribute, the responder MAY still respond with both INTERNAL_DNSSEC_TA and INTERNAL_DNS_DOMAIN attributes.

An initiator MAY convey its current DNSSEC trust anchors for the domain specified in the INTERNAL_DNS_DOMAIN attribute. A responder can use this information to determine that it does not need to send a different trust anchor. If the initiator does not wish to convey this information, it MUST use a length of 0.

The absence of INTERNAL_DNSSEC_TA attributes in the CFG_REQUEST payload indicates that the initiator does not support or is unwilling to accept DNSSEC trust anchor configuration.

3.2. Configuration Reply

Responders MAY send one or more INTERNAL_DNS_DOMAIN attributes in their CFG_REPLY payload. If an INTERNAL_DNS_DOMAIN attribute is included in the CFG_REPLY, the responder MUST also include one or both of the INTERNAL_IP4_DNS and INTERNAL_IP6_DNS attributes in the CFG_REPLY. These DNS server configurations are necessary to define which servers can receive queries for hostnames in internal domains. If the CFG_REQUEST included an INTERNAL_DNS_DOMAIN attribute, but the CFG_REPLY does not include an INTERNAL_DNS_DOMAIN attribute, the initiator MUST behave as if Split DNS configurations are not supported by the server, unless the initiator has been configured with local policy to define a set of Split DNS domains to use by default.

Each INTERNAL_DNS_DOMAIN represents a domain that the DNS servers address listed in INTERNAL_IP4_DNS and INTERNAL_IP6_DNS can resolve.

If the CFG_REQUEST included INTERNAL_DNS_DOMAIN attributes with non-zero lengths, the content MAY be ignored or be interpreted as a suggestion by the responder.

For each DNS domain specified in an INTERNAL_DNS_DOMAIN attribute, one or more INTERNAL_DNSSEC_TA attributes MAY be included by the responder. This attribute lists the corresponding internal DNSSEC trust anchor information of a DS record (see [RFC4034]). The INTERNAL_DNSSEC_TA attribute MUST immediately follow the INTERNAL_DNS_DOMAIN attribute that it applies to.

3.3. Mapping DNS Servers to Domains

All DNS servers provided in the CFG_REPLY MUST support resolving hostnames within all INTERNAL_DNS_DOMAIN domains. In other words, the INTERNAL_DNS_DOMAIN attributes in a CFG_REPLY payload form a single list of Split DNS domains that applies to the entire list of INTERNAL_IP4_DNS and INTERNAL_IP6_DNS attributes.

3.4. Example Exchanges

3.4.1. Simple Case

In this example exchange, the initiator requests INTERNAL_IP4_DNS, INTERNAL_IP6_DNS, and INTERNAL_DNS_DOMAIN attributes in the CFG_REQUEST, but does not specify any value for either. This indicates that it supports Split DNS, but has no preference for which DNS requests will be routed through the tunnel.

The responder replies with two DNS server addresses, and two internal domains, "example.com" and "city.other.test".

Any subsequent DNS queries from the initiator for domains such as "www.example.com" SHOULD use 198.51.100.2 or 198.51.100.4 to resolve.

```
CP(CFG_REQUEST) =  
  INTERNAL_IP4_ADDRESS()  
  INTERNAL_IP4_DNS()  
  INTERNAL_IP6_ADDRESS()  
  INTERNAL_IP6_DNS()  
  INTERNAL_DNS_DOMAIN()  
  
CP(CFG_REPLY) =  
  INTERNAL_IP4_ADDRESS(198.51.100.234)  
  INTERNAL_IP4_DNS(198.51.100.2)  
  INTERNAL_IP4_DNS(198.51.100.4)  
  INTERNAL_IP6_ADDRESS(2001:DB8:0:1:2:3:4:5/64)  
  INTERNAL_IP6_DNS(2001:DB8:99:88:77:66:55:44)  
  INTERNAL_DNS_DOMAIN(example.com)  
  INTERNAL_DNS_DOMAIN(city.other.test)
```

3.4.2. Requesting Domains and DNSSEC trust anchors

In this example exchange, the initiator requests INTERNAL_IP4_DNS, INTERNAL_IP6_DNS, INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA attributes in the CFG_REQUEST.

Any subsequent DNS queries from the initiator for domains such as "www.example.com" or "city.other.test" would be DNSSEC validated using the DNSSEC trust anchor received in the CFG_REPLY.

In this example, the initiator has no existing DNSSEC trust anchors would the requested domain. The "example.com" domain has DNSSEC trust anchors that are returned, while the "other.test" domain has no DNSSEC trust anchors.

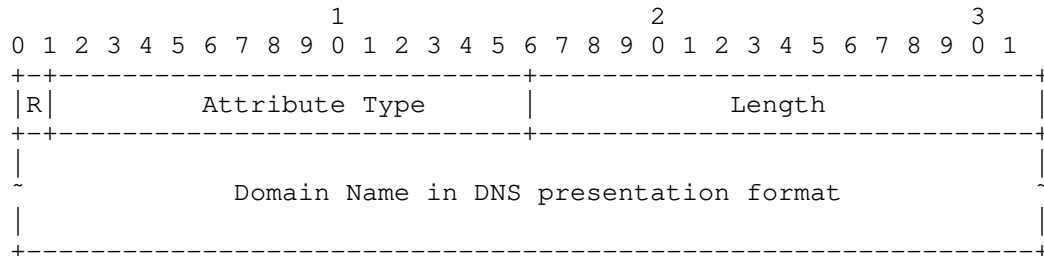
```
CP (CFG_REQUEST) =
INTERNAL_IP4_ADDRESS ()
INTERNAL_IP4_DNS ()
INTERNAL_IP6_ADDRESS ()
INTERNAL_IP6_DNS ()
INTERNAL_DNS_DOMAIN ()
INTERNAL_DNSSEC_TA ()

CP (CFG_REPLY) =
INTERNAL_IP4_ADDRESS (198.51.100.234)
INTERNAL_IP4_DNS (198.51.100.2)
INTERNAL_IP4_DNS (198.51.100.4)
INTERNAL_IP6_ADDRESS (2001:DB8:0:1:2:3:4:5/64)
INTERNAL_IP6_DNS (2001:DB8:99:88:77:66:55:44)
INTERNAL_DNS_DOMAIN (example.com)
INTERNAL_DNSSEC_TA (43547,8,1,B6225AB2CC613E0DCA7962BDC2342EA4...)
INTERNAL_DNSSEC_TA (31406,8,2,F78CF3344F72137235098ECBBD08947C...)
INTERNAL_DNS_DOMAIN (city.other.test)
```

4. Payload Formats

All multi-octet fields representing integers are laid out in big endian order (also known as "most significant byte first", or "network byte order").

4.1. INTERNAL_DNS_DOMAIN Configuration Attribute Type Request and Reply



- o Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- o Attribute Type (15 bits) set to value 25 for INTERNAL_DNS_DOMAIN.
- o Length (2 octets) - Length of domain name.
- o Domain Name (0 or more octets) - A Fully Qualified Domain Name used for Split DNS rules, such as "example.com", in DNS presentation format and using IDNA A-label [RFC5890] for Internationalized Domain Names. Implementors need to be careful that this value is not null-terminated.

4.2. INTERNAL_DNSSEC_TA Configuration Attribute

An `INTERNAL_DNSSEC_TA` Configuration Attribute can either be empty, or it can contain one Trust Anchor by containing a non-zero Length with a DNSKEY Key Tag, DNSKEY Algorithm, Digest Type and Digest Data fields.

An empty INTERNAL_DNSSEC_TA CFG attribute:

										1											2											3
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	
R										Attribute Type										Length (set to 0)												

- o Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- o Attribute Type (15 bits) set to value 26 for INTERNAL_DNSSEC_TA.
- o Length (2 octets) - Set to 0 for an empty attribute.

A non-empty `INTERNAL_DNSSEC_TA` CFG attribute:

1										2										3											
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
R		Attribute Type										Length																			
DNSKEY Key Tag										DNSKEY Alg										Digest Type											
~																															
Digest Data																															

- o Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- o Attribute Type (15 bits) set to value 26 for INTERNAL_DNSSEC_TA.
- o Length (2 octets) - Length of DNSSEC Trust Anchor data (4 octets plus the length of the Digest Data).
- o DNSKEY Key Tag value (2 octets) - Delegation Signer (DS) Key Tag as specified in [RFC4034] Section 5.1.

- o DNSKEY Algorithm (1 octet) - DNSKEY algorithm value from the IANA DNS Security Algorithm Numbers Registry.
- o Digest Type (1 octet) - DS algorithm value from the IANA Delegation Signer (DS) Resource Record (RR) Type Digest Algorithms Registry.
- o Digest Data (1 or more octets) - The DNSKEY digest as specified in [RFC4034] Section 5.1 in presentation format.

Each INTERNAL_DNSSEC_TA attribute in the CFG_REPLY payload MUST immediately follow a corresponding INTERNAL_DNS_DOMAIN attribute. As the INTERNAL_DNSSEC_TA format itself does not contain the domain name, it relies on the preceding INTERNAL_DNS_DOMAIN to provide the domain for which it specifies the trust anchor. Any INTERNAL_DNSSEC_TA attribute that is not immediately preceded by an INTERNAL_DNS_DOMAIN or another INTERNAL_DNSSEC_TA attribute applying to the same domain name MUST be ignored.

5. INTERNAL_DNS_DOMAIN Usage Guidelines

If a CFG_REPLY payload contains no INTERNAL_DNS_DOMAIN attributes, the client MAY use the provided INTERNAL_IP4_DNS or INTERNAL_IP6_DNS servers as the default DNS server(s) for all queries.

If a client is configured by local policy to only accept a limited set of INTERNAL_DNS_DOMAIN values, the client MUST ignore any other INTERNAL_DNS_DOMAIN values.

For each INTERNAL_DNS_DOMAIN entry in a CFG_REPLY payload that is not prohibited by local policy, the client MUST use the provided INTERNAL_IP4_DNS or INTERNAL_IP6_DNS DNS servers as the only resolvers for the listed domains and its sub-domains and it MUST NOT attempt to resolve the provided DNS domains using its external DNS servers. Other domain names SHOULD be resolved using some other external DNS resolver(s), configured independently from IKE. Queries for these other domains MAY be sent to the internal DNS resolver(s) listed in that CFG_REPLY message, but have no guarantee of being answered. For example, if the INTERNAL_DNS_DOMAIN attribute specifies "example.test", then "example.test", "www.example.test" and "mail.eng.example.test" MUST be resolved using the internal DNS resolver(s), but "otherexample.test" and "ple.test" MUST NOT be resolved using the internal resolver and MUST use the system's external DNS resolver(s).

The initiator SHOULD allow the DNS domains listed in the INTERNAL_DNS_DOMAIN attributes to resolve to special IP address ranges, such as those of [RFC1918], even if the initiator host is

otherwise configured to block DNS answer containing these special IP address ranges.

When an IKE SA is terminated, the DNS forwarding MUST be unconfigured. This includes deleting the DNS forwarding rules; flushing all cached data for DNS domains provided by the INTERNAL_DNS_DOMAIN attribute, including negative cache entries; removing any obtained DNSSEC trust anchors from the list of trust anchors; and clearing the outstanding DNS request queue.

INTERNAL_DNS_DOMAIN attributes SHOULD only be used on split tunnel configurations where only a subset of traffic is routed into a private remote network using the IPsec connection. If all traffic is routed over the IPsec connection, the existing global INTERNAL_IP4_DNS and INTERNAL_IP6_DNS can be used without creating specific DNS or DNSSEC exemptions.

6. INTERNAL_DNSSEC_TA Usage Guidelines

DNS records can be used to publish specific records containing trust anchors for applications. The most common record type is the TLSA record specified in [RFC6698]. This DNS record type publishes which Certificate Authority (CA) certificate or End Entity (EE) certificate to expect for a certain host name. These records are protected by DNSSEC and thus are trustable by the application. Whether to trust TLSA records instead of the traditional WebPKI depends on the local policy of the client. By accepting an INTERNAL_DNSSEC_TA trust anchor via IKE from the remote IKE server, the IPsec client might be allowing the remote IKE server to override the trusted certificates for TLS. Similar override concerns apply to other public key or fingerprint-based DNS records, such as OPENPGPKEY, SMIMEA or IPSECKEY records.

Thus, installing an INTERNAL_DNSSEC_TA trust anchor can be seen as the equivalent of installing an Enterprise CA certificate. It allows the remote IKE/IPsec server to modify DNS answers including DNSSEC cryptographic signatures by overriding existing DNS information with trust anchor conveyed via IKE and (temporarily) installed on the IKE client. Of specific concern is the overriding of [RFC6698] based TLSA records, which represent a confirmation or override of an existing WebPKI TLS certificate. Other DNS record types that convey cryptographic materials (public keys or fingerprints) are OPENPGPKEY, SMIMEA, SSHP and IPSECKEY records.

IKE clients willing to accept INTERNAL_DNSSEC_TA attributes MUST use a whitelist of one or more domains that can be updated out of band. IKE clients with an empty whitelist MUST NOT use any INTERNAL_DNSSEC_TA attributes received over IKE. Such clients MAY

interpret receiving an `INTERNAL_DNSSEC_TA` attribute for a non-whitelisted domain as an indication that their local configuration may need to be updated out of band.

IKE clients should take care to only whitelist domains that apply to internal or managed domains, rather than to generic Internet traffic. The DNS root zone (".") MUST be ignored if it appears in a whitelist. Other generic or public domains, such as top-level domains (TLDs), similarly MUST be ignored if these appear in a whitelist unless the entity actually is the operator of the TLD. To determine this, an implementation MAY interactively ask the user when a VPN profile is installed or activated to confirm this. Alternatively, it MAY provide a special override keyword in its provisioning configuration to ensure non-interactive agreement can be achieved only by the party provisioning the VPN client, who presumably is a trusted entity by the end-user. Similarly, an entity might be using a special domain name, such as ".internal", for its internal-only view and might wish to force its provisioning system to accept such a domain in a Split DNS configuration.

Any updates to this whitelist of domain names MUST happen via explicit human interaction or by a trusted automated provision system to prevent malicious invisible installation of trust anchors in case of aIKE server compromise.

IKE clients SHOULD accept any `INTERNAL_DNSSEC_TA` updates for subdomain names of the whitelisted domain names. For example, if "example.net" is whitelisted, then `INTERNAL_DNSSEC_TA` received for "antartica.example.net" SHOULD be accepted.

IKE clients MUST ignore any received `INTERNAL_DNSSEC_TA` attributes for a FQDN for which it did not receive and accept an `INTERNAL_DNS_DOMAIN` Configuration Payload.

In most deployment scenarios, the IKE client has an expectation that it is connecting, using a split-network setup, to a specific organisation or enterprise. A recommended policy would be to only accept `INTERNAL_DNSSEC_TA` directives from that organization's DNS names. However, this might not be possible in all deployment scenarios, such as one where the IKE server is handing out a number of domains that are not within one parent domain.

7. Security Considerations

As stated in Section 2, if the negotiated IPsec connection is not a split tunnel configuration, the `INTERNAL_DNS_DOMAIN` and `INTERNAL_DNSSEC_TA` Configuration Payloads MUST be ignored.

Otherwise, generic VPN service providers could maliciously override DNSSEC based trust anchors of public DNS domains.

An initiator **MUST** only accept `INTERNAL_DNSSEC_TAs` for which it has a whitelist, since this mechanism allows the credential used to authenticate an IKEv2 association to be leveraged into authenticating credentials for other connections. Initiators should ensure that they have sufficient trust in the responder when using this mechanism. An initiator **MAY** treat a received `INTERNAL_DNSSEC_TA` for an non-whitelisted domain as a signal to update the whitelist via a non-IKE provisioning mechanism. See Section 6 for additional security considerations for DNSSEC trust anchors.

The use of Split DNS configurations assigned by an IKEv2 responder is predicated on the trust established during IKE SA authentication. However, if IKEv2 is being negotiated with an anonymous or unknown endpoint (such as for Opportunistic Security [RFC7435]), the initiator **MUST** ignore Split DNS configurations assigned by the responder.

If a host connected to an authenticated IKE peer is connecting to another IKE peer that attempts to claim the same domain via the `INTERNAL_DNS_DOMAIN` attribute, the IKE connection **SHOULD** only process the DNS information if the two connections are part of the same logical entity. Otherwise, the client **SHOULD** refuse the DNS information and potentially warn the end-user. For example, if a VPN profile for "Example Corporation" is installed that provides two IPsec connections, one covering 192.168.100.0/24 and one covering 10.13.14.0/24 it could be that both connections negotiate the same `INTERNAL_DNS_DOMAIN` and `INTERNAL_DNSSEC_TA` values. Since these are part of the same remote organisation (or provisioning profile), the Configuration Payloads can be used. However, if a user installs two VPN profiles from two different unrelated independent entities, both of these could be configured to use the same domain, for example ".internal". These two connections **MUST NOT** be allowed to be active at the same time.

If the initiator is using DNSSEC validation for a domain in its public DNS view, and it requests and receives an `INTERNAL_DNS_DOMAIN` attribute without an `INTERNAL_DNSSEC_TA`, it will need to reconfigure its DNS resolver to allow for an insecure delegation. It **SHOULD NOT** accept insecure delegations for domains that are DNSSEC signed in the public DNS view, for which it has not explicitly requested such delegation by specifying the domain specifically using a `INTERNAL_DNS_DOMAIN` request.

Deployments that configure `INTERNAL_DNS_DOMAIN` domains should pay close attention to their use of indirect reference RRTypes in their

internal-only domain names. Examples of such RRtypes are NS, CNAME, DNAME, MX or SRV records. For example, if the MX record for "internal.example.com" points to "mx.internal.example.net", then both "internal.example.com" and "internal.example.net" should be sent using an INTERNAL_DNS_DOMAIN Configuration Payload.

IKE clients MAY want to require whitelisted domains for Top Level Domains (TLDs) and Second Level Domains (SLDs) to further prevent malicious DNS redirections for well known domains. This prevents users from unknowingly giving DNS queries to third parties. This is even more important if those well known domains are not deploying DNSSEC, as the VPN service provider could then even modify the DNS answers without detection.

The content of INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA may be passed to another (DNS) program for processing. As with any network input, the content SHOULD be considered untrusted and handled accordingly.

8. IANA Considerations

This document defines two new IKEv2 Configuration Payload Attribute Types, which are allocated from the "IKEv2 Configuration Payload Attribute Types" namespace.

Value	Attribute Type	Multi-Valued	Length	Reference
25	INTERNAL_DNS_DOMAIN	YES	0 or more	[this document]
26	INTERNAL_DNSSEC_TA	YES	0 or more	[this document]

Figure 1

9. References

9.1. Normative References

- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [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>>.

- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <<https://www.rfc-editor.org/info/rfc5890>>.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, DOI 10.17487/RFC2775, February 2000, <<https://www.rfc-editor.org/info/rfc2775>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.

Authors' Addresses

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014
US

Email: tpauly@apple.com

Paul Wouters
Red Hat

Email: pwouters@redhat.com

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

D. Migault
Ericsson
T. Guggemos
LMU Munich
C. Bormann
Universitaet Bremen TZI
D. Schinazi
Google LLC
March 11, 2019

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

Abstract

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

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

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

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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.

Table of Contents

1. Requirements notation	3
2. Introduction	3
3. Terminology	4
4. Protocol Overview	4
5. IPsec Compression Mode	7
6. EHC Context	7
6.1. EHC Context Parameters for ESP	8
6.2. EHC Context Parameters for Inner IP	8
6.3. EHC Context Parameters for Transport Protocol	9
7. EHC Rules	11
7.1. EHC Rules for ESP	13
7.2. EHC Rules for inner IPv4	15
7.3. EHC Rules for inner IPv6	17
7.4. EHC Rules for UDP	18
7.5. EHC Rules for UDP-Lite	19
7.6. EHC Rules for TCP	20
8. Diet-ESP EHC Strategy	21
8.1. Outbound Packet Processing	24
8.2. Inbound Packet Processing	26
9. IANA Considerations	29
10. Security Considerations	29
11. Privacy Considerations	30
12. Acknowledgment	31
13. References	31
13.1. Normative References	31
13.2. Informational References	32

Appendix A. Illustrative Examples	33
A.1. Single UDP Session IoT VPN	33
A.2. Single TCP session IoT VPN	36
A.3. Traditional VPN	39
Authors' Addresses	46

1. Requirements notation

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

2. Introduction

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

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

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

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

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

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

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

3. Terminology

This document uses the following terminology:

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

4. Protocol Overview

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

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

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

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

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

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

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

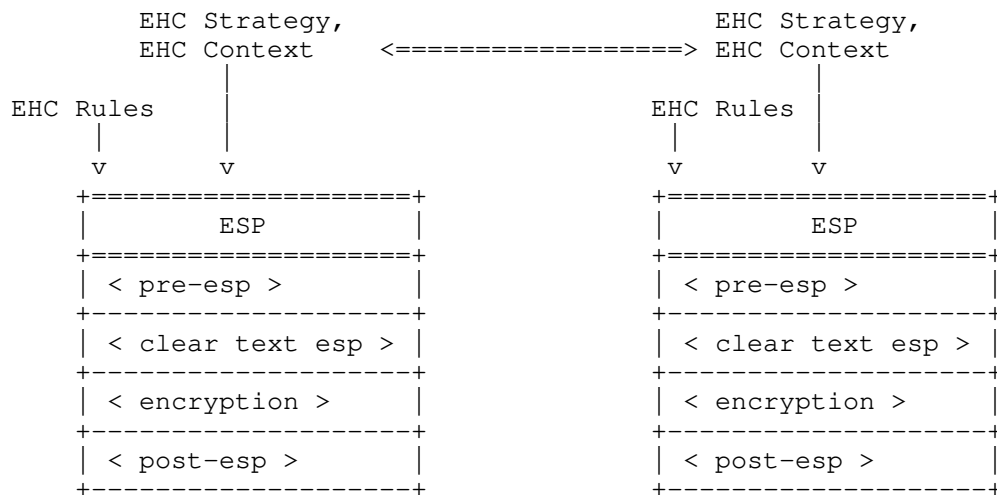


Figure 1: ESP Header Compression Overview

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

- pre-esp: represents treatment performed to a non ESP packet, i.e. before ESP encapsulation or decapsulation is being performed. Any compression of protocols not specific to but encrypted by ESP, such as L4 and higher protocols, is performed here.
- clear text esp: designates the ESP encapsulation / decapsulation processing performed on an non encrypted ESP packet. This layer includes compression for fields which are included during the ESP encapsulation. A typical example is the later encrypted Tunnel IP header and the fields of the ESP trailer.
- encryption: designates the encryption/decryption phase This layer could include compression of encryption information (e.g. Initialization Vector, etc.), but this is currently out of scope of this document.
- post-esp the processing performed on an ESP encrypted packet. This layer includes compression of the ESP header.

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

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

5. IPsec Compression Mode

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

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

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

6. EHC Context

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

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

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

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

6.1. EHC Context Parameters for ESP

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

6.2. EHC Context Parameters for Inner IP

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

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

6.2.1. EHC Context Parameters for inner IPv6

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

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

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

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

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

6.2.2. EHC Context Parameters for inner IPv4

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

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

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

6.3. EHC Context Parameters for Transport Protocol

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

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

6.3.1. EHC Context Parameters for UDP

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

6.3.2. EHC Context Parameters for UDP-Lite

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

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

6.3.3. EHC Context Parameters for TCP

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

tcp_sn holds the current Sequence Number of the TCP session.

tcp_ack holds the current Acknowledgement Number of the TCP session.

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

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

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

7. EHC Rules

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

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

EHC Rule	Field	Action	Parameters
EHC_RULE_NAME	f1	a1	p1_1, ... p1_n

	fm	am	pm_1, ... pm_n

Figure 2: EHC Rules

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

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

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

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

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

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

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

7.1. EHC Rules for ESP

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

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

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

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

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

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

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

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

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

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

7.2. EHC Rules for inner IPv4

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

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

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

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

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

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

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

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

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

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

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

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

7.3. EHC Rules for inner IPv6

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

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

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

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

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

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

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

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

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

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

7.4. EHC Rules for UDP

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

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

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

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

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

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

7.5. EHC Rules for UDP-Lite

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

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

UDP-LITE_SRC works similarly to UDP_SRC

UDP-LITE_DST works similarly to UDP_DST

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

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

7.6. EHC Rules for TCP

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

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

TCP_SRC works similarly to UDP_SRC.

TCP_DST works similarly to UDP_DST.

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

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

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

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

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

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

8. Diet-ESP EHC Strategy

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

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

- o Diet-ESP favors compression methods that required fewer configuration: For IPv6, ip6_tcfl_comp and ip6_hl_com to "Outer" so that ip6_tc, ip6_fl and ip6_hl can be derived from the packet. Similarly, ip4_ttl_comp has is set to "Outer" so ip4_ttl can be derived from the packet.
- o Diet-ESP limits compression method to those foreseen as the most commonly used. As such, esp_sn_gen has been set to "Incremental" as this is the most common method used to generate SN. The other method would be "Time".

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

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

ESP:

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

IPv4:

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

IPv6:

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

UDP:

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

UDP-Lite:

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

TCP:

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

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

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

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

8.1. Outbound Packet Processing

Diet-ESP compression is defined as follows:

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

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

UDP compression is defined as below:

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

UDP-lite compression is defined as below:

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

TCP compression is defined as below:

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

Inner IPv6 Header compression is defined as below:

1. If "ip6_src" designates a "Single" Source IP address, apply the IP6_SRC to compress the IPv6 Source Address.
2. If "ip6_dst" designates a "Single" Destination IP address, apply the IP6_DST to decompress the IPv6 Destination Address.
3. Apply IP6_HL_OUTER to compress the Hop Limit.
4. If "l4_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP6_NH to compress the Next Header.
5. Apply, IP6_LENGTH to compress the Length.
6. Apply IP6_OUTER to compress Version, Traffic Class and Flow Label.

Inner IPv4 Header compression is defined as below:

1. Apply, IP4_LENGTH to compress the Length.
2. Apply IP4_TTL_OUTER to compress Time To Live.
3. Apply, IP4_CHECK to compress the IPv4 header checksum.
4. If "ip4_src" designates a "Single" Source IP address, apply the IP4_SRC to compress the IPv4 Source Address.
5. If "ip4_dst" designates a "Single" Destination IP address, apply the IP4_DST to decompress the IPv4 Destination Address.

ESP compression is defined as below:

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

8.2. Inbound Packet Processing

Diet-ESP decompression is defined as follows:

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

addresses and proceed to the appropriated inner IP address decompression, except for the computation of the checksums and length.

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

ESP decompression is defined as follows:

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

UDP decompression is defined as follows:

1. If "l4_src" designates a "Single" Source Port, apply UDP_SRC to decompress the Source Port.
2. If "l4_dst" designates a "Single" Destination Port, apply UDP_DST to decompress the Destination Port.
3. Apply UDP_LENGTH to compress the Length. The length value is computed from the length provided by the lower layer, with the additional added bytes during the UDP decompression including the length size.
4. Apply UDP_CHECK to decompress the Checksum.
5. Update the Length of the lower layers:
 1. If "ipsec_mode" is set to "Transport" mode, update the Length of the outer IP header (IPv4 or IPv6). The Length is incremented by the number of bytes generated by the decompression of the transport layer.
 2. If "ipsec_mode" is set to "Tunnel" mode, update the Length of the inner IP address (IPv4 or IPv6) as well as the outer IP header (IPv4 or IPv6). The Length is incremented by the

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

UDP-Lite decompression is defined as follows:

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

TCP decompression is defined as follows:

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

Inner IPv6 decompression is defined as follows:

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

Inner IPv4 decompression is defined as follows:

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

9. IANA Considerations

There are no IANA consideration for this document.

10. Security Considerations

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

Security Parameter Index (SPI):

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

Sequence Number (SN):

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

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

11. Privacy Considerations

Security Parameter Index (SPI):

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

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

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

12. Acknowledgment

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

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

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

13. References

13.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <<https://www.rfc-editor.org/info/rfc4309>>.

- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, DOI 10.17487/RFC5795, March 2010, <<https://www.rfc-editor.org/info/rfc5795>>.
- [RFC5858] Ertekin, E., Christou, C., and C. Bormann, "IPsec Extensions to Support Robust Header Compression over IPsec", RFC 5858, DOI 10.17487/RFC5858, May 2010, <<https://www.rfc-editor.org/info/rfc5858>>.
- [RFC7400] Bormann, C., "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 7400, DOI 10.17487/RFC7400, November 2014, <<https://www.rfc-editor.org/info/rfc7400>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

13.2. Informational References

- [I-D.toutain-6lpwa-ipv6-static-context-hc]
Minaburo, A. and L. Toutain, "6LPWA Static Context Header Compression (SCHC) for IPV6 and UDP", draft-toutain-6lpwa-ipv6-static-context-hc-01 (work in progress), June 2016.
- [I-D.mglt-ipsecme-implicit-iv]
Migault, D., Guggemos, T., and Y. Nir, "Implicit IV for Counter-based Ciphers in IPsec", draft-mglt-ipsecme-implicit-iv-04 (work in progress), June 2017.
- [I-D.ietf-tsvwg-udp-options]
Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-07 (work in progress), March 2019.
- [fit-iot-lab]
"Future Internet of Things (FIT) IoT-LAB", <<https://www.iot-lab.info>>.

Appendix A. Illustrative Examples

A.1. Single UDP Session IoT VPN

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

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

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

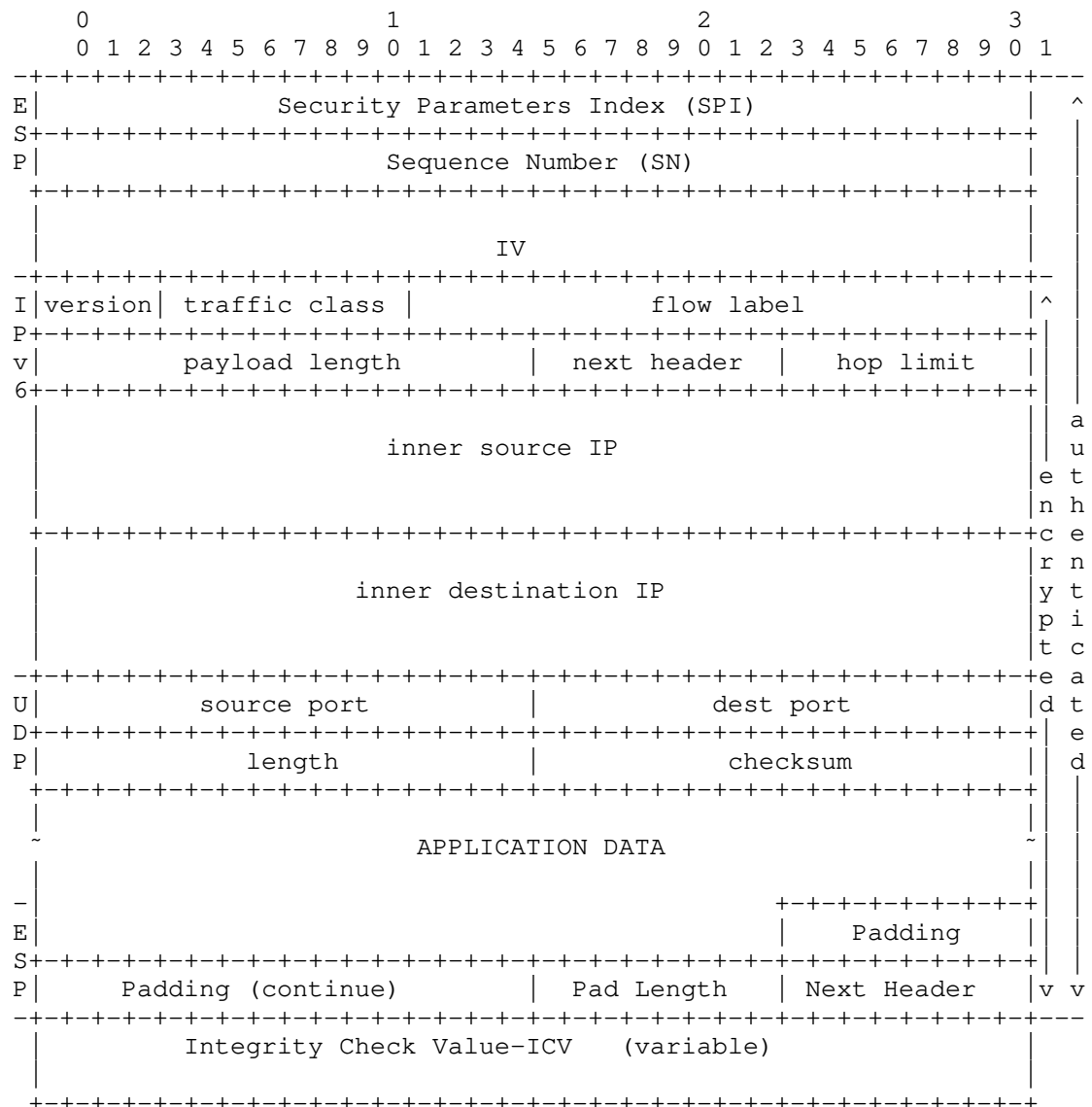


Figure 3: Standard ESP VPN Packet Description

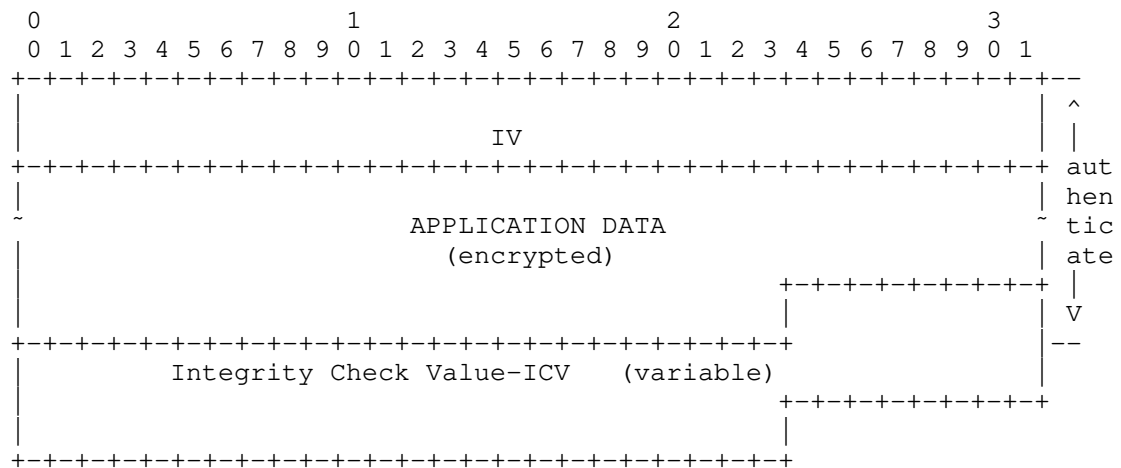


Figure 4: Diet-ESP Single UDP Session IoT VPN Packet Description

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

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

A.2. Single TCP session IoT VPN

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

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

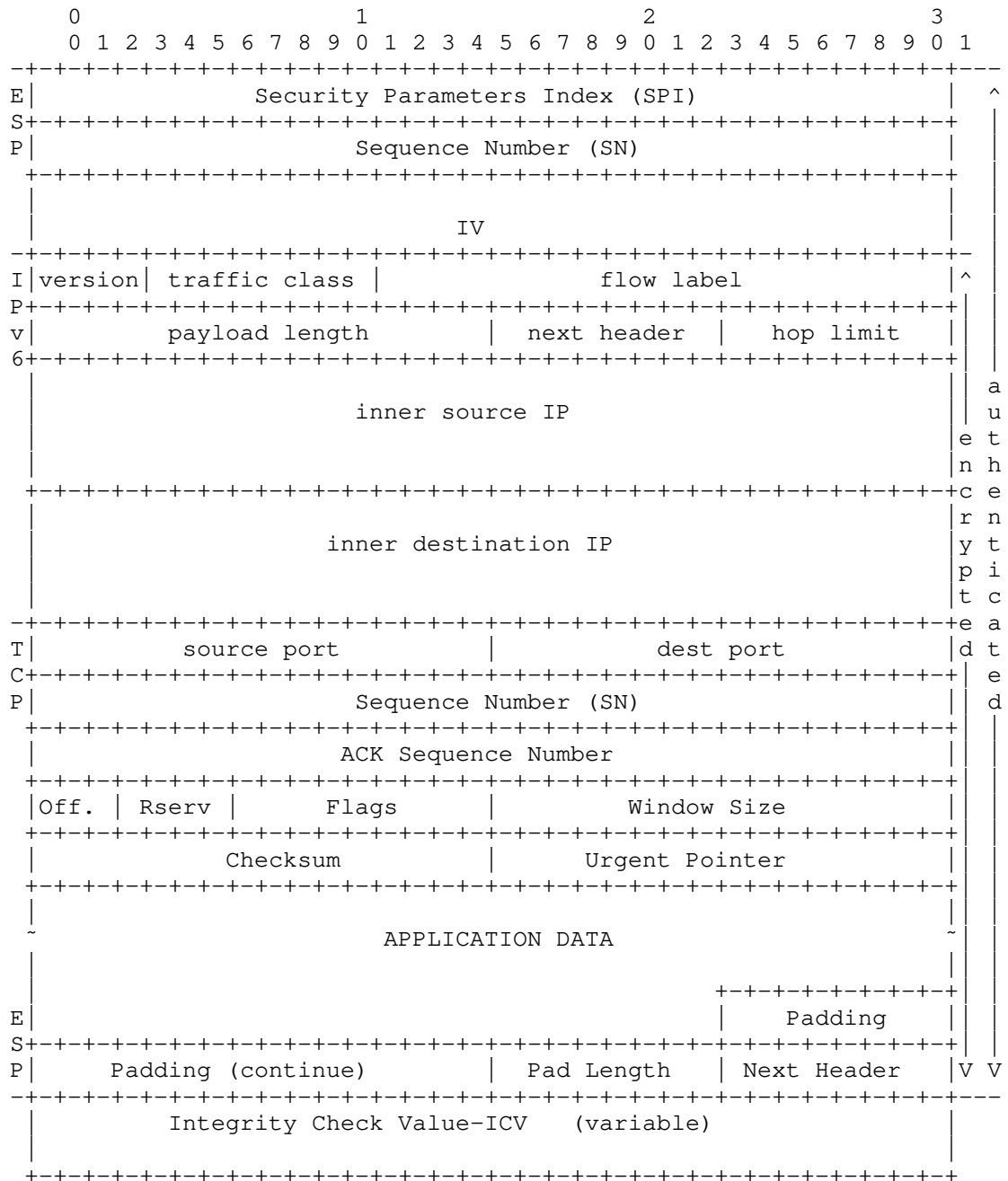


Figure 5: Standard IoT Single TCP Session VPN Packet Description

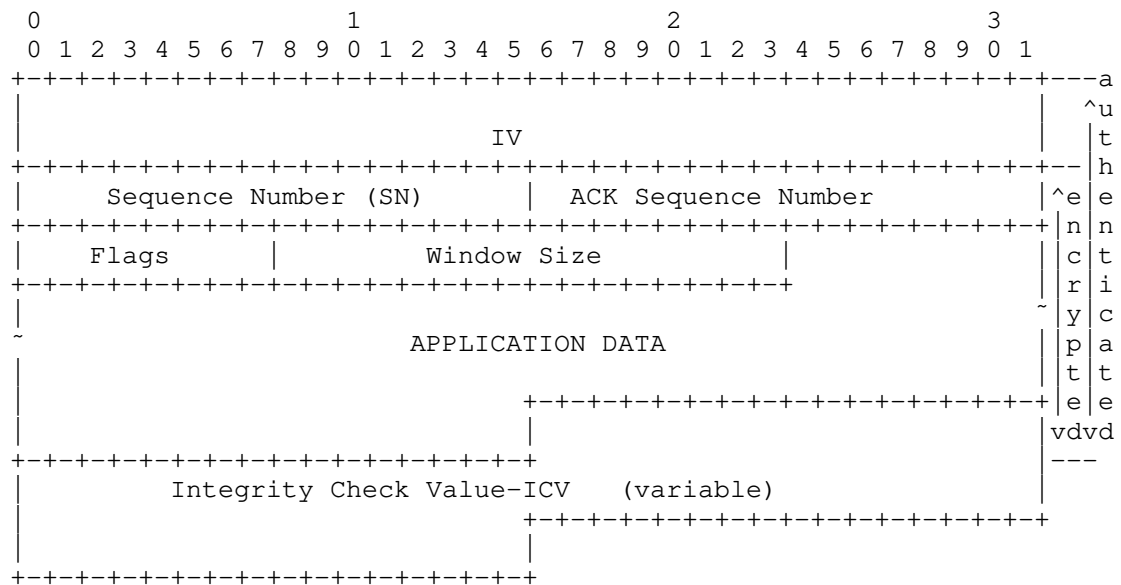


Figure 6: Diet-ESP Single TCP Session IoT VPN Packet Description

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

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

A.3. Traditional VPN

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

A.3.1. IPv6 in IPv6

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

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

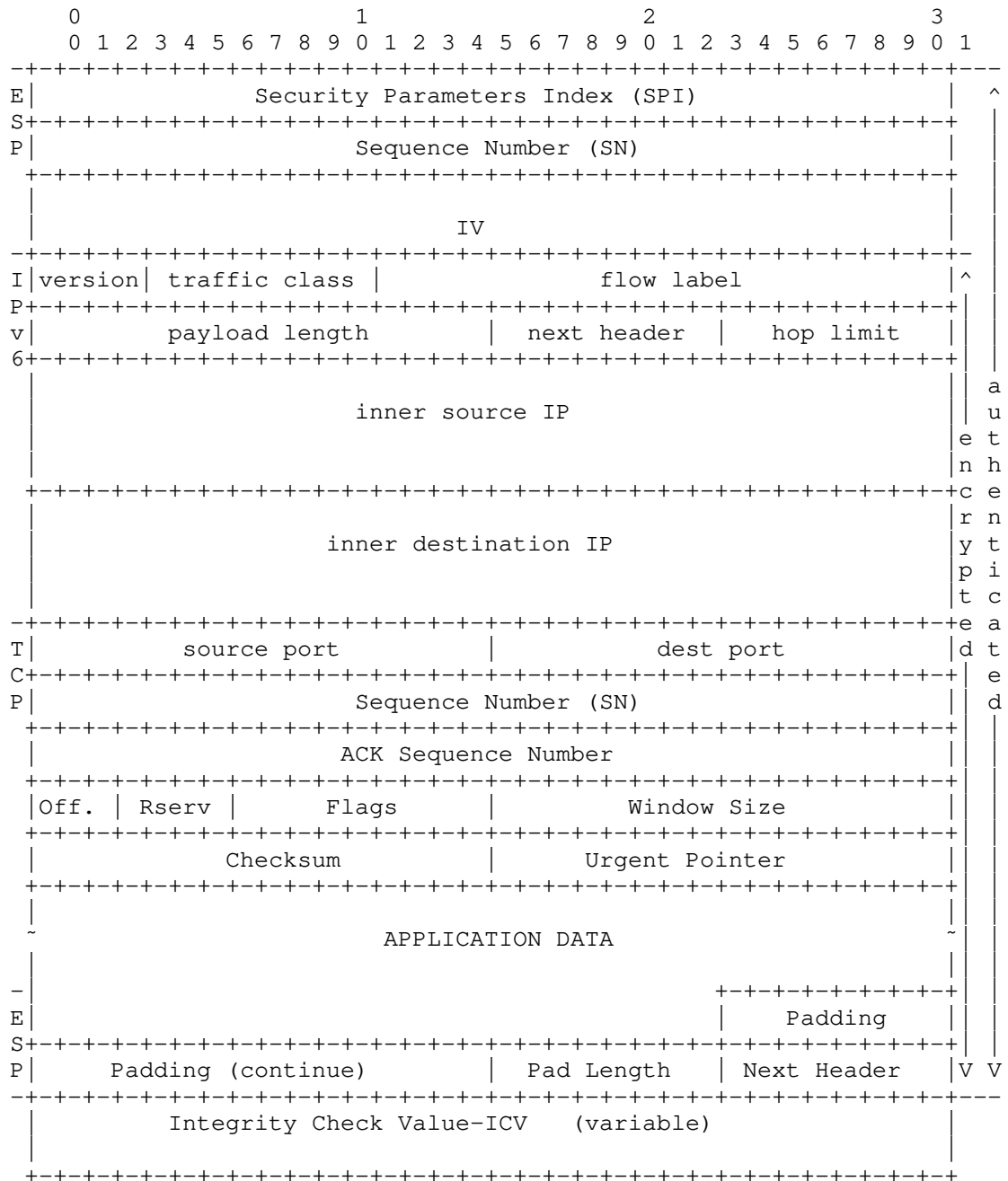


Figure 7: Standard ESP VPN Packet Description

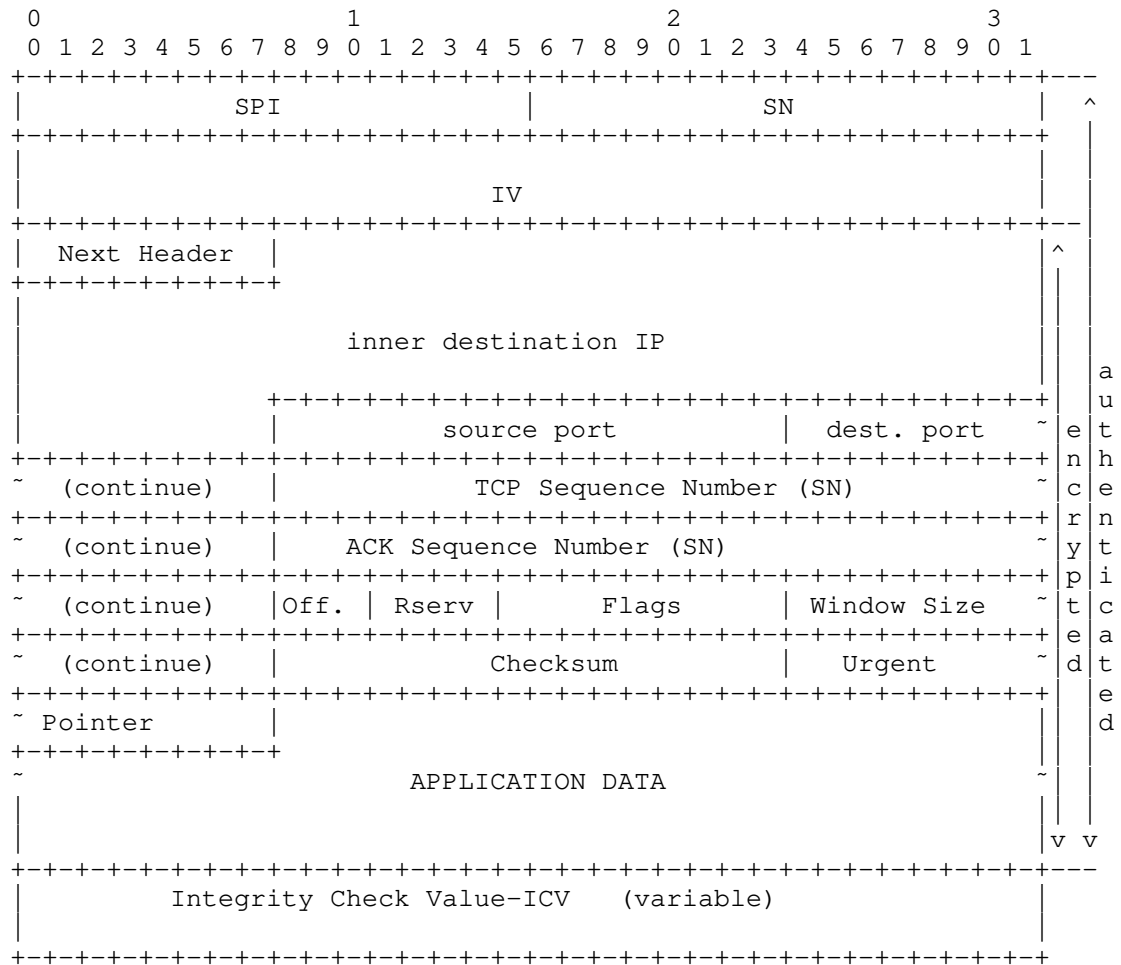


Figure 8: Diet-ESP VPN Packet Description

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

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

A.3.2. IPv6 in IPv4

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

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

A.3.3. IPv4 in IPv4

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

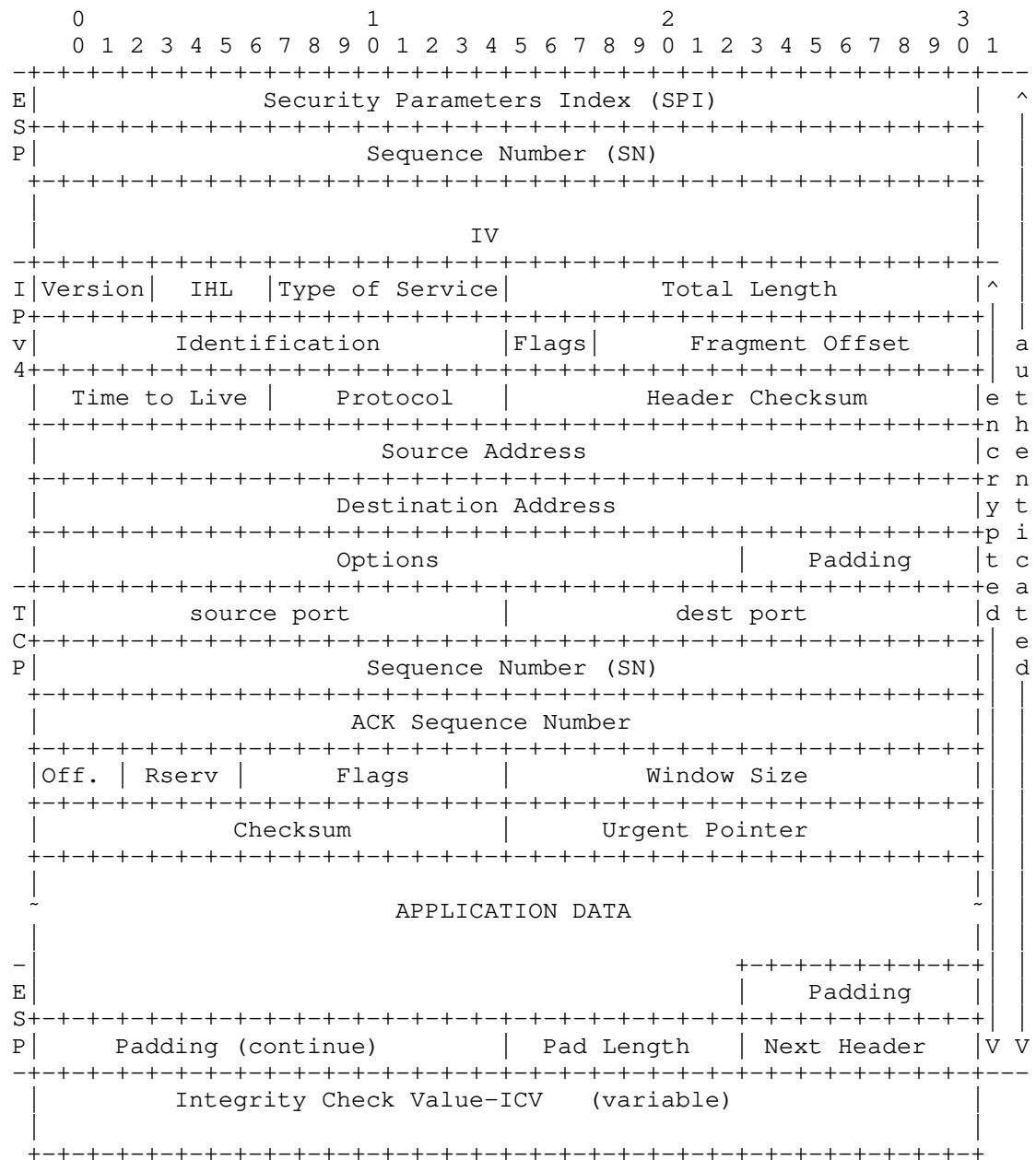


Figure 9: Standard ESP VPN Packet Description

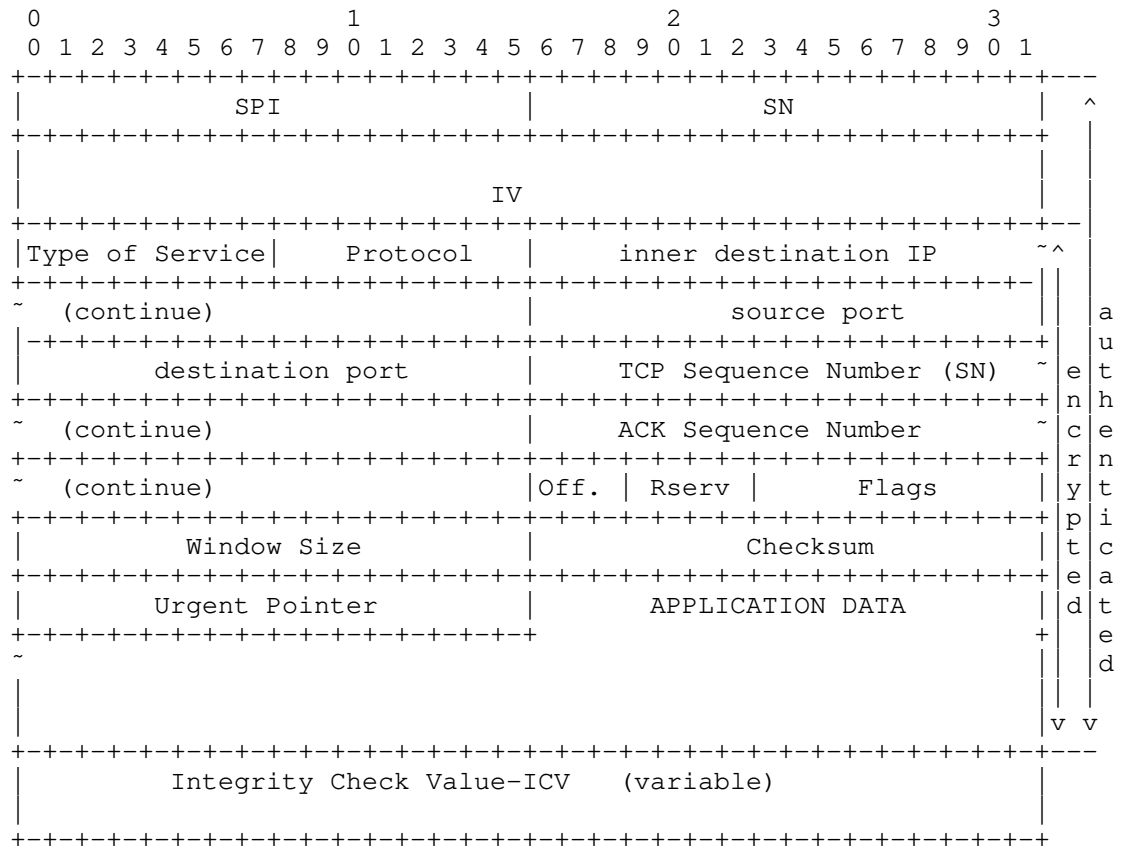


Figure 10: Diet-ESP VPN Packet Description

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

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

A.3.4. IPv4 in IPv6

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

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

Authors' Addresses

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

Email: daniel.migault@ericsson.com

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

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

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

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

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

Email: dschinazi.ietf@gmail.com

ipsecme
Internet-Draft
Intended status: Standards Track
Expires: 14 November 2022

D. Migault
Ericsson
T. Guggemos
LMU Munich
C. Bormann
Universitaet Bremen TZI
D. Schinazi
Google LLC
13 May 2022

ESP Header Compression and Diet-ESP
draft-mglt-ipsecme-diet-esp-08

Abstract

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

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

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

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 14 November 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Requirements notation	3
2. Introduction	3
3. Terminology	4
4. Protocol Overview	5
5. IPsec Compression Mode	7
6. EHC Context	7
6.1. EHC Context Parameters for ESP	8
6.2. EHC Context Parameters for Inner IP	8
6.3. EHC Context Parameters for Transport Protocol	10
7. EHC Rules	12
7.1. EHC Rules for ESP	14
7.2. EHC Rules for inner IPv4	16
7.3. EHC Rules for inner IPv6	19
7.4. EHC Rules for UDP	21
7.5. EHC Rules for UDP-Lite	22
7.6. EHC Rules for TCP	22
8. Diet-ESP EHC Strategy	24
8.1. Outbound Packet Processing	28
8.2. Inbound Packet Processing	30
9. IANA Considerations	32
10. Security Considerations	32
11. Privacy Considerations	34
12. Acknowledgment	34
13. References	34
13.1. Normative References	34

13.2. Informational References	35
Appendix A. Illustrative Examples	36
A.1. Single UDP Session IoT VPN	36
A.2. Single TCP session IoT VPN	39
A.3. Traditional VPN	43
Authors' Addresses	52

1. Requirements notation

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

2. Introduction

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

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

Most compression mechanisms work with dynamic compression contexts. Some mechanisms, such as ROHC [RFC5858], agree and dynamically change the context over a dedicated channel. Others, such as 6LowPAN, send the context together with the actual protocol information in a separate compression header. Those mechanism fail when it comes to compress encrypted payloads as appearing in ESP. This is found to be a major reason, why IPsec and in particular ESP is not widely developed in environments where bandwidth saving is a critical task, such as in IoT scenarios.

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

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

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

3. Terminology

This document uses the following terminology:

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

4. Protocol Overview

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

EHC is able to compress any protocol encapsulated in ESP and ESP itself. Concerned fields include those of the ESP protocol, as well as other protocols in the ESP payload such as the IP header when the tunnel mode is used, but also upper layer protocols, such as the UDP or the TCP header. Non ESP fields may be compressed by ESP under certain circumstances, but EHC is not intended to provide a generic way outside of ESP to compress these protocols. Compression of the unprotected IP header and the unencrypted ESP header may be performed by mechanism such as 6LoWPAN [RFC4944], SCHC [RFC8724], ROHC [RFC5795] or 6LoWPAN-GHC [RFC7400].

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

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

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

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

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

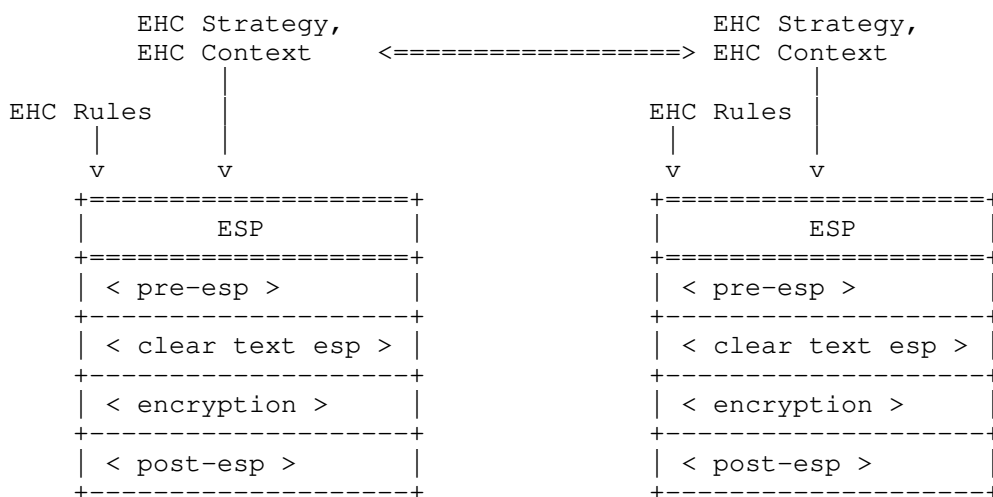


Figure 1: ESP Header Compression Overview

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

pre-esp: represents treatment performed to a non ESP packet, i.e. before ESP encapsulation or decapsulation is being performed.

Any compression of protocols not specific to but encrypted by ESP, such as L4 and higher protocols, is performed here.

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

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

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

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

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

5. IPsec Compression Mode

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

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

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

6. EHC Context

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

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

As this document is limited to the Diet-ESP strategy, the EHC Context in this section used by the Diet-ESP Strategy to activate specific EHC Rules as well as to execute the EHC Rules by providing the necessary parameters..

6.1. EHC Context Parameters for ESP

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

Table 1

6.2. EHC Context Parameters for Inner IP

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

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

Table 2

6.2.1. EHC Context Parameters for inner IPv6

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

Table 3

ip6_tcfl_comp indicates how Traffic Class and Flow Label fields of the inner IP Header are expected to be compressed. "Outer" indicates Traffic Class and Flow Label are read from the outer IP header, "Value" indicates these values are provided by the Diet-ESP Context, while "Uncompress" indicates that no compression occurs and these values are read in the inner IP inner header.

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

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

6.2.2. EHC Context Parameters for inner IPv4

Context Attribute	In SA	Possible Values
ip4_options	No	"Options", "No_Options"
ip4_id	No	IPv4 Identification

ip4_id_lsb	No	0,1,2	
+-----+-----+-----+			
ip4_ttl_comp	No	"Outer", "Value", "UnComp"	
+-----+-----+-----+			
ip4_ttl	No	IPv4 Time To Live	
+-----+-----+-----+			
ip4_src	Yes	IPv4 Source Address	
+-----+-----+-----+			
ip4_dst	Yes	IPv4 Destination Address	
+-----+-----+-----+			
ip4_frag_enable	No	"True", "False"	
+-----+-----+-----+			

Table 4

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

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

6.3. EHC Context Parameters for Transport Protocol

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

Context Attribute	In SA	Possible Values
14_proto	Yes	IPv6/ESP Next Header, IPv4 Protocol
14_src	Yes	UDP/UDP-Lite/TCP Source Port
14_dst	Yes	UDP/UDP-Lite/TCP Destination Port

Table 5

6.3.1. EHC Context Parameters for UDP

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

6.3.2. EHC Context Parameters for UDP-Lite

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

Table 6

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

6.3.3. EHC Context Parameters for TCP

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

Table 7

`tcp_sn` holds the current Sequence Number of the TCP session.

`tcp_ack` holds the current Acknowledgement Number of the TCP session.

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

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

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

7. EHC Rules

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

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

EHC Rule	Field	Action	Parameters
EHC_RULE_NAME	f1	a1	p1_1, ... p1_n
	~	...	~
	fm	am	pm_1, ... pm_n

Figure 2: EHC Rules

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

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

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

Table 8

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

For all actions, the function can be performed only when the appropriated parameters and fields are provided. When a field or a parameters does not have an appropriated value its value is

designated as "Unspecified". Specifically some fields such as inner IP addresses, ports or transport protocols are agreed during the SA negotiation and are specified by the SA. Their value in the SA may take various values that are not appropriated to enable a compression. For example, when these fields are defined as a range of values, or by selectors such as OPAQUE or ANY these fields cannot be retrieved from a local value. Instead, when they are defined as a "Single" value (i.e a single IP address, or a single port number or a single transport protocol number) compression and decompression can be performed. These SA related fields are considered as "Unspecified" when not limited to a "Single" value.

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

7.1. EHC Rules for ESP

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

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

Table 9

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

ESP_SN designates the EHC Rule compressing / decompressing the ESP Sequence Number. ESP_SN is performed in the "post-esp" phase. ESP_SN is only activated if the SN ("esp_sn"), the LSB significant bytes ("esp_sn_lsb") and the method used to generate the SN ("esp_sn_gen") are defined. The Sequence Number is compressed using "lsb". Similarly to the SPI, the Sequence Number MUST be retrieved in order to complete the signature check of the ESP packet. Unlike the SPI, the Sequence Number is not agreed by the peers, but is changing for every packet. As a result, in order to retrieve the Sequence Number from the LSB "esp_sn_lsb", the peers MUST agree on generating Sequence Number in a similar way. This is negotiated with "esp_sn_gen" and the receiver MUST ensure that "esp_sn_lsb" is big enough to absorb minor packet losses or time differences between the peers.

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

ESP_PAD designates the EHC Rule compressing / decompressing the Pad Length and Padding fields. ESP_PAD is performed in the "clear text esp" phase. Pad Length and Padding define the padding. The purpose of padding is to respect a 32 bit alignment for ESP or block sizes of the used cryptographic suite. As the ESP trailer is encrypted,

Padding and Pad Length MUST to be performed by ESP and not by the encryption algorithm. Thus, ESP_PAD always needs to respect the cipher alignment ("esp_encr"), if applicable. Compression may be performed especially when device support alignment smaller than 32 bit. Such alignment is designated as "esp_align" and the padding bytes are the necessary bytes so the ESP packet has a length that is a multiple of "esp_align".

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

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

7.2. EHC Rules for inner IPv4

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

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

Table 10

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

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

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

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

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

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

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

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

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

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

7.3. EHC Rules for inner IPv6

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

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

Table 11

IP6_OUTER designates an EHC Rule for compressing / decompressing the first 32 bits of the inner IPv6 header formed by the Version, Traffic Class and Flow Label. IP6_OUTER only proceeds to compression when both the outer and inner IP header are IPv6 header. When the outer IP header is an IPv4, the compression is bypassed. Bypassing enables to proceed to compression of IPv4 and IPv6 traffic in a VPN use case with a single SA. The Version "ip_version" is defined by the SA and is thus compressed using "elided". The other parameters Traffic

Class and Flow Label are compressed using "lower". More specifically, the fields are not sent. The receiver decompresses them by reading their value from the outer IPv6 header.

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

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

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

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

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

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

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

7.4. EHC Rules for UDP

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

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

Table 12

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

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

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

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

7.5. EHC Rules for UDP-Lite

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

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

Table 13

UDP-LITE_SRC works similarly to UDP_SRC

UDP-LITE_DST works similarly to UDP_DST

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

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

7.6. EHC Rules for TCP

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

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

Table 14

TCP_SRC works similarly to UDP_SRC.

TCP_DST works similarly to UDP_DST.

TCP_SN designates the EHC Rule compressing / decompressing the TCP Sequence Number. TCP_SN is only activated if the SN ("tcp_sn") and the LSB significant bytes ("tcp_lsb") are defined. The TCP SN is compressed using "lsb". The sending peer only places the LSB bytes of the TCP SN ("tcp_sn") and the receiving peer retrieve the TCP SN from the LSB bytes carried in the packets as well as from the TCP SN value stored in EHC Context ("tcp_sn"). The two peers MUST agree on the number of LSB bytes to be sent: "tcp_lsb". Upon agreeing on "tcp_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full TCP SN. Note also that unlike the ESP SN, the TCP SN is not part of the SA. As a result, when the SN is compressed, the value of the TCP SN MUST be stored in the EHC Context. The reserved attribute for that is "tcp_sn"

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

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

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

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

8. Diet-ESP EHC Strategy

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

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

- * Diet-ESP favors compression methods that required fewer configuration: For IPv6, ip6_tcfl_comp and ip6_hl_com to "Outer" so that ip6_tc, ip6_fl and ip6_hl can be derived from the packet. Similarly, ip4_ttl_comp has is set to "Outer" so ip4_tll can be derived from the packet.
- * Diet-ESP limits compression method to those foreseen as the most commonly used. As such, esp_sn_gen has been set to "Incremental" as this is the most common method used to generate SN. The other method would be "Time".
- * Diet-ESP limits compression to the most foreseen scenarios. IPv4 compression has been limited in favor of IPv6 as constraint devices have largely adopted IPv6, and the gain versus the complexity to deploy IPv4 inner IP addresses has not been proved. As a result some compressions for IPv4 are not considered by DIet-ESP. This involved compression of the IPv4 options by setting ip4_options to "No_Options". Similarly IPv4 ID compression has not been enabled by setting ip4_id and ip4_id_lsb to "Unspecified".
- * Diet-ESP negotiated values shared by different rules such as tcp_lsb which is shared for TCP ACK as well as for the TCP SN.

- * Diet-ESP defines a logic to set the necessary parameters from those agreed by the standard ESP agreement, which limits the setting of parameters.

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

ESP:

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

Table 15

IPv4:

EHC Rule	Activated if	Parameter	Value
IP4_OPT_DIS	ip_version==4	ip_version	In SA
IP4_LENGTH	ip_version==4	None	
IP4_FRAG_DIS	ip_version==4	None	
IP4_TTL_OUTER	ip_version==4	None	

IP4_TTL_OUTER	ip_version==4	l4_proto	In SA
IP4_CHECK	ip_version==4	None	
IP4_SRC	ip_version==4	ip4_src	In SA
IP4_DST	ip_version==4	ip4_dst	In SA

Table 16

IPv6:

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

Table 17

UDP:

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

Table 18

UDP-Lite:

EHC Rule	Activated if	Parameter	Value
UDP_LITE_SRC	14_proto==136	14_source	In SA
UDP_LITE_DST	14_proto==136	14_dest	In SA
UDP_LITE_COVERAGE	14_proto==136	udplite_coverage	Negotiated
UDP_LITE_CHECK	14_proto==136	None	

Table 19

TCP:

EHC Rule	Activated if	Parameter	Value
TCP_SRC	14_proto==6	14_source	In SA
TCP_DST	14_proto==6	14_dest	In SA
TCP_SN	14_proto==6	tcp_sn	In SA
		tcp_lsb	Negotiated
TCP_ACK	14_proto==6	tcp_ack	In SA
		tcp_lsb	Negotiated
TCP_OPTIONS	14_proto==6	tcp_options	Negotiated
TCP_CHECK	14_proto==6	None	
TCP_URGENT	14_proto==6	tcp_urgent	Negotiated

Table 20

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

- * esp_sn_lsb
- * esp_spi_lsb
- * esp_align
- * udplite_coverage
- * tcp_lsb
- * tcp_options

* tcp_urgent

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

8.1. Outbound Packet Processing

Diet-ESP compression is defined as follows:

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

UDP compression is defined as below:

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

UDP-lite compression is defined as below:

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

TCP compression is defined as below:

1. If "l4_src" designates a "Single" Source Port than apply the TCP_SRC to compress the Source Port.

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

Inner IPv6 Header compression is defined as below:

1. If "ip6_src" designates a "Single" Source IP address, apply the IP6_SRC to compress the IPv6 Source Address.
2. If "ip6_dst" designates a "Single" Destination IP address, apply the IP6_DST to decompress the IPv6 Destination Address.
3. Apply IP6_HL_OUTER to compress the Hop Limit.
4. If "l4_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP6_NH to compress the Next Header.
5. Apply, IP6_LENGTH to compress the Length.
6. Apply IP6_OUTER to compress Version, Traffic Class and Flow Label.

Inner IPv4 Header compression is defined as below:

1. Apply, IP4_LENGTH to compress the Length.
2. Apply IP4_TTL_OUTER to compress Time To Live.
3. Apply, IP4_CHECK to compress the IPv4 header checksum.
4. If "ip4_src" designates a "Single" Source IP address, apply the IP4_SRC to compress the IPv4 Source Address.
5. If "ip4_dst" designates a "Single" Destination IP address, apply the IP4_DST to decompress the IPv4 Destination Address.

ESP compression is defined as below:

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

8.2. Inbound Packet Processing

Diet-ESP decompression is defined as follows:

1. Match the inbound packet with the SA and determine if the Diet-ESP EHC Strategy has been activated. When Diet-ESP is activated this means that the "esp_spi_lsb" are sufficient to index the SA and proceed to next step, otherwise skip all steps associated to Diet-ESP and proceed to the standard ESP as defined in [RFC4303]
2. In phase "clear text esp" and "post-esp": Proceed to the ESP decompression.
3. In phase "clear text esp": If "ipsec_mode" is set to "Tunnel" mode, determine "ip_version" the IP version of the inner IP addresses and proceed to the appropriated inner IP address decompression, except for the computation of the checksums and length.
4. In phase "pre-esp": If "l4_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), proceed to the decompression of the specific layer, except for the computation of the checksums and length replaced by zero fields.
5. In phase "pre-esp": Proceed to the decompression of the checksums and length.

ESP decompression is defined as follows:

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

UDP decompression is defined as follows:

1. If "l4_src" designates a "Single" Source Port, apply UDP_SRC to decompress the Source Port.
2. If "l4_dst" designates a "Single" Destination Port, apply UDP_DST to decompress the Destination Port.

3. Apply UDP_LENGTH to compress the Length. The length value is computed from the length provided by the lower layer, with the additional added bytes during the UDP decompression including the length size.
4. Apply UDP_CHECK to decompress the Checksum.
5. Update the Length of the lower layers:
 1. If "ipsec_mode" is set to "Transport" mode, update the Length of the outer IP header (IPv4 or IPv6). The Length is incremented by the number of bytes generated by the decompression of the transport layer.
 2. If "ipsec_mode" is set to "Tunnel" mode, update the Length of the inner IP address (IPv4 or IPv6) as well as the outer IP header (IPv4 or IPv6). The Length is incremented by the number of bytes generated by the decompression of the transport layer.

UDP-Lite decompression is defined as follows:

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

TCP decompression is defined as follows:

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

Inner IPv6 decompression is defined as follows:

1. Apply IP6_OUTER to decompress Version, Traffic Class and Flow Label.
2. Set the Length to zero.

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

Inner IPv4 decompression is defined as follows:

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

9. IANA Considerations

There are no IANA consideration for this document.

10. Security Considerations

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

Security Parameter Index (SPI):

The Security Parameter Index (SPI) is used by the receiver to index the Security Association that contains appropriated cryptographic material. If the SPI is not found, the packet is rejected as no further checks can be performed. In EHC, the value of the SPI is not reduced, but compressed why the SPI value may not be fully provided between the compressor and the de-

compressor. On the other hand, its uncompressed value is provided to the ESP-processing and no weakness is introduced to ESP itself. On an implementation perspective, it is strongly recommended that decompression is deterministic. Compression and decompression adds some additional treatment to the ESP packet, which might be used by an attacker. In order to minimize the load associated to decompression, decompression is expected to be deterministic. The incoming compressed SPI with the associated IP addresses should output a single and unique uncompressed SPI value. If an uncompressed SPI values have to be considered, then the receiver could end in n signature checks which may be used by an attacker for a DoS attack.

Sequence Number (SN):

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

11. Privacy Considerations

Security Parameter Index (SPI):

Until Diet-ESP is deployed outside the scope of IoT and small devices, the use of a compressed SPI may provide an indication that one of the endpoint is a sensor. Such information may be used, for example, to evaluate the number of appliances deployed, or - in addition with other information, such as the time interval, the geographic location - be used to derive the type of data transmitted.

Sequence Number (SN):

If incremented for each ESP packet, the SN may leak some information like the amount of transmitted data or the age of the sensor. The age of the sensor may be correlated with the software used and the potential bugs. On the other hand, re-keying will re-initialize the SN, but the cost of a re-keying may not be negligible and thus, frequent re-keying can be considered. In addition to the re-key operation, the SN may be generated in order to reduce the accuracy of the information leaked. In fact, the SN does not have to be incremented by one for each packet it just has to be an increasing function. Using a function such as a TIME may prevent characterizing the age or the use of the sensor. Note that the use of such function may also impact the compression efficiency and result in larger compressed SN. Another possibility may also consists in compressing the SN to the low order bytes to reduce the information related to the age or the number of packets being exchanged.

12. Acknowledgment

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

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

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

13. References

13.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <<https://www.rfc-editor.org/info/rfc4309>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, DOI 10.17487/RFC5795, March 2010, <<https://www.rfc-editor.org/info/rfc5795>>.
- [RFC5858] Ertekin, E., Christou, C., and C. Bormann, "IPsec Extensions to Support Robust Header Compression over IPsec", RFC 5858, DOI 10.17487/RFC5858, May 2010, <<https://www.rfc-editor.org/info/rfc5858>>.
- [RFC7400] Bormann, C., "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 7400, DOI 10.17487/RFC7400, November 2014, <<https://www.rfc-editor.org/info/rfc7400>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

13.2. Informational References

- [RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <<https://www.rfc-editor.org/info/rfc8724>>.
- [RFC8750] Migault, D., Guggemos, T., and Y. Nir, "Implicit Initialization Vector (IV) for Counter-Based Ciphers in Encapsulating Security Payload (ESP)", RFC 8750, DOI 10.17487/RFC8750, March 2020, <<https://www.rfc-editor.org/info/rfc8750>>.
- [I-D.ietf-tsvwg-udp-options] Touch, J., "Transport Options for UDP", Work in Progress, Internet-Draft, draft-ietf-tsvwg-udp-options-18, 26 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-tsvwg-udp-options-18.txt>>.
- [fit-iot-lab] "Future Internet of Things (FIT) IoT-LAB", <<https://www.ietf-lab.info>>.

Appendix A. Illustrative Examples

A.1. Single UDP Session IoT VPN

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

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

Further compression may be done for example by using an implicit IV [RFC8750] and by compressing the outer IP addresses (not represented on the figure). In addition, application data may also be compressed with mechanisms outside of the scope of Diet-ESP.

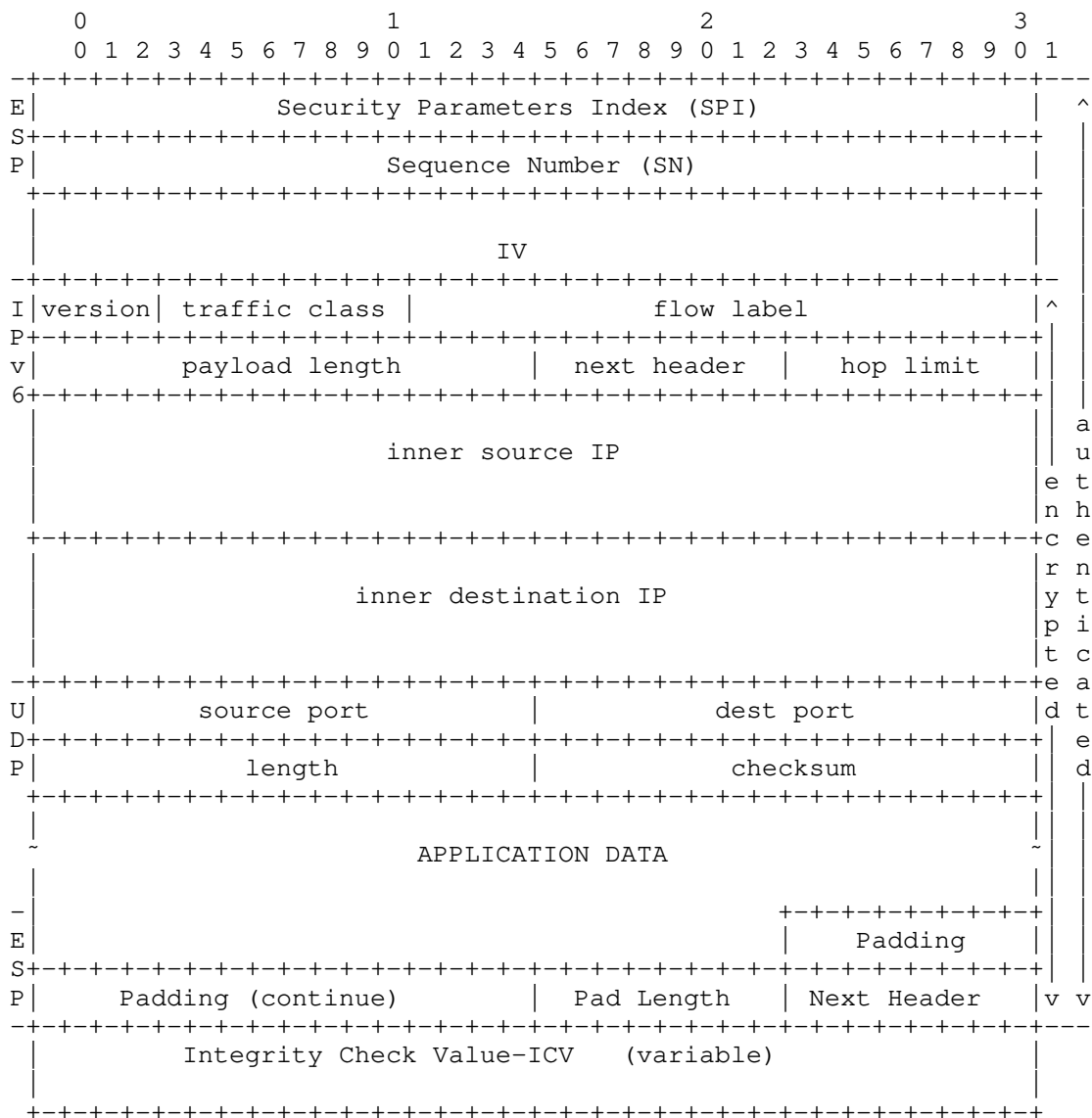


Figure 3: Standard ESP VPN Packet Description

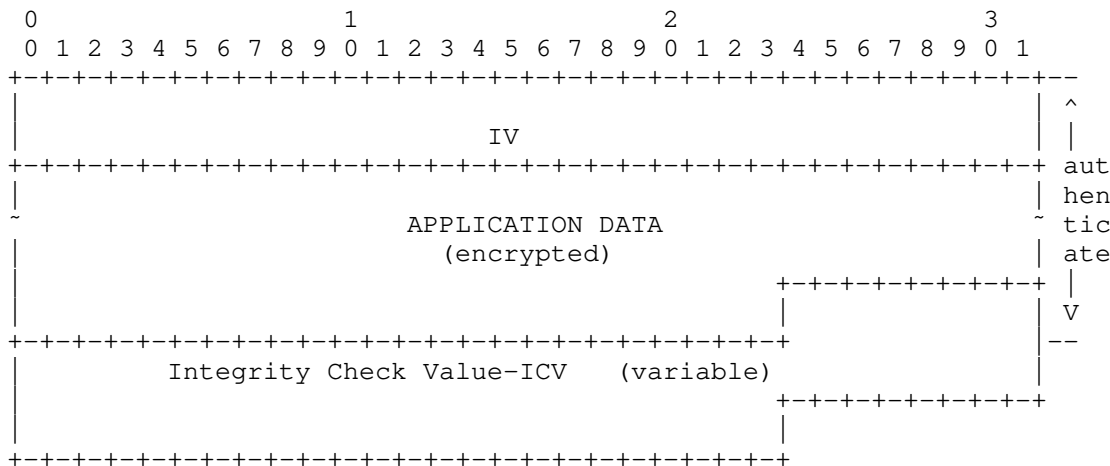


Figure 4: Diet-ESP Single UDP Session IoT VPN Packet Description

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

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

Table 21

A.2. Single TCP session IoT VPN

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

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

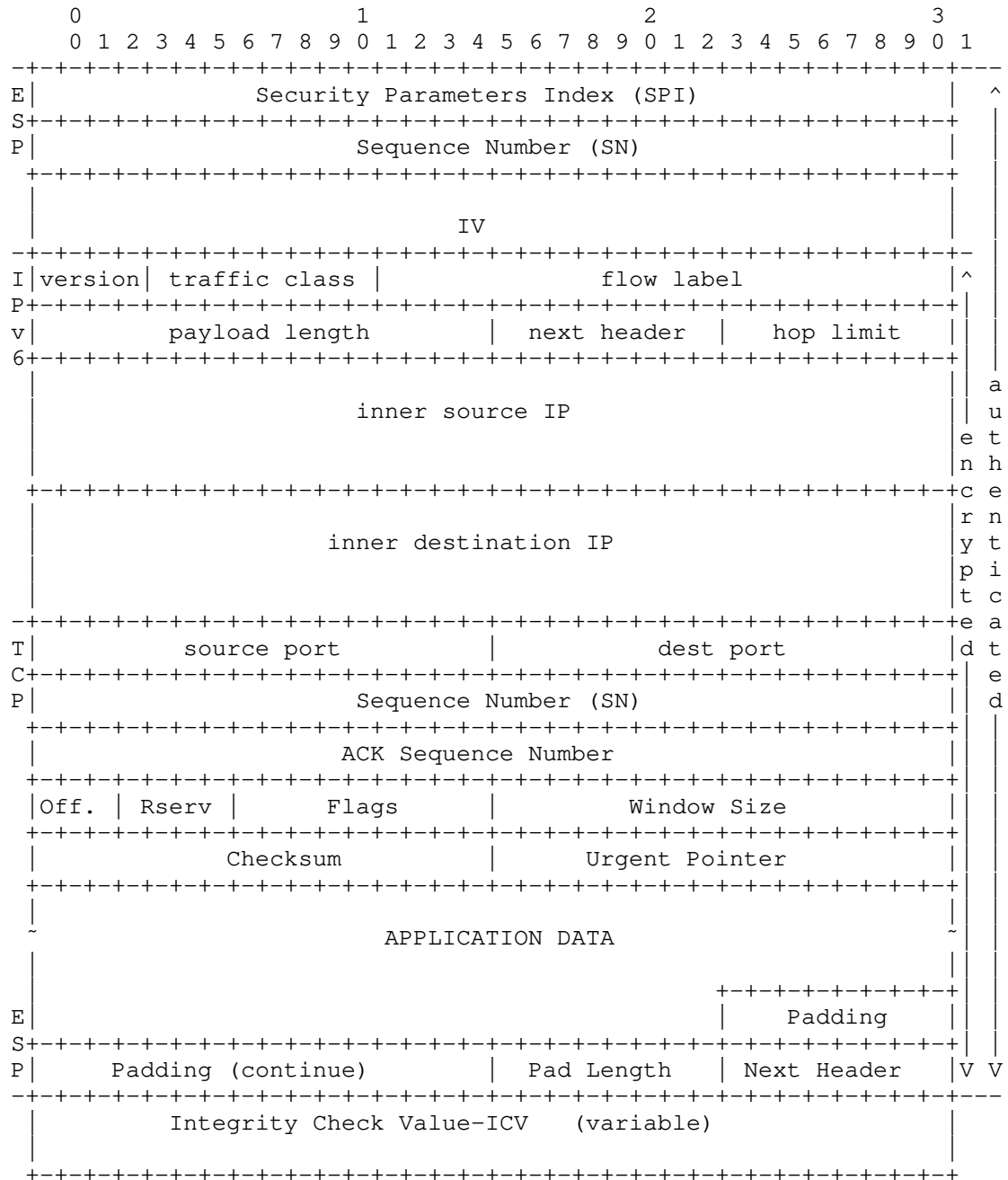


Figure 5: Standard IoT Single TCP Session VPN Packet Description

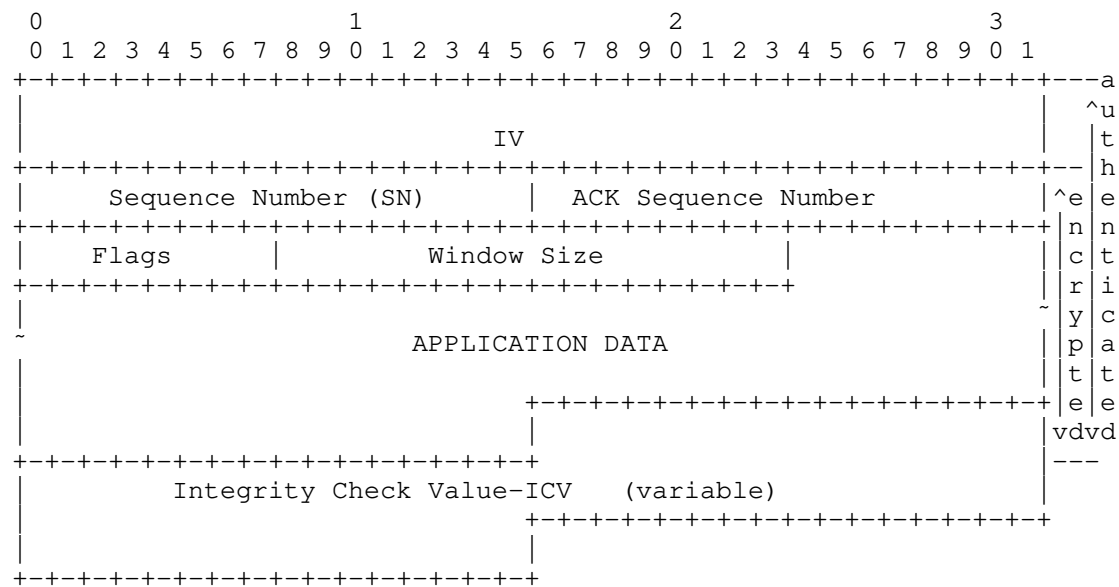


Figure 6: Diet-ESP Single TCP Session IoT VPN Packet Description

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

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	0
ESP_SN	esp_sn_lsb	0
	esp_sn_gen	
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	
	ip6_hl_comp	

IP6_LENGTH		
IP6_NH		
IP6_HL_OUTER		
IP6_SRC		
IP6_DST		
TCP_SRC		
TCP_DST		
TCP_SN	tcp_lsb	2
	tcp_sn	0
TCP_ACK	tcp_lsb	2
	tcp_ack	0
TCP_OPTIONS	tcp_options	"False"
TCP_CHECK		
TCP_URGENT	tcp_urgent	"False"

Table 22

A.3. Traditional VPN

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

A.3.1. IPv6 in IPv6

Figure 7 represents the original TCP packet with IPv6 inner IP addresses and Figure 8 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 32 bytes.

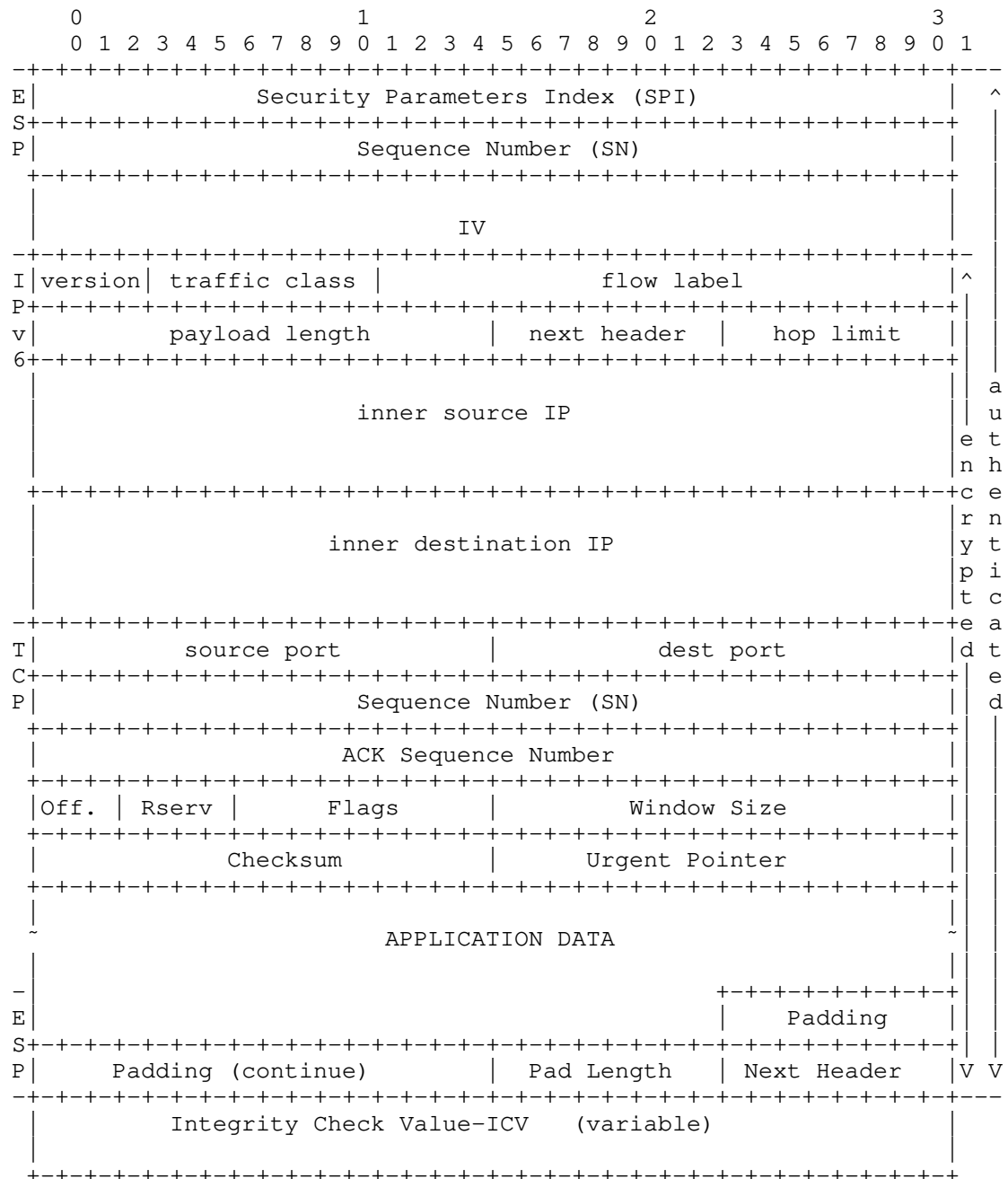


Figure 7: Standard ESP VPN Packet Description

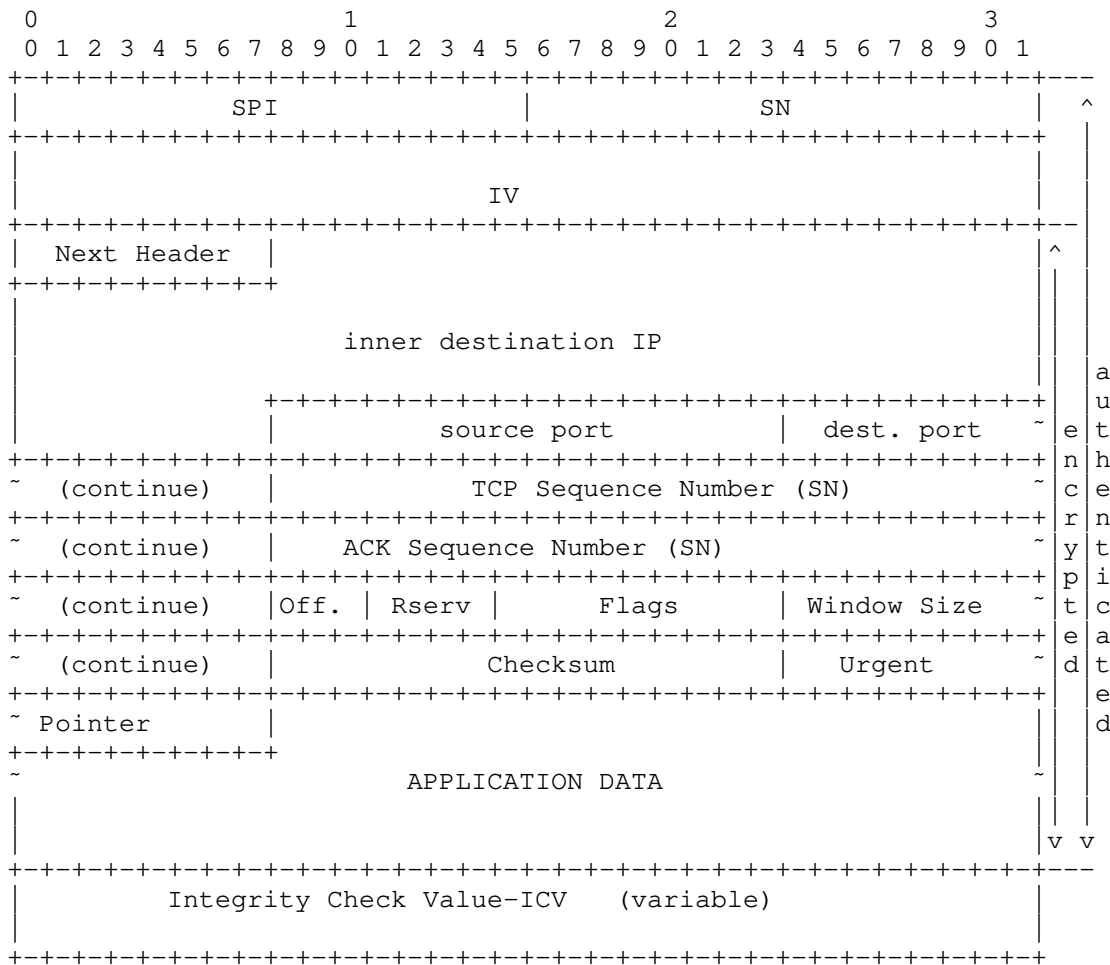


Figure 8: Diet-ESP VPN Packet Description

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

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

Table 23

A.3.2. IPv6 in IPv4

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

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

Table 24

A.3.3. IPv4 in IPv4

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

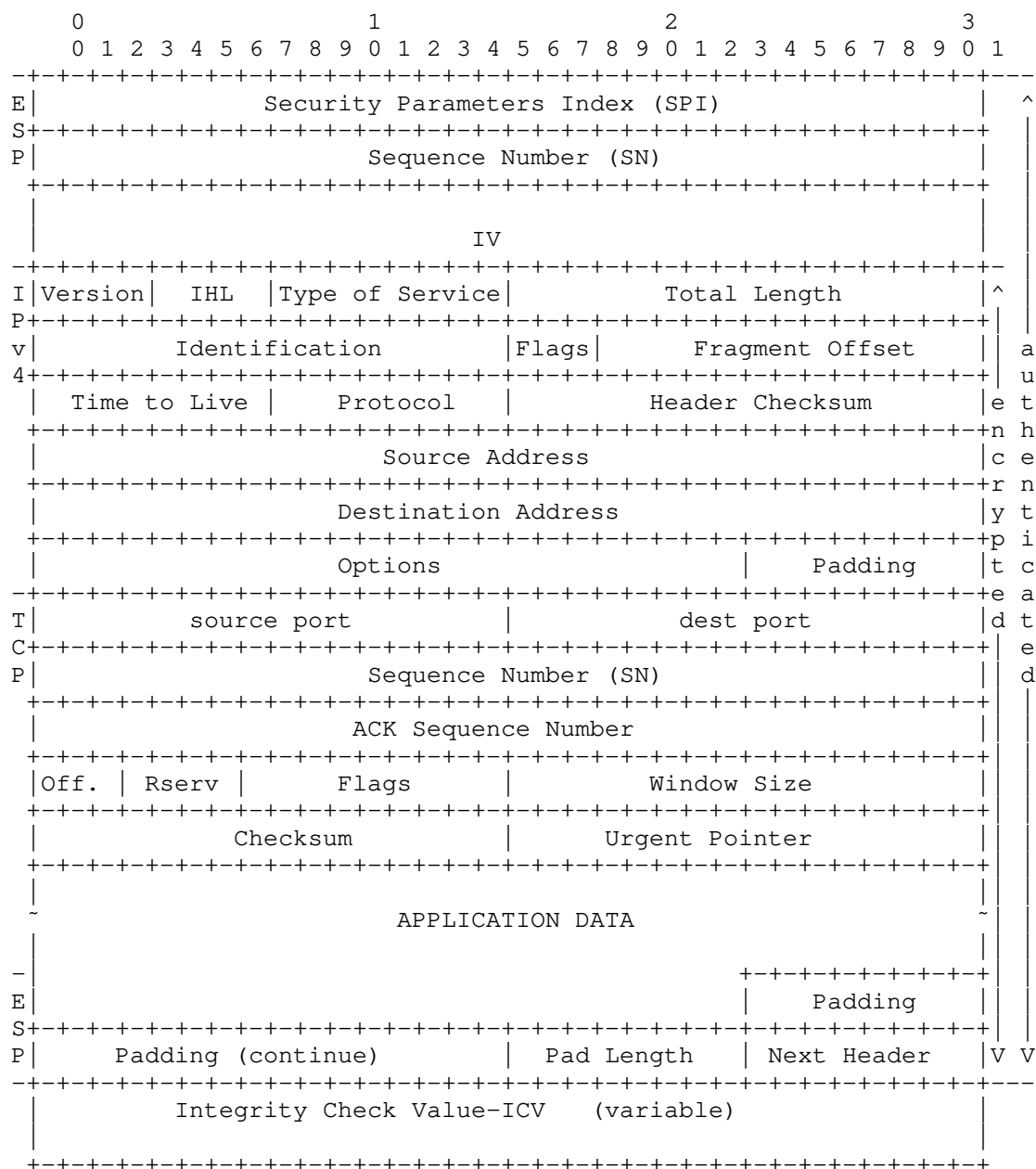


Figure 9: Standard ESP VPN Packet Description

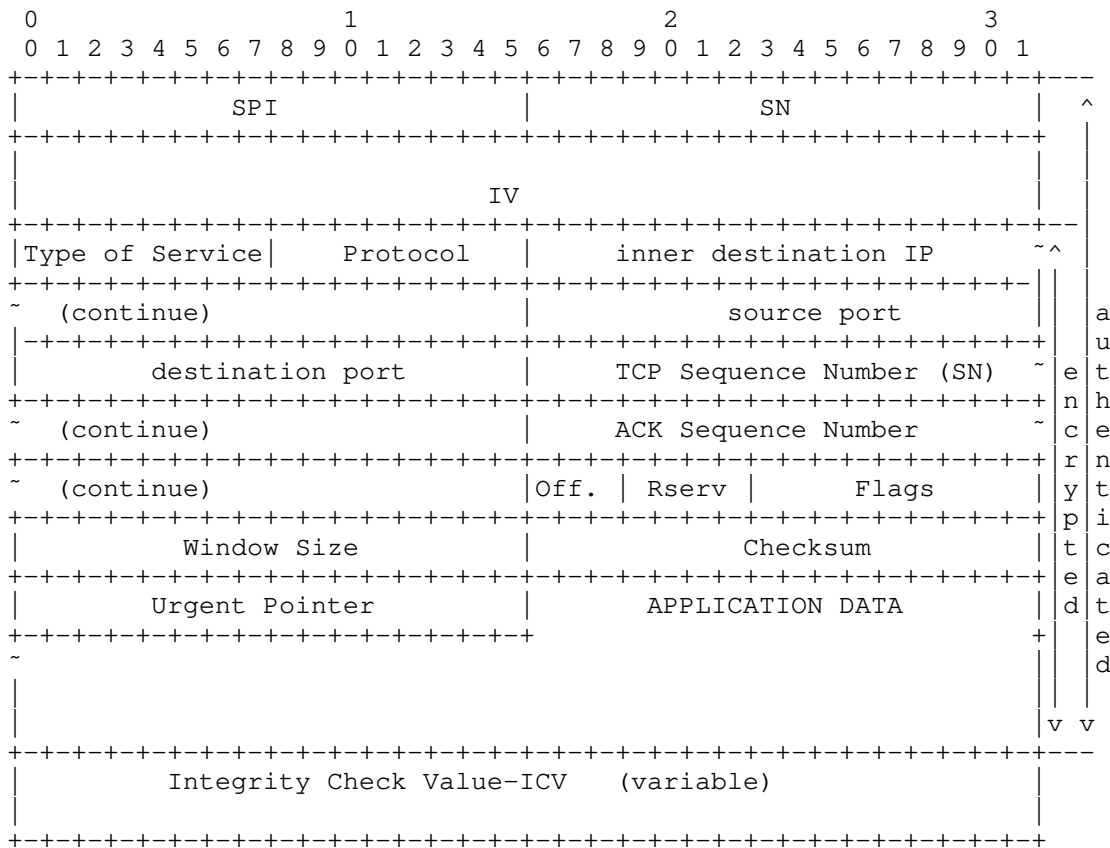


Figure 10: Diet-ESP VPN Packet Description

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

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

Table 25

A.3.4. IPv4 in IPv6

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

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

Table 26

Authors' Addresses

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada
Email: daniel.migault@ericsson.com

Tobias Guggemos
LMU Munich
Oettingenstr. 67
80538 Munchen
Germany
Email: guggemos@nm.ifi.lmu.de
URI: <http://www.nm.ifi.lmu.de/~guggemos>

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
Bremen D-28359
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043
United States of America
Email: dschinazi.ietf@gmail.com

Network
Internet-Draft
Updates: 7296,8221,8247 (if approved)
Intended status: Standards Track
Expires: August 25, 2021

P. Wouters, Ed.
Red Hat
February 21, 2021

Deprecation of IKEv1 and obsoleted algorithms
draft-pwouters-ikev1-ipsec-graveyard-06

Abstract

Internet Key Exchange version 1 (IKEv1) is deprecated. Accordingly, IKEv1 has been moved to Historic status. A number of old algorithms that are associated with IKEv1, and not widely implemented for IKEv2 are deprecated as well. IANA is instructed to close all IKEv1 registries.

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 August 25, 2021.

Copyright Notice

Copyright (c) 2021 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.

Table of Contents

1. Introduction	2
2. Requirements Language	2
3. RFC 2409 to Historic	3
4. Deprecating obsolete algorithms	3
5. Security Considerations	3
6. IANA Considerations	4
7. References	5
7.1. Normative References	5
7.2. Informative References	6
Author's Address	6

1. Introduction

IKEv1 [RFC2409] and its related documents for ISAKMP [RFC2408] and IPsec DOI [RFC2407] were obsoleted by IKEv2 [RFC4306] in December 2005. The latest version of IKEv2 at the time of writing was published in 2014 in [RFC7296]. The Internet Key Exchange (IKE) version 2 has replaced version 1 over 15 years ago. IKEv2 has now seen wide deployment and provides a full replacement for all IKEv1 functionality. No new modifications or new algorithms have been accepted for IKEv1 for at least a decade. IKEv2 addresses various issues present in IKEv1, such as IKEv1 being vulnerable to amplification attacks. IKEv1 has been moved to Historic status, and this document requests IANA to close all IKEv1 registries.

Algorithm implementation requirements and usage guidelines for IKEv2 [RFC8247] and ESP/AH [RFC8223] gives guidance to implementors but limits that guidance to avoid broken or weak algorithms. It does not deprecate algorithms that have aged and are not in use, but leave these algorithms in a state of "MAY be used". This document deprecates those algorithms that are no longer advised but for which there are no known attacks resulting in their earlier deprecation.

2. Requirements Language

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

3. RFC 2409 to Historic

IKEv1 is deprecated. Systems running IKEv1 should be upgraded and reconfigured to run IKEv2. Systems that support IKEv1 but not IKEv2 are most likely also unsuitable candidates for continued operation. Such unsupported systems have a much higher chance of containing an implementation vulnerability that will never be patched. IKEv1 systems can be abused for packet amplification attacks. IKEv1 systems most likely do not support modern algorithms such as AES-GCM or CHACHA20_POLY1305 and quite often only support or have been configured to use the very weak Diffie-Hellman Groups 2 and 5. IKEv1 systems should be upgraded or replaced by IKEv2 systems.

IKEv1 and its way of using Preshared Keys (PSKs) protects against quantum computer based attacks. IKEv2 updated its use of PSK to improve the error reporting, but at the expense of post-quantum security. If post-quantum security is required, these systems should be migrated to use IKEv2 Postquantum Preshared Keys (PPK) [RFC8784]

Some IKEv1 implementations support Labeled IPsec, a method to negotiate an additional Security Context selector to the SPD, but this method was never standardized in IKEv1. Those IKEv1 systems that require Labeled IPsec should migrate to an IKEv2 system supporting Labeled IPsec as specified in [draft-ietf-ipsecme-labeled-ipsec].

4. Deprecating obsolete algorithms

This document deprecates the following algorithms:

- o Encryption Algorithms: RC5, IDEA, CAST, Blowfish, and the unspecified 3IDEA, ENCR_DES_IV64 and ENCR_DES_IV32
- o PRF Algorithms: the unspecified PRF_HMAC_TIGER
- o Integrity Algorithms: HMAC-MD5-128
- o Diffie-Hellman groups: none

5. Security Considerations

There are only security benefits by deprecating IKEv1 for IKEv2.

The deprecated algorithms have long been in disuse and are no longer actively deployed or researched. It presents an unknown security risk that is best avoided. Additionally, these algorithms not being supported in implementations simplifies those implementations and reduces the accidental use of these deprecated algorithms through misconfiguration or downgrade attacks.

6. IANA Considerations

This document instructs IANA to mark all IKEv1 registries as DEPRECATED.

Additionally, this document instructs IANA to add an additional Status column to the IKEv2 Transform Type registries and mark the following entries as DEPRECATED:

Transform Type 1 - Encryption Algorithm IDs

Number	Name	Status
-----	-----	-----
1	ENCR_DES_IV64	DEPRECATED [this document]
2	ENCR_DES	DEPRECATED [RFC8247]
4	ENCR_RC5	DEPRECATED [this document]
5	ENCR_IDEA	DEPRECATED [this document]
6	ENCR_CAST	DEPRECATED [this document]
7	ENCR_BLOWFISH	DEPRECATED [this document]
8	ENCR_3IDEA	DEPRECATED [this document]
9	ENCR_DES_IV32	DEPRECATED [this document]

Figure 1

Transform Type 2 - Pseudorandom Function Transform IDs

Number	Name	Status
-----	-----	-----
1	PRF_HMAC_MD5	DEPRECATED [RFC8247]
1	PRF_HMAC_TIGER	DEPRECATED [this document]

Figure 2

Transform Type 3 - Integrity Algorithm Transform IDs

Number	Name	Status
-----	-----	-----
1	AUTH_HMAC_MD5_96	DEPRECATED [RFC8247]
3	AUTH_DES_MAC	DEPRECATED [RFC8247]
4	AUTH_KPDK_MD5	DEPRECATED [RFC8247]
6	AUTH_HMAC_MD5_128	DEPRECATED [this document]
7	AUTH_HMAC_SHA1_160	DEPRECATED [this document]

Figure 3

Transform Type 4 - Diffie Hellman Group Transform IDs

Number	Name	Status
-----	-----	-----
1	768-bit MODP Group	DEPRECATED [RFC8247]
22	1024-bit MODP Group with 160-bit Prime Order Subgroup	DEPRECATED [RFC8247]

Figure 4

All entries not mentioned here should receive no value in the new Status field.

This document instructs IANA to close and mark as obsolete the Internet Key Exchange (IKE) Attributes registries as well as the "Magic Numbers" for ISAKMP Protocol registries.

The IESG is requested to designate IKEv1 to Historic.

7. References

7.1. Normative References

- [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>>.
- [RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, DOI 10.17487/RFC2407, November 1998, <<https://www.rfc-editor.org/info/rfc2407>>.
- [RFC2408] Maughan, D., Schertler, M., Schneider, M., and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, DOI 10.17487/RFC2408, November 1998, <<https://www.rfc-editor.org/info/rfc2408>>.
- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <<https://www.rfc-editor.org/info/rfc2409>>.
- [RFC4306] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, DOI 10.17487/RFC4306, December 2005, <<https://www.rfc-editor.org/info/rfc4306>>.

- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8223] Esale, S., Torvi, R., Jalil, L., Chunduri, U., and K. Raza, "Application-Aware Targeted LDP", RFC 8223, DOI 10.17487/RFC8223, August 2017, <<https://www.rfc-editor.org/info/rfc8223>>.
- [RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8247, DOI 10.17487/RFC8247, September 2017, <<https://www.rfc-editor.org/info/rfc8247>>.
- [RFC8784] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smyslov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <<https://www.rfc-editor.org/info/rfc8784>>.

7.2. Informative References

- [draft-ietf-ipsecme-labeled-ipsec] Wouters, P. and S. Prasad, "Labeled IPsec Traffic Selector support for IKEv2", draft-ietf-ipsecme-labeled-ipsec (work in progress), March 2019.

Author's Address

Paul Wouters (editor)
Red Hat

Email: pwouters@redhat.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 6, 2019

V. Smyslov
ELVIS-PLUS
December 3, 2018

Intermediate Exchange in the IKEv2 Protocol
draft-smyslov-ipsecme-ikev2-aux-02

Abstract

This documents defines a new exchange, called Intermediate Exchange, for the Internet Key Exchange protocol Version 2 (IKEv2). This exchange can be used for transferring large amount of data in the process of IKEv2 Security Association (SA) establishment. Introducing Intermediate Exchange allows re-using existing IKE Fragmentation mechanism, that helps to avoid IP fragmentation of large IKE messages, but cannot be used in the initial IKEv2 exchange.

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 June 6, 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 (<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 and Notation	3
3. Intermediate Exchange Details	3
3.1. Support for Intermediate Exchange Negotiation	3
3.2. Using Intermediate Exchange	4
3.3. The INTERMEDIATE Exchange Protection and Authentication	5
3.3.1. Protection of the INTERMEDIATE Messages	5
3.3.2. Authentication of the INTERMEDIATE Exchanges	5
3.4. Error Handling in the INTERMEDIATE Exchange	8
4. Interaction with other IKEv2 Extensions	8
5. Security Considerations	8
6. IANA Considerations	9
7. Acknowledgements	9
8. References	9
8.1. Normative References	9
8.2. Informative References	10
Author's Address	10

1. Introduction

The Internet Key Exchange protocol version 2 (IKEv2) defined in [RFC7296] uses UDP as a transport for its messages. If size of the messages is large enough, IP fragmentation takes place, that may interfere badly with some network devices. The problem is described in more detail in [RFC7383], which also defines an extension to the IKEv2 called IKE Fragmentation. This extension allows IKE messages to be fragmented at IKE level, eliminating possible issues caused by IP fragmentation. However, the IKE Fragmentation cannot be used in the initial IKEv2 exchange, `IKE_SA_INIT`. This limitation in most cases is not a problem, since the `IKE_SA_INIT` messages used to be small enough not to cause IP fragmentation.

Recent progress in Quantum Computing has brought a concern that classical Diffie-Hellman key exchange methods will become insecure in a relatively near future and should be replaced with Quantum Computer (QC) resistant ones. Currently most of QC-resistant key exchange methods have large public keys. If these keys are exchanged in the `IKE_SA_INIT`, then most probably IP fragmentation will take place, therefore all the problems caused by it will become inevitable.

A possible solution to the problem would be to use TCP as a transport for IKEv2, as defined in [RFC8229]. However this approach has

significant drawbacks and is intended to be a "last resort" when UDP transport is completely blocked by intermediate network devices.

This document defines a new exchange for the IKEv2 protocol, called Intermediate Exchange or INTERMEDIATE. One or more these exchanges may take place right after the IKE_SA_INIT exchange and prior to the IKE_AUTH exchange. The INTERMEDIATE exchange messages can be fragmented using IKE Fragmentation mechanism, so these exchanges may be used to transfer large amounts of data which don't fit into the IKE_SA_INIT exchange without causing IP fragmentation.

While ability to transfer large public keys of QC-resistant key exchange methods is a primary motivation for introducing of the Intermediate Exchange, its application is not limited to this use case. This exchange may be used whenever some data need to be transferred before the IKE_AUTH exchange and for some reason the IKE_SA_INIT exchange is not suited for this purpose. This document defines the INTERMEDIATE exchange without tying it to any specific use case. It is expected that separate specifications will define for which purposes and how the INTERMEDIATE exchange is used in the IKEv2.

2. Terminology and Notation

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

3. Intermediate Exchange Details

3.1. Support for Intermediate Exchange Negotiation

The initiator indicates its support for Intermediate Exchange by including a notification of type INTERMEDIATE_EXCHANGE_SUPPORTED in the IKE_SA_INIT request message. If the responder also supports this exchange, it includes this notification in the response message.

Initiator	Responder
-----	-----
HDR, SAi1, KEi, Ni, [N(INTERMEDIATE_EXCHANGE_SUPPORTED)] -->	<-- HDR, SAR1, KEr, Nr, [CERTREQ], [N(INTERMEDIATE_EXCHANGE_SUPPORTED)]

The INTERMEDIATE_EXCHANGE_SUPPORTED is a Status Type IKEv2 notification. Its Notify Message Type is <TBA by IANA>. Protocol ID

and SPI Size are both set to 0. This specification doesn't define any data this notification may contain, so the Notification Data is left empty. However, future enhancements of this specification may override this. Implementations MUST ignore the non-empty Notification Data if they don't understand its purpose.

3.2. Using Intermediate Exchange

If both peers indicated their support for the Intermediate Exchange, the initiator may use one or more these exchanges to transfer additional data. Using the INTERMEDIATE exchange is optional, the initiator may find it unnecessary after completing the IKE_SA_INIT exchange.

The Intermediate Exchange is denoted as INTERMEDIATE, its Exchange Type is <TBA by IANA>.

Initiator	Responder
-----	-----
HDR, ..., SK {...} -->	<-- HDR, ..., SK {...}

The initiator may use several INTERMEDIATE exchanges if necessary. Since initiator's Window Size is initially set to one (Section 2.3 of [RFC7296]), these exchanges MUST follow each other and MUST all be completed before the IKE_AUTH exchange is initiated. The IKE SA MUST NOT be considered as established until the IKE_AUTH exchange is successfully completed.

The Message IDs for the INTERMEDIATE exchanges MUST be chosen according to the standard IKEv2 rule, described in the Section 2.2. of [RFC7296], i.e. it is set to 1 for the first INTERMEDIATE exchange, 2 for the next (if any) and so on. The message ID for the first pair of the IKE_AUTH messages is one more than the one that was used in the last INTERMEDIATE exchange.

If the presence of NAT is detected in the IKE_SA_INIT exchange via NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP notifications, then the peers MUST switch to port 4500 immediately once this exchange is completed, i.e. in the first INTERMEDIATE exchange.

The content of the INTERMEDIATE exchange messages depends on the data being transferred and will be defined by specifications utilizing this exchange. However, since the main motivation for the INTERMEDIATE exchange is to avoid IP fragmentation when large amount of data need to be transferred prior to IKE_AUTH, the Encrypted payload MUST be present in the INTERMEDIATE exchange messages and

payloads containing large data MUST be placed inside. This will allow IKE Fragmentation [RFC7383] to take place, provided it is supported by the peers and negotiated in the initial exchange.

3.3. The INTERMEDIATE Exchange Protection and Authentication

3.3.1. Protection of the INTERMEDIATE Messages

The keys $SK_e[i/r]$ and $SK_a[i/r]$ for the Encrypted payload in the INTERMEDIATE exchanges are computed in a standard fashion, as defined in the Section 2.14 of [RFC7296]. Every subsequent INTERMEDIATE exchange uses the most recently calculated keys before this exchange is started. The first INTERMEDIATE exchange always uses $SK_e[i/r]$ and $SK_a[i/r]$ keys that were computed as result the IKE_SA_INIT exchange. If this INTERMEDIATE exchange performs additional key exchange resulting in the update of $SK_e[i/r]$ and $SK_a[i/r]$, then these updated keys are used for encryption and authentication of next INTERMEDIATE exchange, otherwise the current keys are used, and so on.

3.3.2. Authentication of the INTERMEDIATE Exchanges

The data transferred in the INTERMEDIATE exchanges must be authenticated in the IKE_AUTH exchange. For this purpose the definition of the blob to be signed (or MAC'ed) from the Section 2.15 of [RFC7296] is modified as follows:

InitiatorSignedOctets = RealMsg1		NonceRData		MACedIDForI	[IntAuth]
ResponderSignedOctets = RealMsg2		NonceIDData		MACedIDForR	[IntAuth]

IntAuth = IntAuth_1 | [| IntAuth_2 [| IntAuth_3]] ...

IntAuth_1 = IntAuth_1_I		IntAuth_1_R
IntAuth_2 = IntAuth_2_I		IntAuth_2_R
IntAuth_3 = IntAuth_3_I		IntAuth_3_R
...		

IntAuth_1_I = prf(SK_{pi_1} , [IntAuth_1_I_P		IntAuth_1_I_A)
IntAuth_2_I = prf(SK_{pi_2} , [IntAuth_2_I_P		IntAuth_2_I_A)
IntAuth_3_I = prf(SK_{pi_3} , [IntAuth_3_I_P		IntAuth_3_I_A)
...		

IntAuth_1_R = prf(SK_{pr_1} , [IntAuth_1_R_P		IntAuth_1_R_A)
IntAuth_2_R = prf(SK_{pr_2} , [IntAuth_2_R_P		IntAuth_2_R_A)
IntAuth_3_R = prf(SK_{pr_3} , [IntAuth_3_R_P		IntAuth_3_R_A)
...		

IntAuth_1_I/IntAuth_1_R, IntAuth_2_I/IntAuth_2_R, IntAuth_3_I/IntAuth_3_R, etc. represent the results of applying the negotiated prf to the content of the INTERMEDIATE messages sent by the initiator (IntAuth_*_I) and by the responder (IntAuth_*_R) in an order of increasing Message IDs (i.e. in an order the INTERMEDIATE exchanges took place). The prf is applied to the two chunks of data: optional IntAuth_*_[I/R]_P and mandatory IntAuth_*_[I/R]_A. The IntAuth_*_[I/R]_A chunk lasts from the first octet of the IKE Header (not including prepended four octets of zeros, if port 4500 is used) to the last octet of the Encrypted Payload header. The IntAuth_*_[I/R]_P chunk is present if the Encrypted payload is not empty. It consists of the not yet encrypted content of the Encrypted payload, excluding Initialization Vector, Padding, Pad Length and Integrity Checksum Data fields (see 3.14 of [RFC7296] for description of the Encrypted payload). In other words, the IntAuth_*_[I/R]_P chunk is the inner payloads of the Encrypted payload in plaintext form.

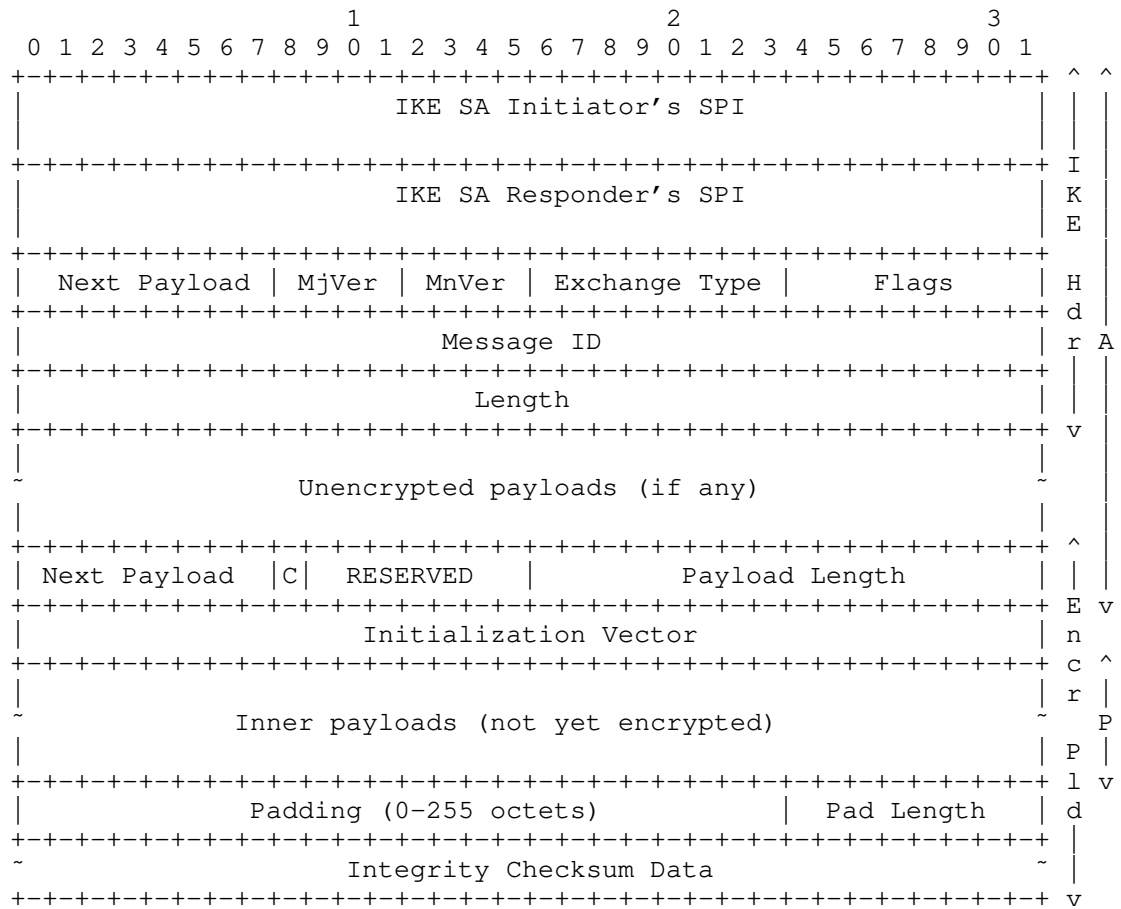


Figure 1: Data to Authenticate in the INTERMEDIATE Exchange Messages

Figure 1 illustrates the layout of the IntAuth_*_[I/R]_P (denoted as P) and the IntAuth_*_[I/R]_A (denoted as A) chunks in case the Encrypted payload is not empty.

The calculations are applied to whole messages only, before possible fragmentation. This ensures that the IntAuth will be the same regardless of whether fragmentation takes place or not ([RFC7383] allows sending first unfragmented message and then trying fragmentation in case of no reply).

Each calculation of IntAuth_*_[I/R] uses its own key SK_p[i/r]_*, which is the most recently updated SK_p[i/r] key available before the corresponded INTERMEDIATE exchange is started. The first INTERMEDIATE exchange always uses SK_p[i/r] key that was computed in

the `IKE_SA_INIT` as `SK_p[i/r]_1`. If the first `INTERMEDIATE` exchange performs additional key exchange resulting in `SK_p[i/r]` update, then this updated `SK_p[i/r]` is used as `SK_p[i/r]_2`, otherwise the original `SK_p[i/r]` is used, and so on. Note, that if keys are updated then for any given `INTERMEDIATE` exchange the keys `SK_e[i/r]` and `SK_a[i/r]` used for its messages protection (see Section 3.3.1) and the keys `SK_p[i/r]` for its authentication are always from the same generation.

3.4. Error Handling in the `INTERMEDIATE` Exchange

Since messages of the `INTERMEDIATE` exchange are not authenticated until the `IKE_AUTH` exchange successfully completes, possible errors need to be handled carefully. There is a trade-off between providing a better diagnostics of the problem and a risk to become a part of DoS attack. See Section 2.21.1 and 2.21.2 of [RFC7296] describe how errors are handled in initial IKEv2 exchanges, these considerations are applied to the `INTERMEDIATE` exchange too.

4. Interaction with other IKEv2 Extensions

The `INTERMEDIATE` exchanges MAY be used in the IKEv2 Session Resumption [RFC5723] between the `IKE_SESSION_RESUME` and the `IKE_AUTH` exchanges.

5. Security Considerations

The data that is transferred by means of the `INTERMEDIATE` exchanges is not authenticated until the subsequent `IKE_AUTH` exchange is completed. However, if the data is placed inside the Encrypted payload, then it is protected from passive eavesdroppers. In addition the peers can be certain that they receives messages from the party he/she performed the `IKE_SA_INIT` with if they can successfully verify the Integrity Checksum Data of the Encrypted payload.

The main application for Intermediate Exchange is to transfer large amount of data before IKE SA is set up without causing IP fragmentation. For that reason it is expected that in most cases IKE Fragmentation will be employed in the `INTERMEDIATE` exchanges. Section 5 of [RFC7383] contains security considerations for IKE Fragmentation.

Note, that if an attacker was able to break key exchange in real time (e.g. by means of Quantum Computer), then the security of the `INTERMEDIATE` exchange would degrade. In particular, such an attacker would be able both to read data contained in the Encrypted payload and to forge it. The forgery would become evident in the `IKE_AUTH` exchange (provided the attacker cannot break employed authentication

mechanism), but the ability to inject forged the INTERMEDIATE exchange messages with valid ICV would allow the attacker to mount Denial-of-Service attack. Moreover, if in this situation the negotiated prf was not secure against preimage attack with known key, then the attacker could forge the INTERMEDIATE exchange messages without later being detected in the IKE_AUTH exchange. To do this the attacker should find the same IntAuth_*_[I|R] value for the forged message as for original.

6. IANA Considerations

This document defines a new Exchange Type in the "IKEv2 Exchange Types" registry:

<TBA> INTERMEDIATE

This document also defines a new Notify Message Types in the "Notify Message Types - Status Types" registry:

<TBA> INTERMEDIATE_EXCHANGE_SUPPORTED

7. Acknowledgements

The idea to use an intermediate exchange between IKE_SA_INIT and IKE_AUTH was first suggested by Tero Kivinen. Scott Fluhrer and Daniel Van Geest identified a possible problem with authentication of the INTERMEDIATE exchange and helped to resolve it.

8. References

8.1. Normative References

- [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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [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>>.

- [RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <<https://www.rfc-editor.org/info/rfc7383>>.

8.2. Informative References

- [RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <<https://www.rfc-editor.org/info/rfc8229>>.
- [RFC5723] Sheffer, Y. and H. Tschofenig, "Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption", RFC 5723, DOI 10.17487/RFC5723, January 2010, <<https://www.rfc-editor.org/info/rfc5723>>.

Author's Address

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
RU

Phone: +7 495 276 0211
Email: svan@elvis.ru

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: July 18, 2019

C. Tjhai
M. Tomlinson
Post-Quantum
G. Bartlett
S. Fluhrer
Cisco Systems
D. Van Geest
ISARA Corporation
O. Garcia-Morchon
Philips
V. Smyslov
ELVIS-PLUS
January 14, 2019

Framework to Integrate Post-quantum Key Exchanges into Internet Key
Exchange Protocol Version 2 (IKEv2)
draft-tjhai-ipsecme-hybrid-qske-ikev2-03

Abstract

This document describes how to extend Internet Key Exchange Protocol Version 2 (IKEv2) so that the shared secret exchanged between peers has resistance against quantum computer attacks. The basic idea is to exchange one or more post-quantum key exchange payloads in conjunction with the existing (Elliptic Curve) Diffie-Hellman payload.

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 July 18, 2019.

Copyright Notice

Copyright (c) 2019 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.

Table of Contents

1. Introduction	2
1.1. Problem Description	2
1.2. Proposed Extension	3
1.3. Changes	4
1.4. Document Organization	4
2. Design Criteria	5
3. The Framework of Hybrid Post-Quantum Key Exchange	6
3.1. Overall design	6
3.2. Overall Protocol	8
3.2.1. IKE_SA_INIT Round: Negotiation	8
3.2.2. INTERMEDIATE Round: Additional Key Exchanges	9
3.2.3. IKE_AUTH Exchange	10
3.2.4. CREATE_CHILD_SA Exchange	10
4. Alternative Design	11
5. IANA Considerations	15
6. Security Considerations	15
7. References	17
7.1. Normative References	17
7.2. Informative References	17
Acknowledgements	18
Authors' Addresses	18

1. Introduction

1.1. Problem Description

Internet Key Exchange Protocol (IKEv2) as specified in RFC 7296 [RFC7296] uses the Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH) algorithm to establish a shared secret between an initiator and a responder. The security of the DH and ECDH algorithms relies on the difficulty to solve a discrete logarithm

problem in multiplicative and elliptic curve groups respectively when the order of the group parameter is large enough. While solving such a problem remains difficult with current computing power, it is believed that general purpose quantum computers will be able to solve this problem, implying that the security of IKEv2 is compromised. There are, however, a number of cryptosystems that are conjectured to be resistant against quantum computer attack. This family of cryptosystems are known as post-quantum cryptography (PQC). It is sometimes also referred to as quantum-safe cryptography (QSC) or quantum-resistant cryptography (QRC).

1.2. Proposed Extension

This document describes a framework to integrate QSC for IKEv2, while maintaining backwards compatibility, to derive a set of IKE keys that have resistance to quantum computer attacks. Our framework allows the negotiation of one or more QSC algorithm to exchange data, in addition to the existing DH or ECDH key exchange data. We believe that the feature of using more than one post-quantum algorithm is important as many of these algorithms are relatively new and there may be a need to hedge the security risk with multiple key exchange data from several distinct QSC algorithms.

The secrets established from each key exchange are combined in a way such that should the post-quantum secrets not be present, the derived shared secret is equivalent to that of the standard IKEv2; on the other hand, a post-quantum shared secret is obtained if both classical and post-quantum key exchange data are present. This framework also applies to key exchanges in IKE Security Associations (SAs) for Encapsulating Security Payload (ESP) [RFC4303] or Authentication Header (AH) [RFC4302], i.e. Child SAs, in order to provide a stronger guarantee of forward security.

Some post-quantum key exchange payloads may have size larger than the standard MTU size, and therefore there could be issues with fragmentation at IP layer. IKE does allow transmission over TCP where fragmentation is not an issue [RFC8229]; however, we believe that a UDP-based solution will be required too. IKE does have a mechanism to handle fragmentation within UDP [RFC7383], however that is only applicable to messages exchanged after the IKE_SA_INIT. To use this mechanism, we use the INTERMEDIATE exchange as outlined in [I-D.smyslov-ipsecme-ikev2-aux]. With this mechanism, we do an initial key exchange, using a smaller, possibly non-quantum resistant primitive, such as ECDH. Then, before we do the IKE_AUTH exchange, we perform one or more INTERMEDIATE exchanges, each of which includes a secondary key exchange. As the INTERMEDIATE exchange is encrypted, the IKE fragmentation protocol RFC7383 can be used. The IKE SK values will be updated after each exchange, and so the final IKE SK

values will depend on all the key exchanges, hence they are secure if any of the key exchanges are secure.

Note that readers should consider the approach in this document as providing a long term solution in upgrading the IKEv2 protocol to support post-quantum algorithms. A short term solution to make IKEv2 key exchange quantum secure is to use post-quantum pre-shared keys as discussed in [I-D.ietf-ipsecme-qr-ikev2].

1.3. Changes

Changes in this draft in each version iterations.

draft-tjhai-ipsecme-hybrid-qske-ikev2-02

- o Use new transform types to negotiate additional key exchanges, rather than using the KE payloads of IKE SA.

draft-tjhai-ipsecme-hybrid-qske-ikev2-01

- o Use INTERMEDIATE to perform multiple key exchanges in succession.
- o Handle fragmentation by keeping the first key exchange (a standard IKE_SA_INIT with a few extra notifies) small, and encrypting the rest of the key exchanges.
- o Simplify the negotiation of the 'extra' key exchanges.

draft-tjhai-ipsecme-hybrid-qske-ikev2-00

- o We added a feature to allow more than one post-quantum key exchange algorithms to be negotiated and used to exchange a post-quantum shared secret.
- o Instead of relying on TCP encapsulation to deal with IP level fragmentation, we introduced a new key exchange payload that can be sent as multiple fragments within IKE_SA_INIT message.

1.4. Document Organization

The remainder of this document is organized as follows. Section 2 summarizes design criteria. Section 3 describes how post-quantum key exchange is performed between two IKE peers and how keying materials are derived for both SAs and child SAs. A summary of alternative approaches that have been considered, but later discarded, are described in Section 4. Section 5 discusses IANA considerations for the namespaces introduced in this document, and lastly Section 6 discusses security considerations.

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

2. Design Criteria

The design of the proposed post-quantum IKEv2 is driven by the following criteria:

- 1) Need for post-quantum cryptography in IPsec. Quantum computers might become feasible in the next 5-10 years. If current Internet communications are monitored and recorded today (D), the communications could be decrypted as soon as a quantum-computer is available (e.g., year Q) if key negotiation only relies on non post-quantum primitives. This is a high threat for any information that must remain confidential for a long period of time $T > Q - D$. The need is obvious if we assume that Q is 2040, D is 2020, and T is 30 years. Such a value of T is typical in classified or healthcare data.
- 2) Hybrid. Currently, there does not exist a post-quantum key exchange that is trusted at the level that ECDH is trusted against conventional (non-quantum) adversaries. A hybrid approach allows introducing promising post-quantum candidates next to well-established primitives, since the overall security is at least as strong as each individual primitive.
- 3) Focus on quantum-resistant confidentiality. A passive attacker can eavesdrop on IPsec communication today and decrypt it once a quantum computer is available in the future. This is a very serious attack for which we do not have a solution. An attacker can only perform active attacks such as impersonation of the communicating peers once a quantum computer is available, sometime in the future. Thus, our design focuses on quantum-resistant confidentiality due to the urgency of this problem. This document does not address quantum-resistant authentication since it is less urgent at this stage.
- 4) Limit amount of exchanged data. The protocol design should be such that the amount of exchanged data, such as public-keys, is kept as small as possible even if initiator and responder need to agree on a hybrid group or multiple public-keys need to be exchanged.
- 5) Future proof. Any cryptographic algorithm could be potentially

broken in the future by currently unknown or impractical attacks: quantum computers are merely the most concrete example of this. The design does not categorize algorithms as "post-quantum" or "non post-quantum" and does not create assumptions about the properties of the algorithms, meaning that if algorithms with different properties become necessary in the future, this framework can be used unchanged to facilitate migration to those algorithms.

- 6) Limited amount of changes. A key goal is to limit the number of changes required when enabling a post-quantum handshake. This ensures easier and quicker adoption in existing implementations.
- 7) Localized changes. Another key requirement is that changes to the protocol are limited in scope, in particular, limiting changes in the exchanged messages and in the state machine, so that they can be easily implemented.
- 8) Deterministic operation. This requirement means that the hybrid post-quantum exchange, and thus, the computed key, will be based on algorithms that both client and server wish to support.
- 9) Fragmentation support. Some PQC algorithms could be relatively bulky and they might require fragmentation. Thus, a design goal is the adaptation and adoption of an existing fragmentation method or the design of a new method that allows for the fragmentation of the key shares.
- 10) Backwards compatibility and interoperability. This is a fundamental requirement to ensure that hybrid post-quantum IKEv2 and a non-post-quantum IKEv2 implementations are interoperable.
- 11) FIPS compliance. IPsec is widely used in Federal Information Systems and FIPS certification is an important requirement. However, algorithms that are believed to be post-quantum are not FIPS compliant yet. Still, the goal is that the overall hybrid post-quantum IKEv2 design can be FIPS compliant.

3. The Framework of Hybrid Post-Quantum Key Exchange

3.1. Overall design

This design assigns new group identifiers (Transform Type 4) to the various post-quantum key exchanges (which will be defined later). We specifically do not make a distinction between classical (DH and ECDH) and post-quantum key exchanges, nor post-quantum algorithms which are true key exchanges versus post-quantum algorithms that act as key transport mechanisms; all are treated equivalently by the

protocol. In order to support both hybrid key exchanges (that is, relying on distinct key exchanges) and fragmentation, the proposed hybrid post-quantum IKEv2 protocol extends IKE [RFC7296] by adding additional key exchange messages (INTERMEDIATE) between the IKE_SA_INIT and the IKE_AUTH exchanges. In order to minimize communication overhead, only the key shares that are agreed to be used are actually exchanged. In order to achieve this, the IKE_SA_INIT exchange now includes notify payloads that negotiate the extra key exchanges to be used. The initiator IKE_SA_INIT message includes a notify that lists the extra key exchange policy required by the initiator; the responder selects one of the listed policies, and includes that as a notify in the response IKE_SA_INIT message. Then, the initiator and the responder perform one (or possibly more) INTERMEDIATE exchange; each such exchange includes a KE payload for the key exchange that was negotiated.

Here is an overview of the initial exchanges:

Initiator	Responder

<-- IKE_SA_INIT	(and extra key exchange negotiation) -->
<-- {INTERMEDIATE	(hybrid post-quantum key exchange)} -->
...	
<-- {INTERMEDIATE	(hybrid post-quantum key exchange)} -->
<-- {IKE_AUTH} -->	

The extra post-quantum key exchanges can use algorithms that are currently considered to be resistant to quantum computer attacks. These algorithms are collectively referred to as post-quantum algorithms in this document.

Most post-quantum key agreement algorithms are relatively new, and thus are not fully trusted. There are also many proposed algorithms, with different trade-offs and relying on different hard problems. The concern is that some of these hard problems may turn out to be easier to solve than anticipated (and thus the key agreement algorithm not be as secure as expected). A hybrid solution allows us to deal with this uncertainty by combining a classical key exchanges with a post-quantum one, as well as leaving open the possibility of multiple post-quantum key exchanges.

The method that we use to perform hybrid key exchange also addresses the fragmentation issue. The initial IKE_INIT messages do not have any inherent fragmentation support within IKE; however that can include a relatively short KE payload (e.g. one for group 14, 19 or 31). The rest of the KE payloads are encrypted within INTERMEDIATE

messages; because they are encrypted, the standard IKE fragmentation solution [RFC7383] is available.

3.2. Overall Protocol

In the simplest case, the initiator is happy with a single key exchange (and has no interest in supporting multiple), and he is not concerned with possible fragmentation of the IKE_SA_INIT messages (either because the key exchange he selects is small enough not to fragment, or he is confident that fragmentation will be handled either by IP fragmentation, or transport via TCP). In the following we overview the two protocol rounds involved in the hybrid post-quantum protocol.

In this case, the initiator performs the IKE_SA_INIT as standard, inserting this preferred key exchange (which is possibly a post-quantum algorithm) as the listed Transform Type 4, and including the initiator KE payload. If the responder accepts the policy, he responds with an IKE_SA_INIT response, and IKE continues as usual.

If the initiator desires to negotiate multiple key exchanges, or he needs IKE to handle any possible fragmentation, then he uses the protocol listed below.

3.2.1. IKE_SA_INIT Round: Negotiation

Multiple key exchanges are negotiated using the standard IKEv2 mechanism, via SA payload. For this purpose several new transform types, namely Additional Key Exchange 1, Additional Key Exchange 2, Additional Key Exchange 3, etc., are defined. They are collectively called Additional Key Exchanges and have slightly different semantics than existing IKEv2 transform types. They are interpreted as additional key exchanges that peers agreed to perform in a series of INTERMEDIATE exchanges. The possible transform IDs for these transform types are the same as IDs for the transform type 4 (Diffie-Hellman Group), so they all share a single IANA registry for transform IDs.

Key exchange method negotiated via transform type 4 MUST always take place in the IKE_SA_INIT exchange. Additional Key Exchanges negotiated via newly defined transforms MUST take place in series of INTERMEDIATE exchanges, in an order of the values of their transform types, so that key exchange negotiated using transform type N always precedes that of transform type N + 1. Each INTERMEDIATE exchange MUST bear exactly one key exchange method. Note that with this semantics, Additional Key Exchanges transforms are not associated with any particular type of key exchange and don't have any specific per transform type transform ID IANA registry. Instead they all

share a single registry for transform IDs - "Diffie-Hellman Group Transform IDs", as well as Transform Type 4. All new key exchange algorithms (both classical or quantum safe) should be added to this registry. This approach gives peers flexibility in defining the ways they want to combine different key exchange methods.

When forming a proposal the initiator adds transforms for the IKE_SA_INIT exchange using transform type 4. In most cases they will contain classical key exchange methods, however it is not a requirement. Additional key exchange methods are proposed using Additional Key Exchanges transform types. All these transform types are optional, the initiator is free to select any of them for proposing additional key exchange methods. Consequently, if none of Additional Key Exchanges are included in the proposal, then this proposal indicates performing standard IKEv2, as defined in [RFC7296]. If the initiator includes any transform of type N (where N is among Additional Key Exchanges) in the proposal, the responder MUST select one of the algorithms proposed using this type. A transform ID NONE may be added to those transform types which contain key exchange methods that the initiator believes are optional.

The responder performs negotiation using standard IKEv2 procedure described in Section 3.3 of [RFC7296]. However, for the Additional Key Exchange types the responder's choice MUST NOT contain equal transform IDs (apart from NONE), and the ID selected for Transform Type 4 MUST NOT appear in any of Additional Key Exchange transforms. In other words, all selected key exchange methods must be different.

3.2.2. INTERMEDIATE Round: Additional Key Exchanges

For each extra key exchange agreed to in the IKE_SA_INIT exchange, the initiator and the responder perform an INTERMEDIATE exchange, as described in [I-D.smyslov-ipsecme-ikev2-aux].

This exchange is as follows:

Initiator		Responder

HDR, SK {Ni2, KEi2}	-->	
	<--	HDR, SK {Nr2, KEr2}

The initiator sends a nonce in the Ni2 payload, and the key exchange payload in the KEi2; the group id of the KEi2 payload MUST match the negotiated extra key exchange. This packet is encrypted with the current IKE SK keys.

On receiving this, the responder sends a nonce in the Nr2 payload, and the key exchange payload KER2; again, this packet is encrypted with the current IKE SA keys.

Once this exchange is done, then both sides compute an updated keying material:

$$\text{SKEYSEED} = \text{prf}(\text{SK_d}(\text{old}), \text{KE2result} \mid \text{Ni2} \mid \text{Nr2})$$

where KE2result is the shared secret of the key exchange. Then, SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, SK_pr are updated as:

$$\begin{aligned} &\{\text{SK_d} \mid \text{SK_ai} \mid \text{SK_ar} \mid \text{SK_ei} \mid \text{SK_er} \mid \text{SK_pi} \mid \text{SK_pr}\} \\ &= \text{prf+}(\text{SKEYSEED}, \text{Ni2} \mid \text{Nr2} \mid \text{SPIi} \mid \text{SPIr}) \end{aligned}$$

Note that the negotiated transform types (the encryption type, hash type, prf type) are not modified.

Both the initiator and the responder will use this updated key values for the next message.

3.2.3. IKE_AUTH Exchange

After the INTERMEDIATE exchanges have completed, then the initiator and the responder will perform an IKE_AUTH exchange. This exchange is the standard IKE exchange, except that the initiator and responder signed octets are modified as described in [I-D.smyslov-ipsecme-ikev2-aux].

3.2.4. CREATE_CHILD_SA Exchange

The CREATE_CHILD_SA exchange is used in IKEv2 for the purpose of creating additional Child SAs, rekeying them and rekeying IKE SA itself. When creating or rekeying Child SAs, the peers may optionally perform a Diffie-Hellmann key exchange to add a fresh entropy into the session keys, in case of IKE SA rekeying, the key exchange is mandatory.

If the IKE SA was created using multiple key exchange methods, the peers may want continue using multiple key exchanges in the CREATE_CHILD_SA exchange too. If the initiator includes any Additional Key Exchanges transform in the SA payload (along with Transform Type 4) and the responder agrees to perform additional key exchanges, then the additional key exchanges are performed in a series of the INFORMATIONAL exchanges that follows the CREATE_CHILD_SA exchange in an order of the values of their transform types, so that key exchange negotiated using transform type N always precedes key exchange negotiated using transform type N + 1. Each

INFORMATIONAL exchange MUST bear exactly one key exchange method. Key exchange negotiated via Transform Type 4 always takes place in the CREATE_CHILD_SA exchange, as per IKEv2 specification.

Since after IKE SA is created the window size may be greater than one, and multiple concurrent exchanges may be active, it is essential to link the INFORMATIONAL exchanges together and with the CREATE_CHILD_SA exchange. A new status type notification ADDITIONAL_KEY_EXCHANGE is used for this purpose. Its Notify Message Type is <TBA by IANA>, Protocol ID and SPI Size are both set to 0. The data associated with this notification is a blob meaningful only to the responder, so that the responder can correctly link successive exchanges. For the initiator the content of this notification is an opaque blob.

The responder MUST include this notification in a CREATE_CHILD_SA or INFORMATIONAL response message in case next exchange is expected, filling it with some data that would allow linking this exchange to the next one. The initiator MUST copy the received notification with its content intact into the request message of the next exchange.

Below is an example of three additional key exchanges.

Initiator	Responder

HDR(CREATE_CHILD_SA), SK {SA, Ni, KEi} -->	<-- HDR(CREATE_CHILD_SA), SK {SA, Nr, KEr, N(ADDITIONAL_KEY_EXCHANGE) (link1)}
HDR(INFORMATIONAL), SK {Ni2, KEi2, N(ADDITIONAL_KEY_EXCHANGE) (link1)} -->	<-- HDR(INFORMATIONAL), SK {Nr2, KEr2, N(ADDITIONAL_KEY_EXCHANGE) (link2)}
HDR(INFORMATIONAL), SK {Ni3, KEi3, N(ADDITIONAL_KEY_EXCHANGE) (link2)} -->	<-- HDR(INFORMATIONAL), SK {Nr3, KEr3, N(ADDITIONAL_KEY_EXCHANGE) (link3)}
HDR(INFORMATIONAL), SK {Ni4, KEi4, N(ADDITIONAL_KEY_EXCHANGE) (link3)} -->	<-- HDR(INFORMATIONAL), SK {Nr4, KEr4}

4. Alternative Design

This section gives an overview on a number of alternative approaches that we have considered, but later discarded. These approaches are:

- o Sending the classical and post-quantum key exchanges as a single transform

We considered combining the various key exchanges into a single large KE payload; this effort is documented in a previous version of this draft (draft-tjhai-ipsecme-hybrid-qske-ikev2-01). This does allow us to cleanly apply hybrid key exchanges during the child SA; however it does add considerable complexity, and requires an independent fragmentation solution.

- o Sending post-quantum proposals and policies in KE payload only

With the objective of not introducing unnecessary notify payloads, we considered communicating the hybrid post-quantum proposal in the KE payload during the first pass of the protocol exchange. Unfortunately, this design is susceptible to the following downgrade attack. Consider the scenario where there is an MitM attacker sitting between an initiator and a responder. The initiator proposes, through SAI payload, to use a hybrid post-quantum group and as a backup a Diffie-Hellman group, and through KEi payload, the initiator proposes a list of hybrid post-quantum proposals and policies. The MitM attacker intercepts this traffic and replies with N(INVALID_KE_PAYLOAD) suggesting to downgrade to the backup Diffie-Hellman group instead. The initiator then resends the same SAI payload and the KEi payload containing the public value of the backup Diffie-Hellman group. Note that the attacker may forward the second IKE_SA_INIT message only to the responder, and therefore at this point in time, the responder will not have the information that the initiator prefers the hybrid group. Of course, it is possible for the responder to have a policy to reject an IKE_SA_INIT message that (a) offers a hybrid group but not offering the corresponding public value in the KEi payload; and (b) the responder has not specifically acknowledged that it does not supported the requested hybrid group. However, the checking of this policy introduces unnecessary protocol complexity. Therefore, in order to fully prevent any downgrade attacks, using KE payload alone is not sufficient and that the initiator MUST always indicate its preferred post-quantum proposals and policies in a notify payload in the subsequent IKE_SA_INIT messages following a N(INVALID_KE_PAYLOAD) response.

- o New payload types to negotiate hybrid proposal and to carry post-quantum public values

Semantically, it makes sense to use a new payload type, which mimics the SA payload, to carry a hybrid proposal. Likewise, another new payload type that mimics the KE payload, could be used to transport hybrid public value. Although, in theory a new

payload type could be made backwards compatible by not setting its critical flag as per Section 2.5 of RFC7296, we believe that it may not be that simple in practice. Since the original release of IKEv2 in RFC4306, no new payload type has ever been proposed and therefore, this creates a potential risk of having a backward compatibility issue from non-conforming RFC IKEv2 implementations. Since we could not see any other compelling advantages apart from a semantic one, we use the existing transform type and notify payloads instead. In fact, as described above, we use the KE payload in the first IKE_SA_INIT request round and the notify payload to carry the post-quantum proposals and policies. We use one or more of the existing KE payloads to carry the hybrid public values.

- o Hybrid public value payload

One way to transport the negotiated hybrid public payload, which contains one classical Diffie-Hellman public value and one or more post-quantum public values, is to bundle these into a single KE payload. Alternatively, these could also be transported in a single new hybrid public value payload, but following the same reasoning as above, this may not be a good idea from a backward compatibility perspective. Using a single KE payload would require an encoding or formatting to be defined so that both peers are able to compose and extract the individual public values. However, we believe that it is cleaner to send the hybrid public values in multiple KE payloads--one for each group or algorithm. Furthermore, at this point in the protocol exchange, both peers should have indicated support of handling multiple KE payloads.

- o Fragmentation

Handling of large IKE_SA_INIT messages has been one of the most challenging tasks. A number of approaches have been considered and the two prominent ones that we have discarded are outlined as follows.

The first approach was to treat the entire IKE_SA_INIT message as a stream of bytes, which we then split it into a number of fragments, each of which is wrapped onto a payload that would fit into the size of the network MTU. The payload that wraps each fragment is a new payload type and it was envisaged that this new payload type will not cause a backward compatibility issue because at this stage of the protocol, both peers should have indicated support of fragmentation in the first pass of the IKE_SA_INIT exchange. The negotiation of fragmentation is performed using a notify payload, which also defines supporting parameters such as the size of fragment in octets and the fragment identifier. The

new payload that wraps each fragment of the messages in this exchange is assigned the same fragment identifier. Furthermore, it also has other parameters such as a fragment index and total number of fragments. We decided to discard this approach due to its blanket approach to fragmentation. In cases where only a few payloads need to be fragmented, we felt that this approach is overly complicated.

Another idea that was discarded was fragmenting an individual payload without introducing a new payload type. The idea was to use the 9-th bit (the bit after the critical flag in the RESERVED field) in the generic payload header as a flag to mark that this payload is fragmented. As an example, if a KE payload is to be fragmented, it may look as follows.

1																2																3															
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																
Next Payload									C	F	RESERVED									Payload Length																											
Diffie-Hellman Group Number																Fragment Identifier																															
Fragment Index																Total Fragments																															
Total KE Payload Data Length																																															
Fragmented KE Payload																																															

When the flag F is set, this means the current KE payload is a fragment of a larger KE payload. The Payload Length field denotes the size of this payload fragment in octets--including the size of the generic payload header. The two-octet RESERVED field following Diffie-Hellman Group Number was to be used as a fragment identifier to help assembly and disassembly of fragments. The Fragment Index and Total Fragments fields are self-explanatory. The Total KE Payload Data Length indicates the size of the assembled KE payload data in octets. Finally, the actual fragment is carried in Fragment KE Payload field.

We discarded this approach because we believe that the working group may not be happy using the RESERVED field to change the format of a packet and that implementers may not like the complexity added from checking the fragmentation flag in each received payload. More importantly, fragmenting the messages in this way may leave the system to be more prone to denial of

service (DoS) attacks. By using INTERMEDIATE to transport the large post-quantum key exchange payloads, there is no longer any issue with fragmentation.

- o Group sub-identifier

As discussed before, each group identifier is used to distinguish a post-quantum algorithm. Further classification could be made on a particular post-quantum algorithm by assigning additional value alongside the group identifier. This sub- identifier value may be used to assign different security parameter sets to a given post-quantum algorithm. However, this level of details does not fit the principles of the document where it should deal with generic hybrid key exchange protocol, not a specific ciphersuite. Furthermore, there are enough Diffie- Hellman group identifiers should this be required in the future.

5. IANA Considerations

This document also adds the following Transform Types to the "Transform Type Values" registry:

Type	Description	Used In	Reference
6	Additional Key Exchange 1	(optional in IKE, AH and ESP)	[RFCXXXX]
7	Additional Key Exchange 2	(optional in IKE, AH and ESP)	[RFCXXXX]
8	Additional Key Exchange 3	(optional in IKE, AH and ESP)	[RFCXXXX]
9	Additional Key Exchange 4	(optional in IKE, AH and ESP)	[RFCXXXX]
10	Additional Key Exchange 5	(optional in IKE, AH and ESP)	[RFCXXXX]
11	Additional Key Exchange 6	(optional in IKE, AH and ESP)	[RFCXXXX]
12	Additional Key Exchange 7	(optional in IKE, AH and ESP)	[RFCXXXX]

This document also defines a new Notify Message Types in the "Notify Message Types - Status Types" registry:

<TBA> ADDITIONAL_KEY_EXCHANGE

6. Security Considerations

The key length of the Encryption Algorithm (Transform Type 1), the Pseudorandom Function (Transform Type 2) and the Integrity Algorithm (Transform Type 3), all have to be of sufficient length to prevent attacks using Grover's algorithm [GROVER]. In order to use the extension proposed in this document, the key lengths of these transforms SHALL be at least 256 bits long in order to provide sufficient resistance to quantum attacks. Accordingly the post-quantum security level achieved is at least 128 bits.

SKEYSEED is calculated from shared, KEx, using an algorithm defined in Transform Type 2. While a quantum attacker may learn the value of KEx', if this value is obtained by means of a classical key exchange, other KEx values generated by means of a quantum-resistant algorithm ensure that the final SKEYSEED is not compromised. This assumes that the algorithm defined in the Transform Type 2 is post-quantum.

The main focus of this document is to prevent a passive attacker performing a "harvest and decrypt" attack. In other words, an attacker that records messages exchanges today and proceeds to decrypt them once he owns a quantum computer. This attack is prevented due to the hybrid nature of the key exchange. Other attacks involving an active attacker using a quantum-computer are not completely solved by this document. This is for two reasons.

The first reason is because the authentication step remains classical. In particular, the authenticity of the SAs established under IKEv2 is protected using a pre-shared key, RSA, DSA, or ECDSA algorithms. Whilst the pre-shared key option, provided the key is long enough, is post-quantum, the other algorithms are not. Moreover, in implementations where scalability is a requirement, the pre-shared key method may not be suitable. Quantum-safe authenticity may be provided by using a quantum-safe digital signature and several quantum-safe digital signature methods are being explored by IETF. For example, if the implementation is able to reliably track state, the hash based method, XMSS has the status of an RFC, see [RFC8391]. Currently, quantum-safe authentication methods are not specified in this document, but are planned to be incorporated in due course.

It should be noted that the purpose of post-quantum algorithms is to provide resistance to attacks mounted in the future. The current threat is that encrypted sessions are subject to eavesdropping and archived with decryption by quantum computers taking place at some point in the future. Until quantum computers become available there is no point in attacking the authenticity of a connection because there are no possibilities for exploitation. These only occur at the time of the connection, for example by mounting a MitM attack. Consequently there is not such a pressing need for quantum-safe authenticity.

This draft does not attempt to address key exchanges with KE payloads longer than 64k; the current IKE payload format does not allow that as a possibility. If such huge KE payloads are required, a work around (such as making the KE payload a URL and a hash of the real payload) would be needed. At the current time, it appears likely that there will be plenty of key exchanges available that would not require such a workaround.

7. References

7.1. Normative References

- [I-D.smyslov-ipsecme-ikev2-aux]
Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-smyslov-ipsecme-ikev2-aux-02 (work in progress), December 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>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

7.2. Informative References

- [GROVER] Grover, L., "A Fast Quantum Mechanical Algorithm for Database Search", Proc. of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996), 1996.
- [I-D.ietf-ipsecme-qr-ikev2]
Fluhrer, S., McGrew, D., Kampanakis, P., and V. Smyslov, "Postquantum Preshared Keys for IKEv2", draft-ietf-ipsecme-qr-ikev2-05 (work in progress), December 2018.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <<https://www.rfc-editor.org/info/rfc7383>>.

- [RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <<https://www.rfc-editor.org/info/rfc8229>>.
- [RFC8391] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <<https://www.rfc-editor.org/info/rfc8391>>.

Acknowledgements

The authors would like to thanks Frederic Detienne and Olivier Pelerin for their comments and suggestions, including the idea to negotiate the post-quantum algorithms using the existing KE payload.

Authors' Addresses

C. Tjhai
Post-Quantum

Email: cjt@post-quantum.com

M. Tomlinson
Post-Quantum

Email: mt@post-quantum.com

G. Bartlett
Cisco Systems

Email: grbartle@cisco.com

S. Fluhrer
Cisco Systems

Email: sfluhrer@cisco.com

D. Van Geest
ISARA Corporation

Email: daniel.vangeest@isara.com

O. Garcia-Morchon
Philips

Email: oscar.garcia-morchon@philips.com

Valery Smyslov
ELVIS-PLUS

Email: svan@elvis.ru