Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 10, 2017

V. Smyslov
ELVIS-PLUS

October 7, 2016

Compact Format of IKEv2 Payloads
draft-smyslov-ipsecme-ikev2-compact-00

Abstract

This document describes a method for reducing the size of the Internet Key Exchange version 2 (IKEv2) messages by using an alternate format for IKE payloads. Standard format of many IKE payloads contains a lot of redundancy. This document takes advantage of this fact and specifies a way to eliminate some redundancy by using more dense encoding. Reducing size of IKEv2 messages is desirable for low power consumption battery powered devices. It also helps to avoid IP fragmentation of IKEv2 messages.

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 April 10, 2017.

Table of Contents

1. Introduction

The Internet Key Exchange protocol version 2 (IKEv2) specified in [RFC7296] is used in the IP Security (IPsec) architecture for the purposes of Security Association (SA) parameters negotiation and authenticated key exchange. The protocol uses UDP as the transport for its messages, which size varies from less than one hundred bytes to several kBytes.

Decreasing the size of IKEv2 messages is highly desirable for the Internet of Things (IoT) devices utilizing a lower power consumption technology. For some of such devices the power consumption for transmitting extra bits over network is prohibitively high (see appendix A of [IPSEC-IOT-REQS]). Many such devices are battery powered without an ability to recharge or to replace the battery which serves for the life cycle of the device (often a few years). For this reason the task of reducing the power consumption for such devices is very important and decreasing messages size is one of the ways to accomplish it.

Large UDP messages may also cause fragmentation on IP level, which may interact badly with Network Address Translators (NAT). In particular, some NATs drop IP fragments that don't contain TCP/UDP port numbers (non-initial IP fragments), so that the IP datagram cannot be reassembled on the receiving end and IKE SA cannot be established. One of the possible solutions to the problem is IKEv2 fragmentation [RFC7383]. However, the IKEv2 fragmentation can only be applied to encrypted messages and thus cannot be used in the initial IKEv2 exchange, IKE_SA_INIT. Usually the IKE_SA_INIT messages are relatively small and don't cause IP fragmentation. However, while more and more new algorithms and protocol extensions are included in IKEv2, these messages become larger and larger thus increasing the risk of IP fragmentation. It is desirable to make IKEv2 messages more compact to help avoid this risk.

IKEv2 messages are comprised of data structures called payloads. Each payload consists of a common part (Generic Payload Header) followed by a payload-specific data which is formatted differently depending on the payload's purpose. Section 3 of [RFC7296] lists formats for all standard IKEv2 payloads. As one can see some IKEv2 payloads are formatted in such a way, that there are substantial redundancy in their encoding. This document defines an alternative format for the IKEv2 payloads, which provides more dense encoding, that allows making IKEv2 messages more compact.

2. Requirements Language

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

3. Overview

The idea behind the protocol is to develop an alternate compact encoding of IKEv2 payloads that meets the following requirements:

1. The compact encoding must be applicable to all already defined IKEv2 payloads as well as to future payloads, so it must not depend on payload format. There may be few special cases if specific encoding is justified by much higher efficiency.

2. The compact encoding must be easily converted to standard encoding and visa versa. This allows implementations to re-use existing composing/parsing code and to apply compact encoding/ decoding as pre/post process steps in message processing.

3. The compact encoding/decoding algorithm must consume low resources in terms of code size, CPU consumption and memory footprint.

4. The compact encoding must never increase the payload size and must be effective enough to noticeably decrease the size of most payloads.

To meet these requirement the encoding algorithm must be independent from the particular payload format. In other words, it must take any given payload and perform some kind of compression. General purpose lossless compression algorithms are usually not very effective when they are applied to small amount of data. Fortunately format of many IKEv2 payloads has a peculiarity that allows to use simple and relatively efficient compression algorithm.

Many IKEv2 payloads contain a lot of zero octets. These octets come from the following sources:

- Many Payload Headers contain RESERVED fields that must be zeroed according to IKEv2 specification.

- Payload length is encoded in two octets, while most IKEv2 payloads are less than 256 octets in size.

- Substantial number of IKEv2 parameters are encoded in two octets, while the number of currently defined values for these parameters is less than one hundred (see [IKEV2-IANA]).

The idea is to omit these zero octets from the compact payload and supply a bitmap that will indicate which octets were omitted.

IKEv2 header also contains some redundancy - in particular the Length

field occupies four octets, while IKEv2 messages are extremely unlikely to exceed few Kb in size. However, some middleboxes perform an inspection of IKEv2 header and changing its format will likely interact badly with these middleboxes. Thus, the possible saving of few octets doesn't justify modification of IKEv2 header.

4. Compact Representation of IKEv2 Payloads

This document defines one generic compact format suitable for compacting any IKEv2 payload and two specific compact formats - one for SA payload and the other for Notify payload containing status notification with no data. The rationale for such a design is the following.

One common generic compact format simplifies implementations and allows to use it with any currently defined and not yet defined payloads. However, it doesn't provide high level of efficiency.

SA payload contains a lot of redundancy and can be encoded in highly efficient way. Moreover, this payload grows up quickly once more new transforms are defined and implementations start using them. In some cases the SA payload can be the largest payload in the IKE_SA_INIT exchange. So, there is a natural desire to encode it differently to gain better result. On the other hand, Notify payload with status notification containing no data is often used in IKEv2 initial exchange to negotiate support for various protocol extensions. So, there are usually several such payloads in the IKE_SA_INIT request and response messages, ant it is anticipated that their number will increase as long as more and more IKEv2 extensions are defined. That's why this payload is also encoded differently to make initial messages more compact.

4.1. Compact Generic Payload

Generic IKEv2 payload is depicted below (Figure 1) for convenience. It consists of generic payload header followed by payload data. Brief description of generic payload header fields is provided below, readers should refer to Section 3.2 of [RFC7296] for full description.

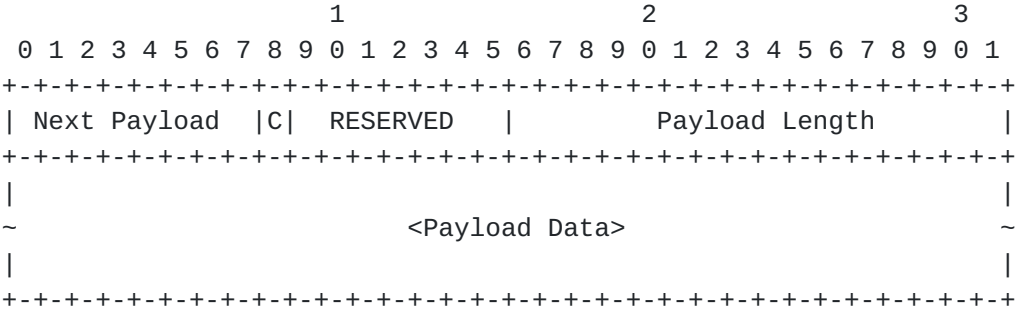

Figure 1: Generic Payload

- Next Payload (1 octet) - Identifier for the payload type of the next payload in the message.

- Critical (1 bit) - This bit is set if the current payload cannot be skipped in case the receiver doen't understand its type and cleared if it is safe to skip this payload.

- RESERVED (7 bits) - MUST be sent as zero; MUST be ignored on receipt.

- Payload Length (2 octets, unsigned integer) - Length in octets of the current payload, including the generic payload header.

- Payload Data (variable) - Contains payload specific data.

Some payloads have extended headers following the generic payload header. For the purpose of compact payload encoding any extended header is treated as part of payload data. Complete description of IKEv2 Payload formats can be found in Section 3 of [RFC7296].

Figure 2 shows generic compact payload format.

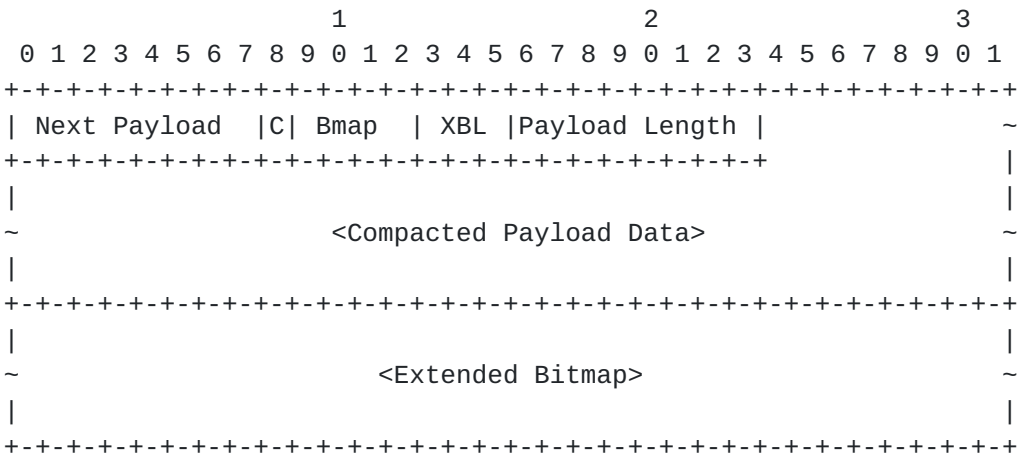

Figure 2: Compact Generic Payload

- Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. The value is taken intact from original payload.

- Critical (1 bit) - This bit is set if the current payload cannot be skipped in case the receiver doen't understand its type and cleared if it is safe to skip this payload. The value is taken intact from original payload.

- Bmap (4 bits) - Bitmap, contains bitmap where each bit indicates whether one of the four first octets of original Payload Data was zero and thus was omitted from Compacted Payload Data. If the bit is set, then the corresponding octet was zero and was omitted, if the bit is cleared, then either the corresponding octet was copied or it didn't present in the original Payload Data (in case the octet would be outside the original payload). The rightmost bit of the map corresponds to the first of the four octets and the leftmost bit corresponds to the last of these four octets.

- XBL (3 bits) - Extended Bitmap Length, specifies the number of Extended Bitmap octets plus one. Note, that by construction this field cannot be zero.

- Payload Length (1 octet) - Length in octets of compacted payload not including Extended Bitmap.

- Compacted Payload Data (variable) - Contains original payload data with some zero octets omitted. Up to 52 zero octets from the beginning of original payload data can be omitted, the rest of original payload data is always copied.

- Extended Bitmap (variable, 0 - 6 octets) - contains extended bitmap. Each octet of this bitmap represents how the corresponding eight octets of the original payload data were handled - original octets corresponding to the set bits were zero and thus were omitted from the compacted payload data, while octets corresponding to the cleared bits were copied. First octet in the Extended Bitmap corresponds to octets from 5 to 12 of the original payload data, next octet - to octets from 13 to 20, etc. Note, that the Bmap field indicates how octets from 1 to 4 were handled. In each octet of extended bitmap the rightmost bit represents the first of corresponding original octets and the leftmost bit represents the last one.

Any IKEv2 payload can be compacted if it meets two requirements:

- The payload length is less than 256 octets. In other words, the high order octet of Payload Length field is zero.

- The RESERVED field in the generic payload header is zero. Section 3.2 of [RFC7296] requires that this field must always be zero when preparing payload.

If payload doesn't meet these requirement it is included into the message without modification. The regular and compact payloads can be present in any order in the compact IKEv2 message. The receiving side can distinguish between them by analyzing the least significant

three bits of RESERVED field in generic payload header. These bits correspond to XBL field in compact generic payload header, and this field will always be non-zero in compact payload while these bits are required to be zero in regular payload.

If payload meets the above requirements then it is compacted as follows.

1. The Next Payload and Critical fields are copied from original payload.

2. The first 4 octets of the original payload data are scanned one by one. If the current octet is zero, then it is omitted from the Compacted Payload data and the corresponding bit (starting from the rightmost bit to the leftmost bit) in Bmap field is set. Otherwise the octet is copied and the corresponding bit is cleared. If there are no more octets in the original payload data then the process stops and the rest of Bmap is zeroed.

3. If any more octets left in the original payload data, then the next 48 of them are scanned one by one. Since the Extended Bitmap position isn't known yet a temporary bitmap of six octets is used. If the current octet is zero, then it is omitted from the Compacted Payload Data and the corresponding bit (starting from the rightmost bit to the leftmost bit) in the corresponding octet of a temporary bitmap is set. Otherwise the original octet is copied and the corresponding bit in temporary bitmap is cleared. If there are no more octets in the original payload data then the process stops and the rest bits of the current temporary bitmap octet are zeroed..

4. If any more octets left in the original payload data, they are copied to the Compacted Payload Data.

5. The Payload Length field is set to indicate the size of Compacted Payload Data plus 3. It will indicate the offset of the Extended Bitmap from the beginning of compact payload.

6. The content of the temporary bitmap is copied to the Extended Bitmap field (after the Compacted Payload Data). The temporary bitmap octets are copied one by one up to the first zero octet (if any). In other words, the Extended Bitmap MUST NOT contain zero octets.

7. The XBL field is set to the number of octets placed in the Extended Bitmap plus one.

4.2. Compact SA Payload

SA payload is compacted differently than generic IKEv2 payload. The specific format for Compact SA payload allows to get very high level of compression. High level of compression is important, because SA payload grows up quickly as more and more cryptographic transforms are defined, get widespread adoption and advertised by Initiators.

Unlike generic compact payload, which retains its payload type and is distinguished from regular IKEv2 payload by analyzing the leftmost three bits of RESERVED field of the generic payload header, the Compact SA payload has its own payload type. The reason is that its format (Figure 3) is completely different and the above method cannot be used. The Compact SA payload is denoted CSA, and its payload type is <TBA by IANA>.

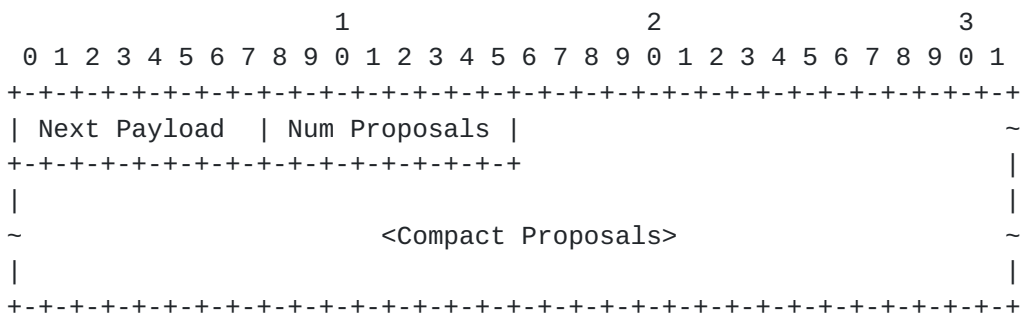

Figure 3: Compact SA Payload

- Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. The value is taken intact from original payload.

- Num Proposals (1 octet) - Specifies the number of Compact Proposals in this SA payload.

- Compact Proposals (variable) - One or more compact proposal substructures.

Despite the fact that Compact SA payload has different payload type and different format than regular SA payload, the associated semantics MUST be the same. Regular SA payload can always be compacted if it is compliant with Section 3.3 of [RFC7296].

4.2.1. Compact Proposal Substructure

Compact Proposal substructure (Figure 4) resembles regular Proposal substructure lacking first four octets. The Compact Proposal substructure fields are briefly described below. Readers should

refer to Section 3.3.1 of [RFC7296] for detailed description of Compact Proposal substructure fields with the same names, as in regular Proposal substructure.

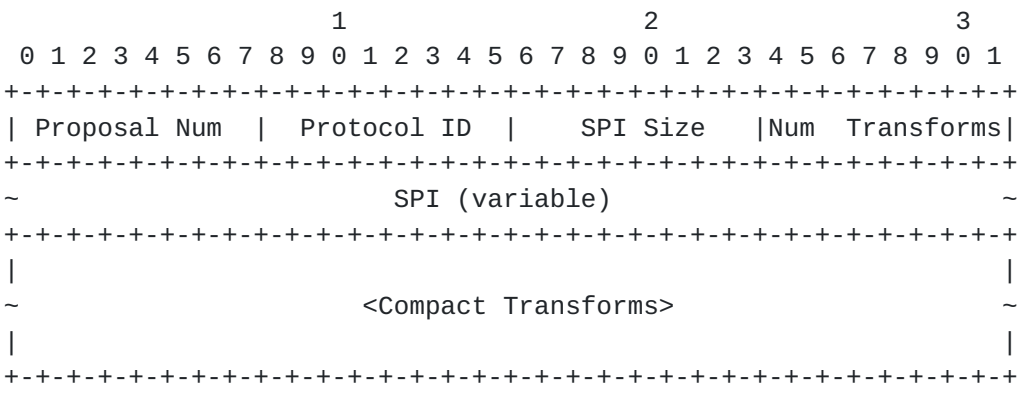

Figure 4: Compact Proposal Substructure

- Proposal Num (1 octet) - Proposal Number.
- Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation.
- SPI Size (1 octet) - Size of SPI.
- Num Transforms (1 octet) - Specifies the number of transforms in this proposal.
- SPI (variable) - The sending entity's SPI.
- Compact Transforms (variable) - One or more compact transform substructures.

4.2.2. Compact Transform Substructures

Compact Transform substructures are encoded differently depending on Transform Type, Transform ID and presence of attributes to get most effective encoding for common use cases. The leftmost bits of the first octet of the Compact Transform substructure are used to distinguish between different formats. These bits are called Tag. The table below shows how Tag value correlates with Compact Transform substructure format.

Tag	Compact Transform	Len	Format
00xxxxxx	Short form - Encryption (Type 1)	1	Figure 5
01xxxxxx	Short form - Encryption (Type 1) with key length	2	Figure 6
10xxxxxx	Short form - Diffie-Hellman Group (Type 4)	1	Figure 7
110xxxxx	Short form - Integrity (Type 3)	1	Figure 8
1110xxxx	Short form - Pseudo-random Function (Type 2)	1	Figure 9
11110xxx	Short form - Extended Sequence Numbers (Type 5)	1	Figure 10
11111xxx	Long form	3	Figure 11
11111000	Full form	var	Figure 12

Table 1: Tag values and corresponding Compact Transform formats

Short forms are the most efficient Compact Transform formats. Most of them occupy only one octet, except for the Encryption Transform with key length, which occupy two octets. The Transform can be encoded using short form if it meets the following requirements:

1. Transform Type is between 1 and 5. At the time the document was written these Transform Types were the only defined (see [IKEV2-IANA]).

2. Transform ID is less or equal to the value, specific to each Transform Type:

    * 63 for Transform Type 1 (Encryption Algorithm)

    * 15 for Transform Type 2 (Pseudo-random Function)

    * 31 for Transform Type 3 (Integrity Algorithm)

    * 63 for Transform Type 4 (Diffie-Hellman Group)

    * 7 for Transform Type 5 (Extended Sequence Numbers)

3. Transform has no attributes, or Transform has Type 1 (Encryption Algorithm), it has only one attribute of type Key Length (Type 14), the value of this attribute is less than 2048 and the value is divisible by 8.

If the Transform doesn't meet these requirements, then it cannot be encoded in short form. In this case either long form (Figure 11) or full form (Figure 12) must be used. Note, that all the transforms defined in [IKEV2-IANA] at the time this document was written meet these requirements.

Figures 5-10 show short form encodings for different Transform Types.

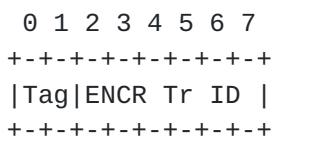

Figure 5: Compact Transform Substructure (Short Form: Encryption)

o Tag (2 bits) - MUST be 00.

o ENCR Tr ID (6 bits) - Encryption Algorithm Transform ID.

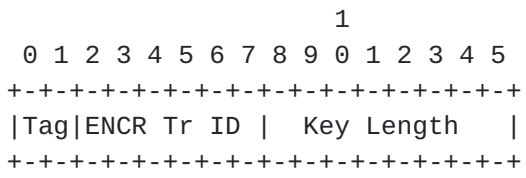

Figure 6: Compact Transform (Short Form: Encryption with Key Length)

o Tag (2 bits) - MUST be 01.

o ENCR Tr ID (6 bits) - Encryption Algorithm Transform ID.

o Key Length (1 octet) - Key Length in bytes (note, that Key Length attribute in IKEv2 specifies key length in bits).

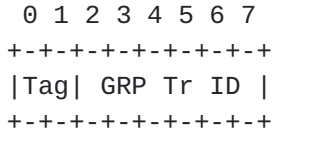

Figure 7: Compact Transform (Short Form: Diffie-Hellman Group)

- Tag (2 bits) - MUST be 10.

- GRP Tr ID (6 bits) - Diffie-Hellman Group Transform ID.

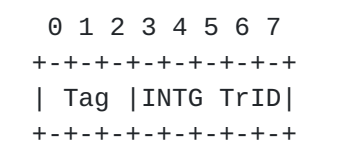

Figure 8: Compact Transform (Short Form: Integrity)

- Tag (3 bits) - MUST be 110.

- INTG TrID (5 bits) - Integrity Algorithm Transform ID.

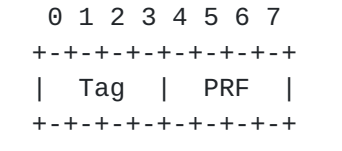

Figure 9: Compact Transform (Short Form: PRF)

- Tag (4 bits) - MUST be 1110.

- PRF (4 bits) - Pseudo-random Function Transform ID.

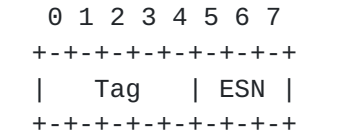

Figure 10: Compact Transform (Short Form: ESN)

- Tag (5 bits) - MUST be 11110.

- ESN (3 bits) - Extended Sequence Numbers Transform ID.

Long form (Figure 11) is used when Transform doesn't meet requirements for short form encoding, but still meets the following requirements:

1. Transform Type is between 1 and 7. At the time this document was written only Transform Types 1 to 5 were defined (see [IKEV2-IANA]).

2. Transform has no attributes.

Long form allows to effectively encode Transform IDs for standard Transform Types that don't fit into the short form, e.g. private Transform IDs (if these transforms don't have associated attributes). It is expected that new Transform IDs will be defined more often than new Transform Types. So, defining effective encoding for standard Transform Type and arbitrary Transform IDs makes sense.

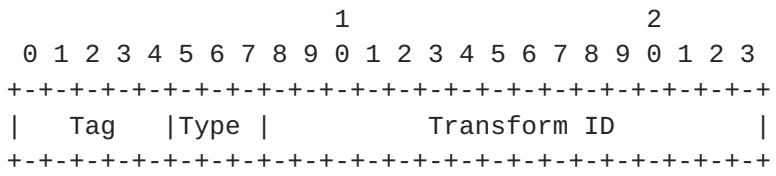

Figure 11: Compact Transform (Long Form)

- Tag (5 bits) - MUST be 11111.

- Transform Type (3 bits) - The type of transform being specified in this substructure.

- Transform ID (2 octets) - The specific instance of the Transform Type being specified.

Full encoding of Compact Transform substructure allows encoding of any transform without restrictions. It is used when transform cannot be encoded neither in short form nor in long form. The format (Figure 12) resembles regular Transform Substructure with all RESERVED fields removed. The Compact Transform substructure fields are briefly described below. Readers should refer to Section 3.3.2 of [RFC7296] for detailed description of Compact Transform substructure fields with the same names, as in regular Transform substructure.

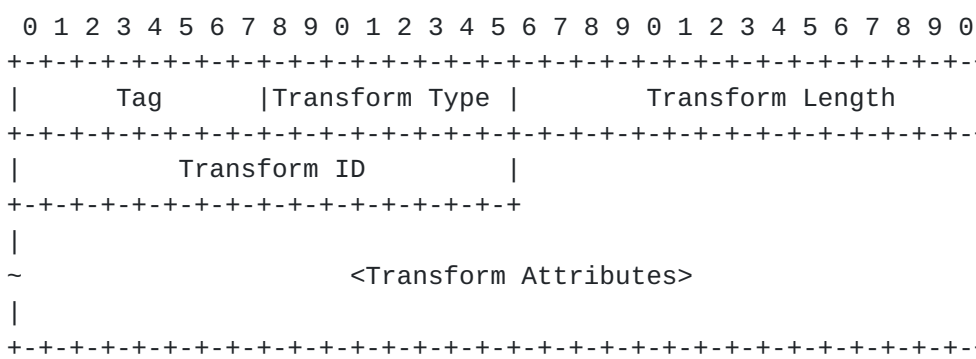

Figure 12: Compact Transform (Full Form)

- Tag (1 octet) - MUST be 11111000.

- Transform Type (1 octet) - The type of transform being specified in this substructure.

- Transform Length (2 octets, unsigned integer) - The length in octets of the Compact Transform Substructure including Header and Attributes.

- Transform ID (2 octets) - The specific instance of the Transform Type being specified.

- Transforms Attributes (variable) - Transform attributes.

4.3. Compact Notify Payload

Notify payloads containing status notifications with no data are often used in IKEv2. This is "de facto" standard way to negotiate various protocol extensions and for that reason usually several such Notify payloads are present in initial IKEv2 exchanges. It is anticipated that the number of IKEv2 extensions will increase and thus the size of initial exchange messages will increase too. This is the reason that this kind of Notify payload is encoded specifically, to get more effective message compression.

As well as Compact SA payload, Compact Notify payload has its own payload type. The Compact Notify payload is denoted CN, and its payload type is <TBA by IANA>.

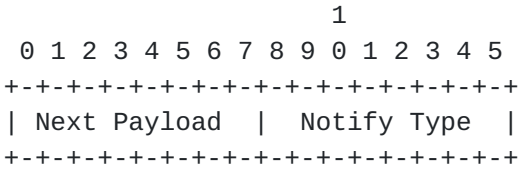

Figure 13: Compact Notify Payload for Status Notification

- Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. The value is taken intact from original payload.

- Notify Type (1 octet) - The type of notification message minus 16384.

Despite the fact that Compact Notify payload has different payload type and different format than regular Notify payload, the associated semantics MUST be the same.

Notify payload can be encoded as Compact Notify payload if it meets the following requirements (see Section 3.10 of [RFC7296] for Notify payload format):

1. Notify Message Type is between 16384 and 16639 (inclusive). This corresponds to status notifications.

2. Protocol ID is zero.

3. SPI Size is zero, meaning no SPI is present.

4. Notification Data is empty.

At the time this document was written about 40 percent of status notifications defined in [IKEV2-IANA] met these requirements. If Notify payload doesn't meet these requirements, the generic compact format (Section 4.1) can be tried.

5. Compact Format Negotiation

Most IKEv2 extensions are negotiated in the following way. The Initiator announces its support for some extension by including corresponding Vendor ID payload or Notify payload containing status notification in the request message of IKE_SA_INIT or IKE_AUTH exchanges. If the Responder supports this extension it returns the same (or some specific) payload to the Initiator in response message. Responder that doesn't support compact format just ignores these payloads in accordance with IKEv2 specification.

This method is inappropriate for negotiation of compact format, because Initiator should be able to send an initial request message in compact form and thus it must inform Responder somehow that the message must be parsed differently than regular IKEv2 message. Since compact message may contain regular IKEv2 payloads, it is possible to define a new status notification and to include it without compacting as the very first payload in the initial request. However, this spoils the whole idea of reducing initial messages size, since this payload will increase message size for no good reason.

Instead this document specifies a different negotiation mechanism. An alternative initial exchange is defined, ALT_IKE_SA_INIT. Initiator wishing to use compact representations of IKEv2 payloads MUST start creating IKEv2 SA using ALT_IKE_SA_INIT exchange instead of IKE_SA_INIT. The very first ALT_IKE_SA_INIT request may contain compact payloads. If Responder receives ALT_IKE_SA_INIT request and doesn't support compact format, then according to Section 2.21.1 of [RFC7296] it discards the request. It may also send INVALID_SYNTAX notification. For the Initiator receiving no response after several retransmissions or receiving INVALID_SYNTAX notification is an indication that the Responder doesn't support compact format. In this case the Initiator MAY restart initial request using regular IKE_SA_INIT request.

If the Responder supports compact format then it response with ALT_IKE_SA_INIT response, confirming to the Initiator that using compact format is successfully negotiated. Once using compact format is negotiated, compact payloads may appear in any message of subsequent exchanges in the context of the IKE SA being negotiated.

The semantics associated with ALT_IKE_SA_INIT exchange MUST be the same, as the semantics associated with IKE_SA_INIT exchange. In other words, when ALT_IKE_SA_INIT exchange is used, the endpoints must behave exactly as if it is IKE_SA_INIT exchange. The only difference (apart from different Exchange Types) is that IKE_SA_INIT messages MUST NOT contain compact payloads, while ALT_IKE_SA_INIT MAY contain them.

6. Interaction with other IKEv2 Extensions

To be added.

7. Security Considerations

To be added.

8. IANA Considerations

This document defines two new Payloads in the "IKEv2 Payload Types" registry:

<TBA>	Compact SA Payload	CSA
<TBA>	Compact Notify Payload	CN

This document also defines new Exchange Type in the "IKEv2 Exchange Types" registry:

<TBA>	ALT_IKE_SA_INIT

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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7383>.

[IKEV2-IANA] "Internet Key Exchange Version 2 (IKEv2) Parameters", <http://www.iana.org/assignments/ikev2-parameters>.

9.2. Informative References

[IPSEC-IOT-REQS] Migault, D., Guggemos, T., and C. Bormann, "Requirements for Diet-ESP the IPsec/ESP protocol for IoT", draft-mglt-6lo-diet-esp-requirements-02 (work in progress), July 2016.

Author's Address

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
RU

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