

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.

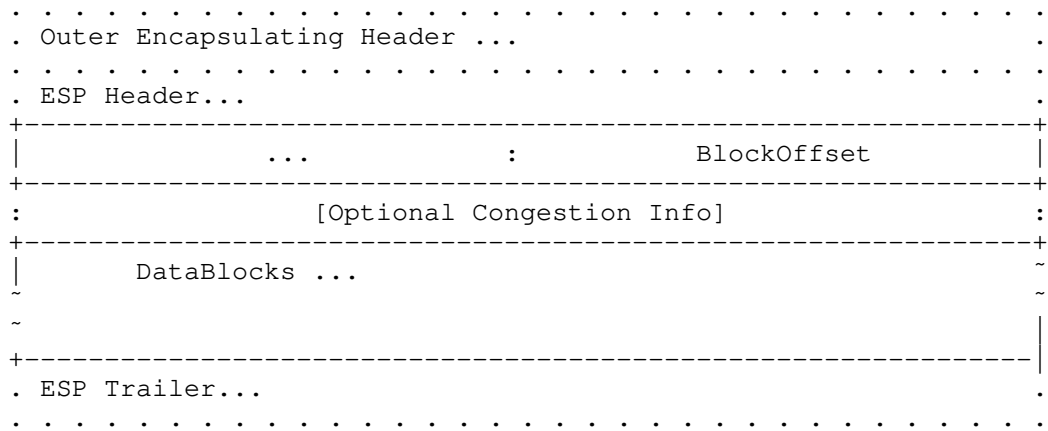


Figure 1: Layout of an IP-TFS IPsec Packet

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

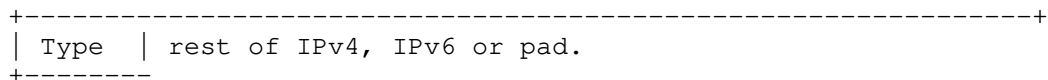


Figure 2: Layout of IP-TFS data block

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.

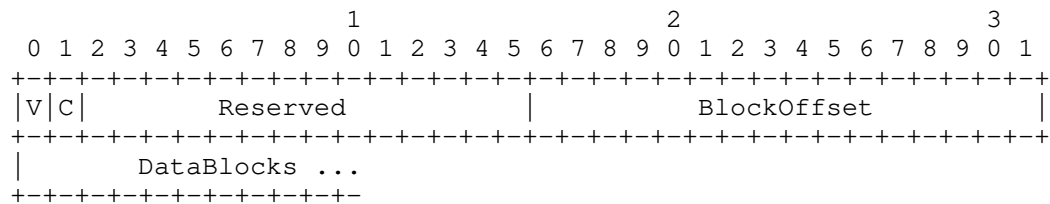
6. Packet and Data Formats

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

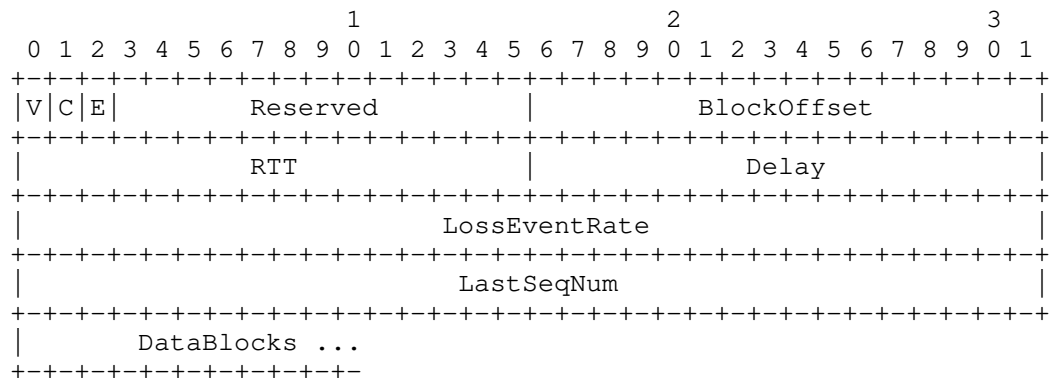
block being re-assembled. If the "BlockOffset" extends into subsequent packets it continues to only count subsequent "DataBlocks" data (i.e., it does not count subsequent packets non-"DataBlocks" octets).

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)

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

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

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

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
<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	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
<hr/>								
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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 September 2022

V. Smyslov
ELVIS-PLUS
5 March 2022

Intermediate Exchange in the IKEv2 Protocol
draft-ietf-ipsecme-ikev2-intermediate-10

Abstract

This document defines a new exchange, called Intermediate Exchange, for the Internet Key Exchange protocol Version 2 (IKEv2). This exchange can be used for transferring large amounts of data in the process of IKEv2 Security Association (SA) establishment. An example of the need to do this is using Quantum Computer resistant key exchange methods for IKE SA establishment. Introducing the Intermediate Exchange allows re-using the 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 <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 6 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
2. Terminology and Notation	4
3. Intermediate Exchange Details	4
3.1. Support for Intermediate Exchange Negotiation	4
3.2. Using Intermediate Exchange	4
3.3. The IKE_INTERMEDIATE Exchange Protection and Authentication	5
3.3.1. Protection of the IKE_INTERMEDIATE Messages	5
3.3.2. Authentication of the IKE_INTERMEDIATE Exchanges	6
3.4. Error Handling in the IKE_INTERMEDIATE Exchange	10
4. Interaction with other IKEv2 Extensions	11
5. Security Considerations	11
6. IANA Considerations	12
7. Implementation Status	13
8. Acknowledgements	13
9. References	13
9.1. Normative References	13
9.2. Informative References	14
Appendix A. Example of IKE_INTERMEDIATE exchange	14
Author's Address	16

1. Introduction

The Internet Key Exchange protocol version 2 (IKEv2) defined in [RFC7296] uses UDP as a transport for its messages. If the size of a message is larger than the PMTU, IP fragmentation takes place, which has been shown to cause operational challenge in certain network configurations and devices. The problem is described in more detail in [RFC7383], which also defines an extension to IKEv2 called IKE fragmentation. This extension allows IKE messages to be fragmented at the IKE level, eliminating possible issues caused by IP fragmentation. However, 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 are usually small enough not to cause IP fragmentation.

However, the situation has been changing recently. One example of the need to transfer large amount of data before an IKE SA is created is using Quantum Computer resistant key exchange methods in IKEv2. 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 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 specification describes a way to transfer a large amount of data in IKEv2 using UDP transport. For this purpose the document defines a new exchange for the IKEv2 protocol, called Intermediate Exchange or IKE_INTERMEDIATE. One or more these exchanges may take place right after the IKE_SA_INIT exchange and prior to the IKE_AUTH exchange. The IKE_INTERMEDIATE exchange messages can be fragmented using the 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.

The Intermediate Exchange can be used to transfer large public keys of QC-resistant key exchange methods, but its application is not limited to this use case. This exchange can also 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 IKE_INTERMEDIATE exchange without tying it to any specific use case. It is expected that separate specifications will define for which purposes and how the IKE_INTERMEDIATE exchange is used in IKEv2. Some considerations must be taken into account when designing such specifications:

- * The IKE_INTERMEDIATE exchange is not intended for bulk transfer. This document doesn't set a hard cap on the amount of data that can be safely transferred using this mechanism, as it depends on its application. But it is anticipated that in most cases the amount of data will be limited to tens of Kbytes (few hundred Kbytes in extreme cases), which is believed to cause no network problems (see [RFC6928] as an example of experiments with sending similar amounts of data in the first TCP flight). See also Section 5 for the discussion of possible DoS attack vectors when amount of data sent in IKE_INTERMEDIATE is too large.
- * It is expected that the IKE_INTERMEDIATE exchange will only be used for transferring data that is needed to establish IKE SA and not for data that can be send later when this SA is established.

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.

It is expected that readers are familiar with the terms used in the IKEv2 specification [RFC7296].

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
-----	-----
<pre>HDR, SAi1, KEi, Ni, [N(INTERMEDIATE_EXCHANGE_SUPPORTED)] --></pre>	<pre><-- HDR, SAR1, KEr, Nr, [CERTREQ], [N(INTERMEDIATE_EXCHANGE_SUPPORTED)]</pre>

The `INTERMEDIATE_EXCHANGE_SUPPORTED` is a Status Type IKEv2 notification. Its Notify Message Type is 16438, Protocol ID and SPI Size are both set to 0. This specification doesn't define any data that this notification may contain, so the Notification Data is left empty. However, future enhancements to this specification may override this. Implementations MUST ignore 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 even when support for this exchanged has been negotiated.

The Intermediate Exchange is denoted as `IKE_INTERMEDIATE`, its Exchange Type is 43.

Initiator	Responder
-----	-----
HDR, ..., SK {...} -->	<-- HDR, ..., SK {...}

The initiator may use several IKE_INTERMEDIATE exchanges if necessary. Since window size is initially set to one for both peers (Section 2.3 of [RFC7296]), these exchanges MUST be sequential 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 IKE_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 IKE_INTERMEDIATE exchange, 2 for the next (if any) and so on. Implementations MUST verify that Message IDs in the IKE_INTERMEDIATE messages they receive actually follow this rule. The Message ID for the first pair of the IKE_AUTH messages is one more than the value used in the last IKE_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 switch to port 4500 in the first IKE_INTERMEDIATE exchange and use this port for all subsequent exchanges, as described in Section 2.23 of [RFC7296].

The content of the IKE_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 IKE_INTERMEDIATE exchange is to avoid IP fragmentation when large amounts of data need to be transferred prior to IKE_AUTH, the Encrypted payload MUST be present in the IKE_INTERMEDIATE exchange messages and payloads containing large data MUST be placed inside it. This will allow IKE fragmentation [RFC7383] to take place, provided it is supported by the peers and negotiated in the initial exchange.

Appendix A contains an example of using an IKE_INTERMEDIATE exchange in creating an IKE SA.

3.3. The IKE_INTERMEDIATE Exchange Protection and Authentication

3.3.1. Protection of the IKE_INTERMEDIATE Messages

The keys SK_e[i/r] and SK_a[i/r] for the IKE_INTERMEDIATE exchanges protection are computed in the standard fashion, as defined in the Section 2.14 of [RFC7296].

Every subsequent IKE_INTERMEDIATE exchange uses the most recently calculated IKE SA keys before this exchange is started. So, the first IKE_INTERMEDIATE exchange always uses SK_e[i/r] and SK_a[i/r] keys that were computed as a result of the IKE_SA_INIT exchange. If additional key exchange is performed in the first IKE_INTERMEDIATE exchange, resulting in the update of SK_e[i/r] and SK_a[i/r], then these updated keys are used for protection of the second IKE_INTERMEDIATE exchange. Otherwise, the original SK_e[i/r] and SK_a[i/r] keys are used again, and so on.

Once all the IKE_INTERMEDIATE exchanges are completed, the most recently calculated SK_e[i/r] and SK_a[i/r] keys are used for protection of the IKE_AUTH and all the subsequent exchanges.

3.3.2. Authentication of the IKE_INTERMEDIATE Exchanges

The IKE_INTERMEDIATE messages must be authenticated in the IKE_AUTH exchange, which is performed by adding their content into the AUTH payload calculation. It is anticipated that in many use cases IKE_INTERMEDIATE messages will be fragmented using IKE fragmentation [RFC7383] mechanism. According to [RFC7383], when IKE fragmentation is negotiated, the initiator may first send a request message in unfragmented form, but later turn on IKE fragmentation and re-send it fragmented if no response is received after a few retransmissions. In addition, peers may re-send fragmented message using different fragment sizes to perform simple PMTU discovery.

The requirement to support this behavior makes authentication challenging: it is not appropriate to add on-the-wire content of the IKE_INTERMEDIATE messages into the AUTH payload calculation, because implementations are generally unaware in which form these messages are received by peers. Instead, a more complex scheme is used -- authentication is performed by adding content of these messages before their encryption and possible fragmentation, so that data to be authenticated doesn't depend on the form the messages are delivered in.

If any IKE_INTERMEDIATE exchange took place, 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 | NonceIData | MACedIDForR | IntAuth

```

```

IntAuth = IntAuth_iN | IntAuth_rN | IKE_AUTH_MID

```

```

IntAuth_i1 = prf(SK_pi1, IntAuth_i1A [| IntAuth_i1P])
IntAuth_i2 = prf(SK_pi2, IntAuth_i1 | IntAuth_i2A [| IntAuth_i2P])
IntAuth_i3 = prf(SK_pi3, IntAuth_i2 | IntAuth_i3A [| IntAuth_i3P])
...
IntAuth_iN = prf(SK_piN, IntAuth_iN-1 | IntAuth_iNA [| IntAuth_iNP])

IntAuth_r1 = prf(SK_pr1, IntAuth_r1A [| IntAuth_r1P])
IntAuth_r2 = prf(SK_pr2, IntAuth_r1 | IntAuth_r2A [| IntAuth_r2P])
IntAuth_r3 = prf(SK_pr3, IntAuth_r2 | IntAuth_r3A [| IntAuth_r3P])
...
IntAuth_rN = prf(SK_prN, IntAuth_rN-1 | IntAuth_rNA [| IntAuth_rNP])

```

The essence of this modification is that a new chunk called IntAuth is appended to the string of octets that is signed (or MAC'ed) by the peers. IntAuth consists of three parts: IntAuth_iN, IntAuth_rN, and IKE_AUTH_MID.

The IKE_AUTH_MID chunk is a value of the Message ID field from the IKE Header of the first round of the IKE_AUTH exchange. It is represented as a four octet integer in network byte order (in other words, exactly as it appears on the wire).

The IntAuth_iN and IntAuth_rN chunks each represent the cumulative result of applying the negotiated prf to all IKE_INTERMEDIATE exchange messages sent during IKE SA establishment by the initiator and the responder respectively. After the first IKE_INTERMEDIATE exchange is completed peers calculate the IntAuth_i1 value by applying the negotiated prf to the content of the request message from this exchange and calculate the IntAuth_r1 value by applying the negotiated prf to the content of the response message. For every following IKE_INTERMEDIATE exchange (if any) peers re-calculate these values as follows. After the n-th exchange is completed they compute IntAuth_[i/r]n by applying the negotiated prf to the concatenation of IntAuth_[i/r](n-1) (computed for the previous IKE_INTERMEDIATE exchange) and the content of the request (for IntAuth_in) or response (for IntAuth_rn) messages from this exchange. After all IKE_INTERMEDIATE exchanges are over the resulted IntAuth_[i/r]N values (assuming N exchanges took place) are used in the computing the AUTH payload.

For the purpose of calculating the `IntAuth_[i/r]*` values the content of the `IKE_INTERMEDIATE` messages is represented as two chunks of data: mandatory `IntAuth_[i/r]*A` optionally followed by `IntAuth_[i/r]*P`.

The `IntAuth_[i/r]*A` chunk consists of the sequence of octets from the first octet of the IKE Header (not including prepended four octets of zeros, if UDP encapsulation or TCP encapsulation of ESP packets is used) to the last octet of the generic header of the Encrypted payload. The scope of `IntAuth_[i/r]*A` is identical to the scope of Associated Data defined for use of AEAD algorithms in IKEv2 (see Section 5.1 of [RFC5282]), which is stressed by using "A" suffix in its name. Note, that calculation of `IntAuth_[i/r]*A` doesn't depend on whether an AEAD algorithm or a plain cipher is used in IKE SA.

The `IntAuth_[i/r]*P` chunk is present if the Encrypted payload is not empty. It consists of the content of the Encrypted payload that is fully formed, but not yet encrypted. The Initialization Vector, the Padding, the Pad Length and the Integrity Checksum Data fields (see Section 3.14 of [RFC7296]) are not included into the calculation. In other words, the `IntAuth_[i/r]*P` chunk is the inner payloads of the Encrypted payload in plaintext form, which is stressed by using "P" suffix in its name.

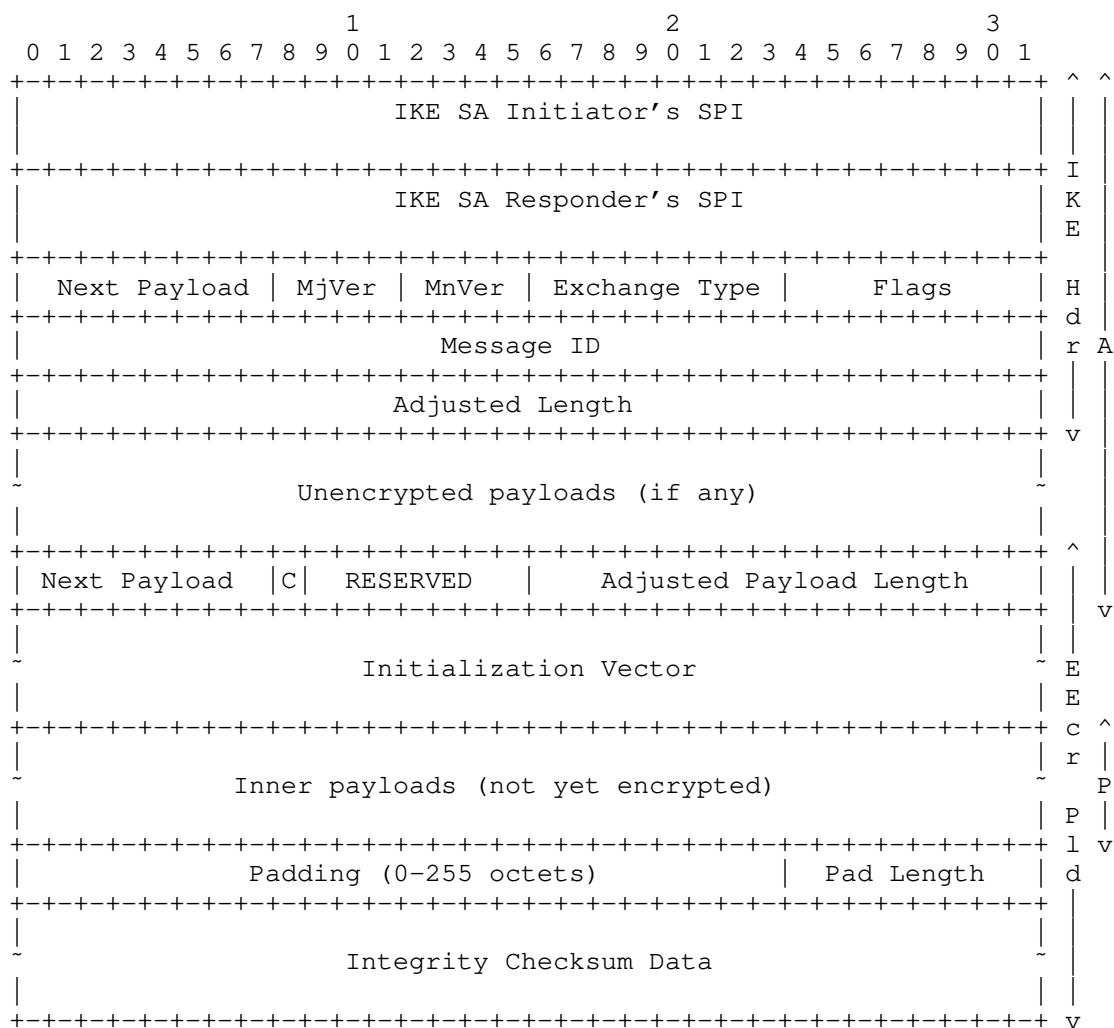


Figure 1: Data to Authenticate in the IKE_INTERMEDIATE Exchange Messages

Figure 1 illustrates the layout of the IntAuth__[i/r]*A (denoted as A) and the IntAuth__[i/r]*P (denoted as P) chunks in case the Encrypted payload is not empty.

For the purpose of prf calculation the Length field in the IKE Header and the Payload Length field in the Encrypted payload header are adjusted so that they don't count the lengths of Initialization Vector, Integrity Checksum Data, Padding and Pad Length fields. In other words, the Length field in the IKE Header (denoted as Adjusted

Length in Figure 1) is set to the sum of the lengths of `IntAuth_[i/r]*A` and `IntAuth_[i/r]*P`, and the Payload Length field in the Encrypted payload header (denoted as Adjusted Payload Length in Figure 1) is set to the length of `IntAuth_[i/r]*P` plus the size of the Encrypted payload header (four octets).

The prf calculations MUST be applied to whole messages only, before possible IKE fragmentation. This ensures that the `IntAuth` will be the same regardless of whether IKE fragmentation takes place or not. If the message was received in fragmented form, it MUST be reconstructed before calculating the prf as if it were received unfragmented. While reconstructing, the RESERVED field in the reconstructed Encrypted payload header MUST be set to the value of the RESERVED field in the Encrypted Fragment payload header from the first fragment (with Fragment Number field set to 1).

Note that it is possible to avoid actual reconstruction of the message by incrementally calculating prf on decrypted (or ready to be encrypted) fragments. However, care must be taken to properly replace the content of the Next Header and the Length fields so that the result of computing the prf is the same as if it were computed on the reconstructed message.

Each calculation of `IntAuth_[i/r]*` uses its own keys `SK_p[i/r]*`, which are the most recently updated `SK_p[i/r]` keys available before the corresponded `IKE_INTERMEDIATE` exchange is started. The first `IKE_INTERMEDIATE` exchange always uses the `SK_p[i/r]` keys that were computed in the `IKE_SA_INIT` as `SK_p[i/r]1`. If the first `IKE_INTERMEDIATE` exchange performs additional key exchange resulting in `SK_p[i/r]` update, then this updated `SK_p[i/r]` are used as `SK_p[i/r]2`, otherwise the original `SK_p[i/r]` are used, and so on. Note that if keys are updated, then for any given `IKE_INTERMEDIATE` exchange the keys `SK_e[i/r]` and `SK_a[i/r]` used for protection of its messages (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 `IKE_INTERMEDIATE` Exchange

Since messages of the `IKE_INTERMEDIATE` exchange are not authenticated until the `IKE_AUTH` exchange successfully completes, possible errors need to be handled with care. There is a trade-off between providing better diagnostics of the problem and risk of becoming part of DoS attack. Section 2.21.1 and 2.21.2 of [RFC7296] describe how errors are handled in initial IKEv2 exchanges; these considerations are also applied to the `IKE_INTERMEDIATE` exchange with a qualification, that not all error notifications may appear in the `IKE_INTERMEDIATE` exchange (for example, errors concerning authentication are generally only applicable to the `IKE_AUTH` exchange).

4. Interaction with other IKEv2 Extensions

The IKE_INTERMEDIATE exchanges MAY be used during the IKEv2 Session Resumption [RFC5723] between the IKE_SESSION_RESUME and the IKE_AUTH exchanges. To be able to use it peers MUST negotiate support for intermediate exchange by including INTERMEDIATE_EXCHANGE_SUPPORTED notifications in the IKE_SESSION_RESUME messages. Note, that a flag whether peers supported the IKE_INTERMEDIATE exchange is not stored in the resumption ticket and is determined each time from the IKE_SESSION_RESUME exchange.

5. Security Considerations

The data that is transferred by means of the IKE_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 receive messages from the party they performed the IKE_SA_INIT with if they can successfully verify the Integrity Checksum Data of the Encrypted payload.

The main application for the Intermediate Exchange is to transfer large amounts of data before an 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 IKE_INTERMEDIATE exchanges. Section 5 of [RFC7383] contains security considerations for IKE fragmentation.

Since authentication of the peers occurs only in the IKE_AUTH exchange, malicious initiator may use the Intermediate Exchange to mount Denial of Service attack on responder. In this case it starts creating IKE SA, negotiates using the Intermediate Exchanges and transfers a lot of data to the responder that may also require some computationally expensive processing. Then it aborts the SA establishment before the IKE_AUTH exchange. Specifications utilizing the Intermediate Exchange MUST NOT allow unlimited number of these exchanges to take place on initiator's discretion. It is recommended that these specifications are defined in such a way, that the responder would know (possibly via negotiation with the initiator) the exact number of these exchanges that need to take place. In other words: it is preferred that both the initiator and the responder know after the IKE_SA_INIT is completed the exact number of the IKE_INTERMEDIATE exchanges they have to perform; it is allowed that some IKE_INTERMEDIATE exchanges are optional and are performed on the initiator's discretion, but in this case the maximum number of optional exchanges must be hard capped by the corresponding specification. In addition, [RFC8019] provides guidelines for the responder of how to deal with DoS attacks during IKE SA establishment.

Note that if an attacker was able to break the key exchange in real time (e.g. by means of a Quantum Computer), then the security of the IKE_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 the employed authentication mechanism), but the ability to inject forged IKE_INTERMEDIATE exchange messages with valid ICV would allow the attacker to mount a Denial-of-Service attack. Moreover, if in this situation the negotiated prf was not secure against second preimage attack with known key, then the attacker could forge the IKE_INTERMEDIATE exchange messages without later being detected in the IKE_AUTH exchange. To do this the attacker would 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:

43	IKE_INTERMEDIATE
----	------------------

This document also defines a new Notify Message Type in the "IKEv2 Notify Message Types - Status Types" registry:

16438	INTERMEDIATE_EXCHANGE_SUPPORTED
-------	---------------------------------

7. Implementation Status

[Note to RFC Editor: please, remove this section before publishing RFC.]

At the time of writing the -05 version of the draft there were at least three independent interoperable implementations of this specification from the following vendors:

- * ELVIS-PLUS
- * strongSwan
- * libreswan (only one IKE_INTERMEDIATE exchange is supported)

8. Acknowledgements

The idea to use an intermediate exchange between IKE_SA_INIT and IKE_AUTH was first suggested by Tero Kivinen. He also helped with writing an example of using IKE_INTERMEDIATE exchange (shown in Appendix A). Scott Fluhrer and Daniel Van Geest identified a possible problem with authentication of the IKE_INTERMEDIATE exchange and helped to resolve it. Author is grateful to Tobias Brunner who raised good questions concerning authentication of the IKE_INTERMEDIATE exchange and proposed how to make the size of authentication chunk constant regardless of the number of exchanges. Author is also grateful to Paul Wouters and to Benjamin Kaduk who suggested a lot of text improvements for the document.

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>>.
- [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>>.

9.2. Informative References

- [RFC5282] Black, D. and D. McGrew, "Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol", RFC 5282, DOI 10.17487/RFC5282, August 2008, <<https://www.rfc-editor.org/info/rfc5282>>.
- [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>>.
- [RFC6928] Chu, J., Dukkkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <<https://www.rfc-editor.org/info/rfc6928>>.
- [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>>.
- [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>>.

Appendix A. Example of IKE_INTERMEDIATE exchange

This appendix contains an example of the messages using IKE_INTERMEDIATE exchanges. This appendix is purely informative; if it disagrees with the body of this document, the other text is considered correct.

In this example there is one IKE_SA_INIT exchange and two IKE_INTERMEDIATE exchanges, followed by the IKE_AUTH exchange to authenticate all initial exchanges. The xxx in the HDR(XXX,MID=yyy) indicates the exchange type, and yyy tells the message id used for that exchange. The keys used for each SK {} payload are indicated in the parenthesis after the SK. Otherwise, the payload notation is the same as is used in [RFC7296].

Initiator	Responder
-----	-----
HDR(IKE_SA_INIT,MID=0), SAi1, KEi, Ni, N(INTERMEDIATE_EXCHANGE_SUPPORTED) -->	<-- HDR(IKE_SA_INIT,MID=0), SAr1, KEr, Nr, [CERTREQ], N(INTERMEDIATE_EXCHANGE_SUPPORTED)

At this point peers calculate SK_* and store them as SK_*1. SK_e[i/r]1 and SK_a[i/r]1 will be used to protect the first IKE_INTERMEDIATE exchange and SK_p[i/r]1 will be used for its authentication.

Initiator	Responder
-----	-----
HDR(IKE_INTERMEDIATE,MID=1), SK(SK_ei1,SK_ai1) {...} -->	<-- HDR(IKE_INTERMEDIATE,MID=1), SK(SK_er1,SK_ar1) {...}
<Calculate IntAuth_i1 = prf(SK_pi1, ...)>	
	<Calculate IntAuth_r1 = prf(SK_pr1, ...)>

If after completing this IKE_INTERMEDIATE exchange the SK_*1 keys are updated (e.g., as a result of a new key exchange), then the peers store the updated keys as SK_*2, otherwise they use SK_*1 as SK_*2. SK_e[i/r]2 and SK_a[i/r]2 will be used to protect the second IKE_INTERMEDIATE exchange and SK_p[i/r]2 will be used for its authentication.

Initiator	Responder
-----	-----
HDR(IKE_INTERMEDIATE,MID=2), SK(SK_ei2,SK_ai2) {...} -->	<-- HDR(IKE_INTERMEDIATE,MID=2), SK(SK_er2,SK_ar2) {...}
<Calculate IntAuth_i2 = prf(SK_pi2, ...)>	
	<Calculate IntAuth_r2 = prf(SK_pr2, ...)>

If after completing the second IKE_INTERMEDIATE exchange the SK_*2 keys are updated (e.g., as a result of a new key exchange), then the peers store the updated keys as SK_*3, otherwise they use SK_*2 as

SK_*3. SK_e[i/r]3 and SK_a[i/r]3 will be used to protect the IKE_AUTH exchange, SK_p[i/r]3 will be used for authentication, and SK_d3 will be used for derivation of other keys (e.g. for Child SAs).

Initiator -----	Responder -----
HDR(IKE_AUTH,MID=3),	
SK(SK_ei3,SK_ai3)	
{IDi, [CERT,] [CERTREQ,]	
{IDr,] AUTH, SAi2, TSi, TSr} -->	
	<-- HDR(IKE_AUTH,MID=3),
	SK(SK_er3,SK_ar3)
	{IDr, [CERT,] AUTH, SAr2, TSi, TSr}

In this example two IKE_INTERMEDIATE exchanges took place, therefore SK_*3 keys would be used as SK_* keys for further cryptographic operations in the context of the created IKE SA, as defined in [RFC7296].

Author's Address

Valery Smyslov
 ELVIS-PLUS
 PO Box 81
 Moscow (Zelenograd)
 124460
 Russian Federation
 Phone: +7 495 276 0211
 Email: svan@elvis.ru

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.

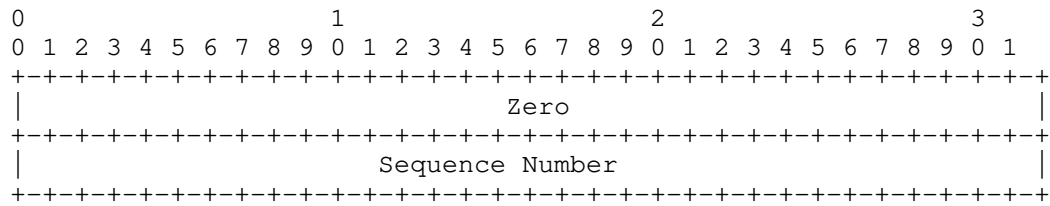


Figure 1: Implicit IV with a 4 byte Sequence Number

- 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.

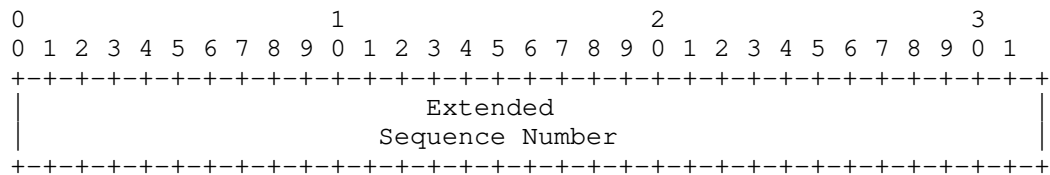


Figure 2: Implicit IV with an 8-byte Extended Sequence Number

- 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

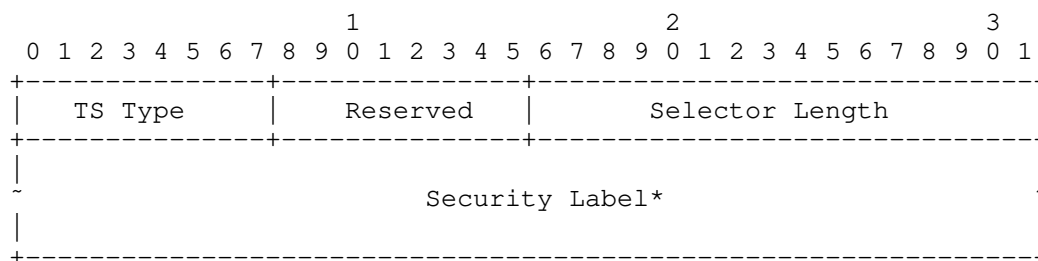


Figure 1: Labeled IPsec Traffic Selector

*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 or 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:


```

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:

```
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 )
```

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 2^n 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

IPSECME Working Group
Internet-Draft
Intended status: Standards Track
Expires: 14 August 2022

S. Kompoti
W. Pan
Huawei
P. Wouters
Aiven
M. Bharath
Mavenir
M. Chen
CMCC
M. Richardson, Ed.
Sandelman Software Works
10 February 2022

IKEv2 Optional SA&TS Payloads in Child Exchange
draft-kompoti-ipsecme-ikev2-sa-ts-payloads-opt-08

Abstract

This document describes a method for reducing the size of the Internet Key Exchange version 2 (IKEv2) CREATE_CHILD_SA exchanges used for rekeying of the IKE or Child SA by replacing the SA and TS payloads with a Notify Message payload. Reducing size and complexity of IKEv2 exchanges is especially useful for low power consumption battery powered devices.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-kompoti-ipsecme-ikev2-sa-ts-payloads-opt/>.

Discussion of this document takes place on the ipsec Working Group mailing list (<mailto:ipsec@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/ipsec/>.

Source for this draft and an issue tracker can be found at <https://github.com/mcr/ipsecme-ikev2-sa-ts-payloads.git>.

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 August 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	3
2. Conventions Used in This Document	4
2.1. Requirements Language	4
3. Negotiation of Support for OPTIMIZED REKEY	4
4. Optimized Rekey of the IKE SA	5
5. Optimized Rekey of Child SAs	5
6. Payload Formats	6
6.1. OPTIMIZED_REKEY_SUPPORTED Notify	6
6.2. OPTIMIZED_REKEY Notify	7
7. IANA Considerations	7
8. Operational Considerations	8
9. Security Considerations	8
10. Acknowledgments	8
11. Normative References	8
Authors' Addresses	8

1. Introduction

The Internet Key Exchange protocol version 2 (IKEv2) [RFC7296] is used to negotiate Security Association (SA) parameters for the IKE SA and the Child SAs. Cryptographic key material for these SAs have a limited lifetime before it needs to be refreshed, a process referred to as "rekeying". IKEv2 uses the CREATE_CHILD_SA exchange to rekey either the IKE SA or the Child SAs.

When rekeying, a full set of negotiation parameters are exchanged. However, most of these parameters will be the same as before, and some of these parameters MUST not change.

For example, the Traffic Selector (TS) negotiated for the new Child SA MUST cover the Traffic Selectors negotiated for the old Child SA. And in practically all cases, a new Child SA does not need to cover a wider set of Traffic. In the rare case where this would be needed, either a standard rekey could be used or a new Child SA could be negotiated followed by a deletion of the replaced Child SA.

Similarly, IKEv2 states that the cryptographic parameters negotiated for rekeying SHOULD NOT be different. This means that the security properties of the IKE or Child SA in practise do not change during a typical rekey.

This document specifies a method to omit these parameters and replace them with a single Notify Message declaring that all these parameters are identical to the originally negotiated parameters.

Large scale IKEv2 gateways such as Evolved Packet Data Gateway (ePDG) in 4G networks or Centralized Radio Access Network (cRAN/Cloud) gateways in 5G networks typically support more than 100,000 IKE/IPsec connections. At any point in time, there will be hundreds or thousands of IKE SAs and Child SAs that are being rekeyed. This takes a large amount of bandwidth and CPU power and any protocol simplification or bandwidth reducing would result in a significant resource saving.

For Internet of Things (IoT) devices which utilize low power consumption technology, reducing the size of the CREATE_CHILD_SA exchange for rekeying reduces its power consumption, as sending bytes over the air is usually the most power consuming operation of such a device. Reducing the CPU operations required to verify the rekey exchanges parameters will also save power and extend the lifetime for these devices.

When using identical parameters for the IKE SA or Child SA rekey, the SA and TS payloads can be omitted thanks to the optimization defined in this document. For an IKE SA rekey, instead of the (large) SA payload, only a Key Exchange (KE) payload and a new Notify Type payload with the new SPI are required. For a Child SA payload, instead of the SA or TS payloads, only an optional nonce payload (when using PFS) and a new Notify Type payload with the new SPI are needed. This makes the rekey exchange packets much smaller and the peers do not need to verify that the SA or TS parameters are compatible with the old SA parameters.

2. Conventions Used in This Document

2.1. 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.

3. Negotiation of Support for OPTIMIZED REKEY

To indicate support for the optimized rekey negotiation, the initiator includes the OPTIMIZED_REKEY_SUPPORTED notify payload in the IKE_AUTH exchange request. During this initial key request, the entire SA and TA payloads are included as normal. A responder that supports the optimized rekey exchange includes the OPTIMIZED_REKEY_SUPPORTED notify payload in its response. Note that the notify indicates support for optimized rekey for both IKE and Child SAs.

A responder that does not support the optimized rekey exchange processes the SA and TA payloads as normal, and does not include the new Notify. As per regular IKEv2 processing, a responder that does not recognize this new Notify, MUST ignore the notify. Responders may have been administratively configured with the optimization turned off for local reasons. The absence of the Notify indicates to the initiator that the optimization is not available, and normal, full rekey should be done.

When a peer wishes to rekey an IKE SA or Child SA, it MAY use the optimized rekey method during the CREATE_CHILD_SA exchange. If both peers have exchanged OPTIMIZED_REKEY_SUPPORTED notifies, peers SHOULD use the optimized rekey method for rekeys. Non-optimized, regular rekey requests MUST always be accepted.

The IKE_AUTH message exchange in this case is shown below:

Initiator	Responder

HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr, N(OPTIMIZED_REKEY_SUPPORTED)} -->	<-- HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr, N(OPTIMIZED_REKEY_SUPPORTED)}

4. Optimized Rekey of the IKE SA

The initiator of an optimized rekey request sends a CREATE_CHILD_SA payload with the OPTIMIZED_REKEY notify payload containing the new Security Parameter Index (SPI) for the new IKE SA. It omits the SA payload.

The responder of an optimized rekey request replies with an included OPTIMIZED_REKEY notify with its new IKE SPI and also omits the SA payload.

Both parties send their nonce and KE payloads just as they would do for a regular IKE SA rekey.

Using the old SPI from the IKE header and the two new SPIs respectively from the initiator and responder's OPTIMIZED_REKEY payloads, both parties can perform the IKE SA rekey operation.

The CREATE_CHILD_SA message exchange in this case is shown below:

Initiator	Responder

HDR, SK {N(OPTIMIZED_REKEY,newSPIi), Ni, KEi} -->	<-- HDR, SK {N(OPTIMIZED_REKEY,newSPIr), Nr, KEr}

5. Optimized Rekey of Child SAs

The initiator of an optimized rekey request sends a CREATE_CHILD_SA payload with the OPTIMIZED_REKEY notify payload containing the new Security Parameter Index (SPI) for the new Child SA. It omits the SA and TS payloads. If the current Child SA was negotiated with Perfect Forward Secrecy (PFS), a KEi payload MUST be included as well. If no PFS was negotiated for the current Child SA, a KEi payload MUST NOT be included.

The responder of an optimized rekey request performs the same process. It includes the OPTIMIZED_REKEY notify with its new IKE SPI and omits the SA and TS payloads. Depending on the PFS negotiation of the current Child SA, the responder includes a KEr payload.

Both parties send their nonce payloads just as they would do for a regular Child SA rekey.

Using the old SPI from the REKEY_SA payload and the two new SPIs respectively from the initiator and responder's OPTIMIZED_REKEY payloads, both parties can perform the Child SA rekey operation.

The CREATE_CHILD_SA message exchange in this case is shown below:

Initiator	Responder

HDR, SK {N(REKEY_SA,oldSPI), N(OPTIMIZED_REKEY,newSPIi), Ni, [KEi,]} -->	<-- HDR, SK {N(OPTIMIZED_REKEY,newSPIr), Nr, [KEr,]}

6. Payload Formats

6.1. OPTIMIZED_REKEY_SUPPORTED Notify

The OPTIMIZED_REKEY_SUPPORTED Notify Message type notification is used by the initiator and responder to indicate their support for the optimized rekey negotiation.

										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	RESERVED										Payload Length																		
Protocol ID(=0)										SPI Size (=0)										Notify Message Type																			

* Protocol ID (1 octet) - MUST be 0.

* SPI Size (1 octet) - MUST be 0, meaning no SPI is present.

* Notify Message Type (2 octets) - MUST be set to the value TBD1.

This Notify Message type contains no data.

The Critical bit MUST be 0. A non-zero value MUST be ignored.

6.2. OPTIMIZED_REKEY Notify

The OPTIMIZED_REKEY Notify Message type is used to perform an optimized IKE SA or Child SA rekey.

0									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	RESERVED								Payload Length																		
+-----+-----+-----+-----+																																				
Protocol ID										SPI Size (=8)								Notify Message Type																		
+-----+-----+-----+-----+																																				
									Security Parameter Index (SPI)																											
+-----+-----+-----+-----+																																				

- * Protocol ID (1 octet) - For an IKE SA rekey, this field MUST contain (1). For Child SAs, this field MUST contain either (2) to indicate AH or (3) to indicate ESP.
- * SPI Size (1 octet) - MUST be 8 when rekeying an IKE SA. MUST be 4 when rekeying a Child SA.
- * Notify Message Type (2 octets) - MUST be set to the value TBD2.
- * SPI (4 octets or 8 octets) - Security Parameter Index. The new SPI.

The Critical bit MUST be 1. A value of 0 MUST be ignored.

7. IANA Considerations

This document defines two new Notify Message Types in the "IKEv2 Notify Message Types - Status Types" registry. IANA is requested to assign codepoints in this registry.

NOTIFY messages: status types	Value
OPTIMIZED_REKEY_SUPPORTED	TBD1
OPTIMIZED_REKEY	TBD2

8. Operational Considerations

Some implementations allow sending rekey messages with a different set of Traffic Selectors or cryptographic parameters in response to a configuration update. IKEv2 states this SHOULD NOT be done. Whether or not optimized rekeying is used, a configuration change that changes the Traffic Selectors or cryptographic parameters MUST NOT use the optimized rekey method. It SHOULD also not use a regular rekey method but instead start an entire new IKE and Child SA negotiation with the new parameters.

9. Security Considerations

The optimized rekey removes sending unnecessary new parameters that originally would have to be validated against the original parameters. In that sense, this optimization enhances the security of the rekey process by reducing the complexity and code required.

10. Acknowledgments

Special thanks to Valery Smyslov and Antony Antony.

11. 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/rfc/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/rfc/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/rfc/rfc8174>>.

Authors' Addresses

Sandeep Kampati
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore 560066
Karnataka
India

Email: sandeepkampati@huawei.com

Wei Pan
Huawei Technologies
101 Software Avenue, Yuhuatai District
Nanjing
Jiangsu,
China

Email: william.panwei@huawei.com

Paul Wouters
Aiven

Email: paul.wouters@aiven.io

Meduri S S Bharath
Mavenir Systems Pvt Ltd
Manyata Tech Park
Bangalore
Karnataka
India

Email: bharath.meduri@mavenir.com

Meiling Chen
China Mobile
32 Xuanwumen West Street, West District
Beijing
100053
China

Email: chenmeiling@chinamobile.com

Michael Richardson (editor)
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Internet Engineering Task Force
Internet-Draft
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: January 10, 2020

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
July 9, 2019

Framework to Integrate Post-quantum Key Exchanges into Internet Key
Exchange Protocol Version 2 (IKEv2)
draft-tjhai-ipsecme-hybrid-qske-ikev2-04

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 January 10, 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. Introduction	2
1.1. Problem Description	2
1.2. Proposed Extension	3
1.3. Changes	4
1.4. Document Organization	5
2. Design Criteria	5
3. The Framework of Hybrid Post-Quantum Key Exchange	7
3.1. Overall design	7
3.2. Overall Protocol	8
3.2.1. IKE_SA_INIT Round: Negotiation	9
3.2.2. IKE_INTERMEDIATE Round: Additional Key Exchanges	10
3.2.3. IKE_AUTH Exchange	11
3.2.4. CREATE_CHILD_SA Exchange	11
4. IANA Considerations	14
5. Security Considerations	14
6. Acknowledgements	16
7. References	16
7.1. Normative References	16
7.2. Informative References	16
Appendix A. Alternative Design	17
Authors' Addresses	21

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 maximum transmission unit (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 IKE_INTERMEDIATE exchange as outlined in [I-D.ietf-ipsecme-ikev2-intermediate]. 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 IKE_INTERMEDIATE exchanges, each of which includes a secondary key exchange. As the IKE_INTERMEDIATE exchange is encrypted, the IKE fragmentation protocol RFC7383 can be used.

The IKE SK_* values are updated after each exchange, and so the final IKE SK_* values 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].

Note also, that the proposed approach of performing multiple successive key exchanges in such a way that resulting session keys depend on all of them is not limited to achieving quantum resistance only. It can also be used when all the performed key exchanges are classical (EC)DH ones, but for some reasons (e.g. policy requirements) it is essential to perform multiple of them.

1.3. Changes

RFC EDITOR PLEASE DELETE THIS SECTION.

Changes in this draft in each version iterations.

draft-tjhai-ipsecme-hybrid-qske-ikev2-04

- o Clarification about key derivation in case of multiple key exchanges in CREATE_CHILD_SA is added.
- o Resolving rekey collisions in case of multiple key exchanges is clarified.

draft-tjhai-ipsecme-hybrid-qske-ikev2-03

- o Using multiple key exchanges CREATE_CHILD_SA is defined.

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 IKE_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 Appendix A. Section 4 discusses IANA considerations for the namespaces introduced in this document, and lastly Section 5 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 Transform Type 4 identifiers 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. To make this more clear for implementers this document renames Transform Type 4 from "Diffie-Hellman Group Transform IDs" to "Key Exchange Method Transform IDs".

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 between the IKE_SA_INIT and the IKE_AUTH exchanges by utilizing IKE_INTERMEDIATE exchange described in [I-D.ietf-ipsecme-ikev2-intermediate].

In order to minimize communication overhead, only the key shares that are agreed to be used are actually exchanged. In order to achieve this several new Transform Types are defined, each sharing possible Transform IDs with Transform Type 4. The IKE_SA_INIT message includes one or more newly defined SA transforms that lists the extra key exchange policy required by the initiator; the responder selects single transform of each type, and returns them back in the response IKE_SA_INIT message. Then, provided that additional key exchanges are negotiated the initiator and the responder perform one or more IKE_INTERMEDIATE exchanges; each such exchange includes a KE payload for one of the negotiated key exchanges.

Here is an overview of the initial exchanges:

Initiator

Responder

```
----->
<-- IKE_SA_INIT (additional key exchanges negotiation) -->
```

```
<-- {IKE_INTERMEDIATE (additional key exchange)} -->
```

```
...
```

```
<-- {IKE_INTERMEDIATE (additional 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 IKE_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 a 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 IKE_INTERMEDIATE exchanges. The possible transform IDs for these transform types are the same as IDs for the Transform Type 4, 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 a series of IKE_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 IKE_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 IDs IANA registry. Instead they all share a single registry for transform IDs - "Key Exchange Method 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 Exchange transforms 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. IKE_INTERMEDIATE Round: Additional Key Exchanges

For each extra key exchange agreed to in the IKE_SA_INIT exchange, the initiator and the responder perform one or more IKE_INTERMEDIATE exchanges, as described in [I-D.ietf-ipsecme-ikev2-intermediate].

These exchanges are as follows:

Initiator		Responder
HDR, SK {Ni(n), KEi(n)}	-->	
	<--	HDR, SK {Nr(n), KER(n)}

The initiator sends a nonce in the Ni(n) payload, and the key exchange payload in the KEi(n). This packet is encrypted with the current IKE SK_* keys.

On receiving this, the responder sends a nonce in the Nr(n) payload, and the key exchange payload KER(n); again, this packet is encrypted with the current IKE SA keys.

The Diffie-Hellman Group Num field in the KEi(n) and KER(n) payloads MUST match the n-th negotiated extra key exchange. Note that the negotiated transform types (the encryption type, hash type, prf type) are not modified.

Once this exchange is done, then both sides compute an updated keying material:

$$\text{SKEYSEED}(n) = \text{prf}(\text{SK}_d(n-1), \text{KE}(n) \mid \text{Ni}(n) \mid \text{Nr}(n))$$

where KE(n) is the resulting shared secret of this key exchange and SK_d(n-1) is the last generated SK_d, (derived from the previous IKE_INTERMEDIATE exchange, or the IKE_SA_INIT if there haven't already been any IKE_INTERMEDIATE exchanges). Then, SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, SK_pr are updated as:

$$\{SK_d(n) \mid SK_ai(n) \mid SK_ar(n) \mid SK_ei(n) \mid SK_er(n) \mid SK_pi(n) \mid SK_pr(n)\} = \text{prf+} (SKEYSEED(n), Ni(n) \mid Nr(n) \mid SPIi \mid SPIr)$$

Both the initiator and the responder use this updated key values in the next exchange.

3.2.3. IKE_AUTH Exchange

After all IKE_INTERMEDIATE exchanges have completed, the initiator and the responder 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.ietf-ipsecme-ikev2-intermediate].

Note, that despite the fact, that a fresh pair of nonces is exchanged in each IKE_INTERMEDIATE exchange, only nonces from the IKE_SA_INIT are included in calculation of AUTH payload (see Section 2.15 of [RFC7296]).

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 rekey, 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. These key exchanges are performed 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 corresponding 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}

It is possible that due to some unexpected events (e.g. reboot) the Initiator could forget that he/she is in the process of performing additional key exchanges and never starts next INFORMATIONAL exchanges. The Responder MUST handle this situation gracefully and delete the associated state if he/she doesn't receive the next expected INFORMATIONAL request after some reasonable period of time.

If Responder receives INFORMATIONAL request containing ADDITIONAL_KEY_EXCHANGE notification and the content of this notify doesn't correspond to any active key exchange state the Responder has, he/she MUST send back a new error type notification STATE_NOT_FOUND. This is a non-fatal notification, its Notify Message Type is <TBA by IANA>, Protocol ID and SPI Size are both set to 0 and the data is empty. If Initiator receives this notification

in response to INFORMATIONAL exchange performing additional key exchange, he/she MUST cancel this exchange and MUST treat the whole series of exchanges started from the CREATE_CHILD_SA exchange as failed. In most cases, the receipt of this notification is caused by premature deletion of the corresponding state on the Responder (the time period between INFORMATIONAL exchanges appeared too long from Responder's point of view, e.g. due to a temporary network failure). After receiving this notification the Initiator MAY start a new CREATE_CHILD_SA exchange (eventually followed by the INFORMATIONAL exchanges) to retry the failed attempt. If the Initiator continues to receive STATE_NOT_FOUND notifications after several retries, he/she MUST treat it as fatal error and delete IKE SA (sending DELETE payload).

When rekeying IKE SA or Child SA it is possible that the peers start doing this at the same time, which is called simultaneous rekeying. Sections 2.8.1 and 2.8.2 of [RFC7296] describes how IKEv2 handles this situation. In a nutshell IKEv2 follows the rule that if in case of simultaneous rekeying two identical new IKE SAs (or two pairs of Child SAs) are created, then one of them should be deleted. Which one is to be deleted is determined by comparing the values of four nonces, that were used in the colliding CREATE_CHILD_SA exchanges - the IKE SA (or pair of Child SAs) that was created by the exchange in which the smallest nonce was used should be deleted by the initiator of this exchange.

With multiple key exchanges the SAs are not yet created once the CREATE_CHILD_SA is completed, they would be created only after the series of INFORMATIONAL exchanges is finished. For this reason if additional key exchanges were negotiated in the CREATE_CHILD_SA initiated by the losing side, there is nothing to delete and this side just stops the rekeying process - he/she MUST not initiate INFORMATIONAL exchange with next key exchange.

In most cases rekey collisions are resolved in the CREATE_CHILD_SA exchange. However, a situation may occur when due to packet loss one of the peers receives CREATE_CHILD_SA message requesting rekeying SA that is already being rekeyed by this peer (i.e. the CREATE_CHILD_SA exchange initiated by this peer has been already completed and the series of INFORMATIONAL exchanges is in progress). In this case TEMPORARY_FAILURE notification MUST be sent in response to such request.

If multiple key exchanges were negotiated in the CREATE_CHILD_SA exchange, then the resulting keys are computed as follows. In case of IKE SA rekey:

$$\text{SKEYSEED} = \text{prf}(\text{SK_d}, \text{KE} \mid \text{Ni} \mid \text{Nr} \mid \begin{array}{c} \text{KE}(1) \\ \text{KE}(n) \end{array} \mid \begin{array}{c} \text{Ni}(1) \\ \text{Ni}(n) \end{array} \mid \begin{array}{c} \text{Nr}(1) \\ \text{Nr}(n) \end{array} \dots)$$

In case of Child SA creating or rekey:

$$\text{KEYMAT} = \text{prf+}(\text{SK_d}, \text{KE} \mid \text{Ni} \mid \text{Nr} \mid \begin{array}{c} \text{KE}(1) \\ \text{KE}(n) \end{array} \mid \begin{array}{c} \text{Ni}(1) \\ \text{Ni}(n) \end{array} \mid \begin{array}{c} \text{Nr}(1) \\ \text{Nr}(n) \end{array} \dots)$$

In both cases SK_d is from existing IKE SA; KE, Ni, Nr - shared key and nonces from the CREATE_CHILD_SA; KE(1)..KE(n), Ni(1)..Ni(n), Nr(1)..Nr(n) - shared keys and nonces from additional key exchanges.

4. IANA Considerations

This document renames "Transform Type 4 - Diffie-Hellman Group Transform IDs" to "Transform Type 4 - Key Exchange Method Transform IDs"

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 Type in the "Notify Message Types - Status Types" registry:

<TBA> ADDITIONAL_KEY_EXCHANGE

and a new Notify Message Type in the "Notify Message Types - Error Types" registry:

<TBA> STATE_NOT_FOUND

5. 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, K_{Ex}, using an algorithm defined in Transform Type 2. While a quantum attacker may learn the value of K_{Ex}', if this value is obtained by means of a classical key exchange, other K_{Ex} 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.

6. 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. The authors are also grateful to Tobias Heider and Tobias Guggemos for valuable comments.

7. References

7.1. Normative References

- [I-D.ietf-ipsecme-ikev2-intermediate]
Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-ietf-ipsecme-ikev2-intermediate-00 (work in progress), June 2019.
- [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-ikev2-08 (work in progress), March 2019.

- [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>>.

Appendix A. 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_KEY_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_KEY_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

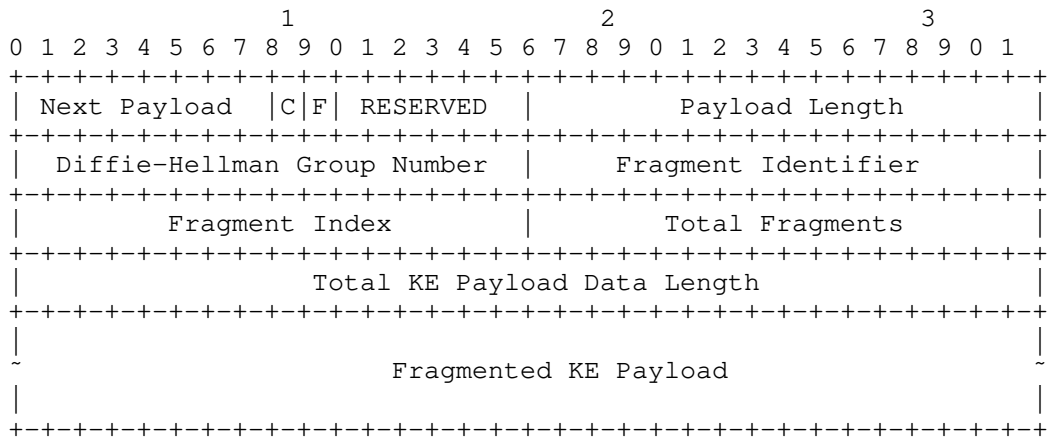
reasoning as above, this may not be a good idea from a backward compatibility perspective. Using a single KE payload would require an encoding or formatting to be defined so that both peers are able to compose and extract the individual public values. However, we believe that it is cleaner to send the 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.



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 IKE_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.

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

Network Working Group
Internet-Draft
Obsoletes: 6407 (if approved)
Intended status: Standards Track
Expires: January 9, 2020

B. Weis
Independent
V. Smyslov
ELVIS-PLUS
July 8, 2019

Group Key Management using IKEv2
draft-yeung-g-ikev2-16

Abstract

This document presents a set of IKEv2 exchanges that comprise a group key management protocol. The protocol is in conformance with the Multicast Security (MSEC) key management architecture, which contains two components: member registration and group rekeying. Both components require a Group Controller/Key Server to download IPsec group security associations to authorized members of a group. The group members then exchange IP multicast or other group traffic as IPsec packets. This document obsoletes RFC 6407.

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 January 9, 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. Introduction and Overview	3
1.1. Requirements Language	5
1.2. G-IKEv2 Integration into IKEv2 Protocol	5
1.2.1. G-IKEv2 Transport and Port	5
1.2.2. IKEv2 Header Initialization	6
1.3. G-IKEv2 Protocol	6
1.3.1. G-IKEv2 Payloads	6
1.4. G-IKEv2 Member Registration and Secure Channel Establishment	7
1.4.1. GSA_AUTH exchange	7
1.4.2. GSA_REGISTRATION Exchange	9
1.4.3. GM Registration Operations	10
1.4.4. GCKS Registration Operations	11
1.4.5. Group Maintenance Channel	12
1.4.6. Counter-based modes of operation	19
1.5. Interaction with IKEv2 Protocol Extensions	21
1.5.1. Postquantum Preshared Keys for IKEv2	21
2. Header and Payload Formats	23
2.1. The G-IKEv2 Header	23
2.2. Group Identification (IDg) Payload	24
2.3. Security Association - GM Supported Transforms (SAg)	24
2.4. Group Security Association Payload	24
2.4.1. GSA Policy	24
2.4.2. KEK Policy	26
2.4.3. GSA TEK Policy	29
2.4.4. GSA Group Associated Policy	33
2.5. Key Download Payload	34
2.5.1. TEK Download Type	36
2.5.2. KEK Download Type	37
2.5.3. LKH Download Type	38
2.5.4. SID Download Type	40
2.6. Delete Payload	42
2.7. Notify Payload	42
2.8. Authentication Payload	43
3. Security Considerations	43
3.1. GSA Registration and Secure Channel	43
3.2. GSA Maintenance Channel	44
3.2.1. Authentication/Authorization	44
3.2.2. Confidentiality	44
3.2.3. Man-in-the-Middle Attack Protection	44
3.2.4. Replay/Reflection Attack Protection	44

4.	IANA Considerations	44
4.1.	New Registries	44
4.2.	New Payload and Exchange Types Added to the Existing IKEv2 Registry	45
4.3.	Changes to Previous Allocations	45
5.	Acknowledgements	45
6.	Contributors	46
7.	References	46
7.1.	Normative References	47
7.2.	Informative References	48
Appendix A.	Use of LKH in G-IKEv2	50
A.1.	Group Creation	50
A.2.	Group Member Exclusion	51
Authors' Addresses	52

1. Introduction and Overview

A group key management protocol provides IPsec keys and policy to a set of IPsec devices which are authorized to communicate using a Group Security Association (GSA) defined in [RFC3740]. The data communications within the group (e.g., IP multicast packets) are protected by a key pushed to the group members (GMs) by the Group Controller/Key Server (GCKS). This document presents a set of IKEv2 [RFC7296] exchanges that comprise a group key management protocol.

A GM begins a "registration" exchange when it first joins the group. With G-IKEv2, the GCKS authenticates and authorizes GMs, then pushes policy and keys used by the group to the GM. G-IKEv2 includes two "registration" exchanges. The first is the GSA_AUTH exchange (Section 1.4.1), which follows an IKE_SA_INIT exchange. The second is the GSA_REGISTRATION exchange (Section 1.4.2), which a GM can use within an established IKE SA. Group rekeys are accomplished using either the GSA_REKEY exchange (a single message distributed to all GMs, usually as a multicast message), or as a GSA_INBAND_REKEY exchange delivered individually to group members using existing IKE SAs).

Large and small groups may use different sets of these protocols. When a large group of devices are communicating, the GCKS is likely to use the GSA_REKEY message for efficiency. This is shown in Figure 1. (Note: For clarity, IKE_SA_INIT is omitted from the figure.)

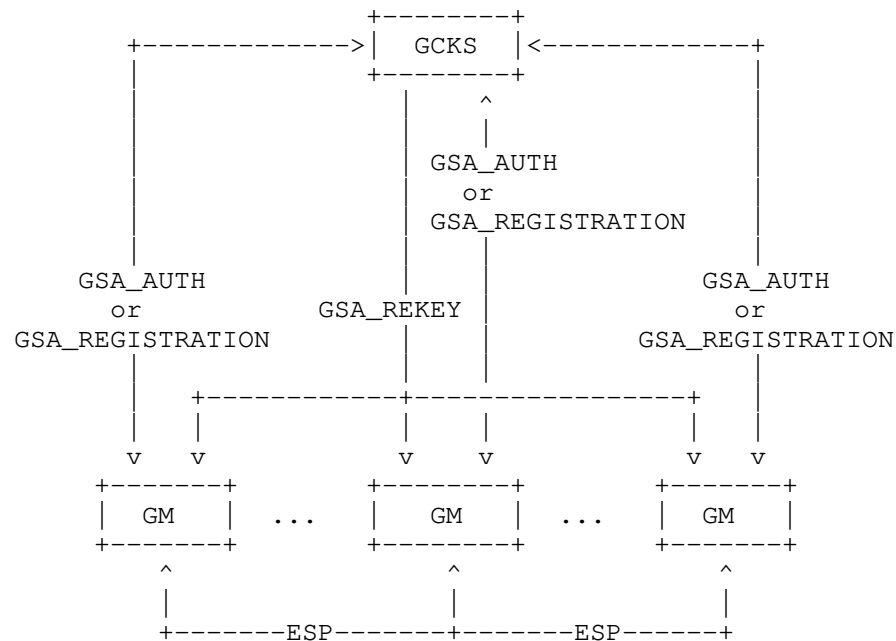


Figure 1: G-IKEv2 used in large groups

Alternatively, a small group may simply use the GSA_AUTH as a registration protocol, where the GCKS issues rekeys using the GSA_INBAND_REKEY within the same IKEv2 SA. The GCKS is also likely to be a GM in a small group (as shown in Figure 2.)

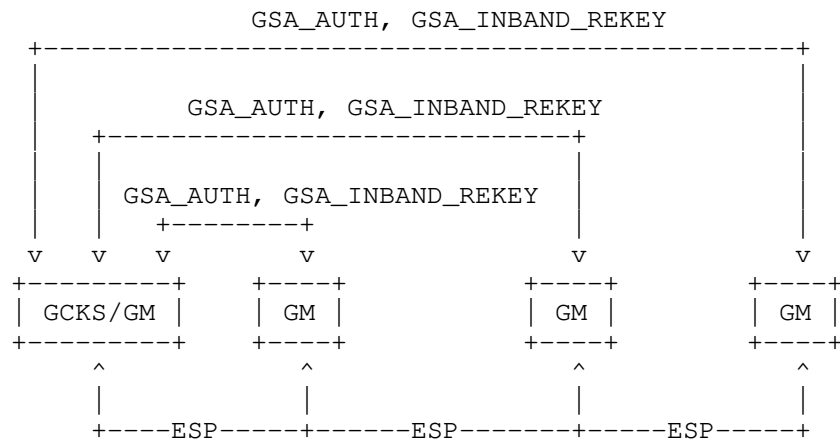


Figure 2: G-IKEv2 used in small groups

IKEv2 message semantics are preserved in that all communications consists of message request-response pairs. The exception to this rule is the GSA_REKEY exchange, which is a single message delivering group updates to the GMs.

G-IKEv2 conforms with the Multicast Group Security Architecture [RFC3740], and the Multicast Security (MSEC) Group Key Management Architecture [RFC4046]. G-IKEv2 replaces GDOI [RFC6407], which defines a similar group key management protocol using IKEv1 [RFC2409] (since deprecated by IKEv2). When G-IKEv2 is used, group key management use cases can benefit from the simplicity, increased robustness and cryptographic improvements of IKEv2 (see Appendix A of [RFC7296]).

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. G-IKEv2 Integration into IKEv2 Protocol

G-IKEv2 uses the security mechanisms of IKEv2 (peer authentication, confidentiality, message integrity) to ensure that only authenticated devices have access to the group policy and keys. The G-IKEv2 exchange further provides group authorization, and secure policy and key download from the GCKS to GMs. Some IKEv2 extensions require special handling if used with G-IKEv2. See Section 1.5 for more details.

It is assumed that readers are familiar with the IKEv2 protocol, so this document skips many details that are described in [RFC7296].

1.2.1. G-IKEv2 Transport and Port

G-IKEv2 SHOULD use UDP port 848, the same as GDOI [RFC6407], because they serve a similar function. They can use the same ports, just as IKEv1 and IKEv2 can share port 500. The version number in the IKE header distinguishes the G-IKEv2 protocol from GDOI protocol [RFC6407]. G-IKEv2 MAY also use the IKEv2 ports (500, 4500), which would provide a better integration with IKEv2. G-IKEv2 MAY also use TCP transport for registration (unicast) IKE SA, as defined in [RFC8229].

1.2.2. IKEv2 Header Initialization

The Major Version is (2) and Minor Version is (0) according to IKEv2 [RFC7296], and maintained in this document. The G-IKEv2 IKE_SA_INIT, GSA_AUTH, GSA_REGISTRATION and GSA_INBAND_REKEY use the IKE SPI according to IKEv2 [RFC7296], section 2.6.

1.3. G-IKEv2 Protocol

1.3.1. G-IKEv2 Payloads

In the following descriptions, the payloads contained in the G-IKEv2 messages are indicated by names as listed below.

Notation	Payload
AUTH	Authentication
CERT	Certificate
CERTREQ	Certificate Request
GSA	Group Security Association
HDR	IKEv2 Header
IDg	Identification - Group
IDi	Identification - Initiator
IDr	Identification - Responder
KD	Key Download
KE	Key Exchange
Ni, Nr	Nonce
SA	Security Association
SAg	Security Association - GM Supported Transforms

Payloads defined as part of other IKEv2 extensions MAY also be included in these messages. Payloads that may optionally appear will be shown in brackets, such as [CERTREQ], to indicate that a certificate request payload can optionally be included.

G-IKEv2 defines several new payloads not used in IKEv2:

- o IDg (Group ID) - The GM requests the GCKS for membership into the group by sending its IDg payload.
- o GSA (Group Security Association) - The GCKS sends the group policy to the GM using this payload.
- o KD (Key Download) - The GCKS sends the control and data keys to the GM using the KD payload.

- o SAg (Security Association - GM Supported Transforms) - the GM sends supported transforms, so that GCKS may select a policy appropriate for all members of the group.

The details of the contents of each payload are described in Section 2.

1.4. G-IKEv2 Member Registration and Secure Channel Establishment

The registration protocol consists of a minimum of two messages exchanges, `IKE_SA_INIT` and `GSA_AUTH`; member registration may have a few more messages exchanged if the EAP method, cookie challenge (for DoS protection) or negotiation of Diffie-Hellman group is included. Each exchange consists of request/response pairs. The first exchange `IKE_SA_INIT` is defined in IKEv2 [RFC7296]. This exchange negotiates cryptographic algorithms, exchanges nonces and does a Diffie-Hellman exchange between the group member (GM) and the Group Controller/Key Server (GCKS).

The second exchange `GSA_AUTH` authenticates the previous messages, exchanges identities and certificates. These messages are encrypted and integrity protected with keys established through the `IKE_SA_INIT` exchange, so the identities are hidden from eavesdroppers and all fields in all the messages are authenticated. The GCKS SHOULD authorize group members to be allowed into the group as part of the `GSA_AUTH` exchange. Once the GCKS accepts a group member to join a group it will download the data security keys (TEKs) and/or group key encrypting key (KEK) or KEK array as part of the `GSA_AUTH` response message.

1.4.1. `GSA_AUTH` exchange

After the group member and GCKS use the `IKE_SA_INIT` exchange to negotiate cryptographic algorithms, exchange nonces, and perform a Diffie-Hellman exchange as defined in IKEv2 [RFC7296], the `GSA_AUTH` exchange MUST complete before any other exchanges can be done. The security properties of the `GSA_AUTH` exchange are the same as the properties of the `IKE_AUTH` exchange. It is used to authenticate the `IKE_SA_INIT` messages, exchange identities and certificates. G-IKEv2 also uses this exchange for group member registration and authorization. Even though the `IKE_AUTH` does contain the `SA2`, `TSi`, and `TSr` payload the `GSA_AUTH` does not. They are not needed because policy is not negotiated between the group member and the GCKS, but instead downloaded from the GCKS to the group member.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK { IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, IDg, [SAg,] [N] }	-->

Figure 3: GSA_AUTH Request

After the IKE_SA_INIT exchange completes, the group member initiates a GSA_AUTH request to join a group indicated by the IDg payload. The GM MAY include an SAg payload declaring which Transforms that it is willing to accept. A GM that intends to emit data packets SHOULD include a Notify payload status type of SENDER, which enables the GCKS to provide any additional policy necessary by group senders.

Initiator (Member)	Responder (GCKS)
-----	-----
	<-- HDR, SK { IDr, [CERT,] AUTH, [GSA, KD,] [D,] }

Figure 4: GSA_AUTH Normal Response

The GCKS responds with IDr, optional CERT, and AUTH material as if it were an IKE_AUTH. It also informs the group member of the cryptographic policies of the group in the GSA payload and the key material in the KD payload. The GCKS can also include a Delete (D) payload instructing the group member to delete existing SAs it might have as the result of a previous group member registration. Note, that since the GCKS generally doesn't know which SAs the GM has, the SPI field in the Delete payload(s) SHOULD be set to zero in this case. (See more discussion on the Delete payload in Section 2.6.)

In addition to the IKEv2 error handling, the GCKS can reject the registration request when the IDg is invalid or authorization fails, etc. In these cases, see Section 2.7, the GSA_AUTH response will not include the GSA and KD, but will include a Notify payload indicating errors. If the group member included an SAg payload, and the GCKS chooses to evaluate it, and it detects that that group member cannot support the security policy defined for the group, then the GCKS SHOULD return a NO_PROPOSAL_CHOSEN. Other types of notifications can be AUTHORIZATION_FAILED or REGISTRATION_FAILED.

Initiator (Member)	Responder (GCKS)
-----	-----
	<-- HDR, SK { IDr, [CERT,] AUTH, N }

Figure 5: GSA_AUTH Error Response

If the group member finds the policy sent by the GCKS is unacceptable, the member SHOULD initiate GSA_REGISTRATION exchange sending IDg and the Notify NO_PROPOSAL_CHOSEN (see Section 1.4.2)).

1.4.2. GSA_REGISTRATION Exchange

When a secure channel is already established between a GM and the GCKS, the GM registration for a group can reuse the established secure channel. In this scenario the GM will use the GSA_REGISTRATION exchange. Payloads in the exchange are generated and processed as defined in Section 1.4.1.

Initiator (Member)	Responder (GCKS)
<hr/>	
HDR, SK {IDg, [SAG,][N] } -->	
	<-- HDR, SK { GSA, KD, [D] }

Figure 6: GSA_REGISTRATION Normal Exchange

As with GSA_AUTH exchange, the GCKS can reject the registration request when the IDg is invalid or authorization fails, or GM cannot support the security policy defined for the group (which can be concluded by GCKS by evaluation of SAG payload). In this case the GCKS returns an appropriate error notification as described in Section 1.4.1.

Initiator (Member)	Responder (GCKS)
<hr/>	
HDR, SK {IDg, [SAG,][N] } -->	
	<-- HDR, SK { N }

Figure 7: GSA_REGISTRATION Error Exchange

This exchange can also be used if the group member finds the policy sent by the GCKS is unacceptable or for some reason wants to unregister itself from the group. The group member SHOULD notify the GCKS by sending IDg and the Notify type NO_PROPOSAL_CHOSEN or REGISTRATION_FAILED, as shown below. The GCKS MUST unregister the group member.

Initiator (Member)	Responder (GCKS)
<hr/>	
HDR, SK {IDg, N } -->	
	<-- HDR, SK {}

Figure 8: GM Reporting Errors in GSA_REGISTRATION Exchange

1.4.3. GM Registration Operations

A G-IKEv2 Initiator (GM) requesting registration contacts the GCKS using the `IKE_SA_INIT` exchange and receives the response from the GCKS. This exchange is unchanged from the `IKE_SA_INIT` in IKEv2 protocol.

Upon completion of parsing and verifying the `IKE_SA_INIT` response, the GM sends the `GSA_AUTH` message with the IKEv2 payloads from `IKE_AUTH` (without the `SAi2`, `TSi` and `TSr` payloads) along with the Group ID informing the GCKS of the group the initiator wishes to join. An initiator intending to emit data traffic SHOULD send a `SENDER` Notify payload status. The `SENDER` not only signifies that it is a sender, but provides the initiator the ability to request Sender-ID values, in case the Data Security SA supports a counter mode cipher. Section 1.4.6) includes guidance on requesting Sender-ID values.

An initiator may be limited in the types of Transforms that it is able or willing to use, and may find it useful to inform the GCKS which Transforms that it is willing to accept. It can OPTIONALLY include an `SAG` payload, which can include ESP and/or AH Proposals. Each Proposal contains a list of Transforms that it is willing to support for that protocol. A Proposal of type ESP can include `ENCR`, `INTEG`, and `ESN` Transforms. A Proposal of type AH can include `INTEG`, and `ESN` Transforms. The SPI length of each Proposal in an `SAG` is set to zero, and thus the SPI field is null. The GCKS MUST ignore SPI field in the `SAG` payload. Generally, a single Proposal of each type will suffice, because the group member is not negotiating Transform sets, simply alerting the GCKS to restrictions it may have, however if the GM has restrictions on combination of algorithms, this can be expressed by sending several proposals.

Upon receiving the `GSA_AUTH` response, the initiator parses the response from the GCKS authenticating the exchange using the IKEv2 method, then processes the `GSA` and `KD`.

The `GSA` payload contains the security policy and cryptographic protocols used by the group. This policy describes the Rekey SA (`KEK`), if present, Data-security SAs (`TEK`), and other group policy (`GAP`). If the policy in the `GSA` payload is not acceptable to the GM, it SHOULD notify the GCKS by initiating a `GSA_REGISTRATION` exchange with a `NO_PROPOSAL_CHOSEN` Notify payload (see Section 1.4.2). Note, that this should normally not happen if the GM includes `SAG` payload in the `GSA_AUTH` request and the GCKS takes it into account. Finally the `KD` is parsed providing the keying material for the `TEK` and/or `KEK`. The GM interprets the `KD` key packets, where each key packet includes the keying material for SAs distributed in the `GSA` payload.

Keying material is matched by comparing the SPIs in the key packets to SPIs previously included in the GSA payloads. Once TEK keys and policy are matched, the GM provides them to the data security subsystem, and it is ready to send or receive packets matching the TEK policy.

The GSA KEK policy MUST include KEK attribute KEK_MESSAGE_ID with a Message ID. The Message ID in the KEK_MESSAGE_ID attribute MUST be checked against any previously received Message ID for this group. If it is less than the previously received number, it should be considered stale and ignored. This could happen if two GSA_AUTH exchanges happened in parallel, and the Message ID changed. This KEK_MESSAGE_ID is used by the GM to prevent GSA_REKEY message replay attacks. The first GSA_REKEY message that the GM receives from the GCKS must have a Message ID greater or equal to the Message ID received in the KEK_MESSAGE_ID attribute.

Once a GM has received GSA_REKEY policy during a registration the IKE SA may be closed. However, the GM SHOULD NOT close IKE SA, it is the GCKS who makes the decision whether to close or keep it, because depending on the policy the IKE SA may be used for inband rekeying for small groups.

1.4.4. GCKS Registration Operations

A G-IKEv2 GCKS passively listens for incoming requests from group members. When the GCKS receives an IKE_SA_INIT request, it selects an IKE proposal and generates a nonce and DH to include them in the IKE_SA_INIT response.

Upon receiving the GSA_AUTH request, the GCKS authenticates the group member using the same procedures as in the IKEv2 IKE_AUTH. The GCKS then authorizes the group member according to group policy before preparing to send the GSA_AUTH response. If the GCKS fails to authorize the GM, it will respond with an AUTHORIZATION_FAILED notify message.

The GSA_AUTH response will include the group policy in the GSA payload and keys in the KD payload. If the GCKS policy includes a group rekey option, this policy is constructed in the GSA KEK and the key is constructed in the KD KEK. The GSA KEK MUST include the KEK_MESSAGE_ID attribute, specifying the starting Message ID the GCKS will use when sending the GSA_REKEY message to the group member. This Message ID is used to prevent GSA_REKEY message replay attacks and will be increased each time a GSA_REKEY message is sent to the group. The GCKS data traffic policy is included in the GSA TEK and keys are included in the KD TEK. The GSA GAP MAY also be included to provide the ATD and/or DTD (Section 2.4.4.1) specifying activation

and deactivation delays for SAs generated from the TEKs. If the group member has indicated that it is a sender of data traffic and one or more Data Security SAs distributed in the GSA payload included a counter mode of operation, the GCKS responds with one or more SIDs (see Section 1.4.6).

If the GCKS receives a GSA_REGISTRATION exchange with a request to register a GM to a group, the GCKS will need to authorize the GM with the new group (IDg) and respond with the corresponding group policy and keys. If the GCKS fails to authorize the GM, it will respond with the AUTHORIZATION_FAILED notification.

If a group member includes an SAg in its GSA_AUTH or GSA_REGISTRATION request, the GCKS MAY evaluate it according to an implementation specific policy.

- o The GCKS could evaluate the list of Transforms and compare it to its current policy for the group. If the group member did not include all of the ESP or AH Transforms in its current policy, then it could return a NO_PROPOSAL_CHOSEN Notification.
- o The GCKS could store the list of Transforms, with the goal of migrating the group policy to a different Transform when all of the group members indicate that they can support that Transform.
- o The GCKS could store the list of Transforms and adjust the current group policy based on the capabilities of the devices as long as they fall within the acceptable security policy of the GCKS.

Depending on its policy, the GCKS may have no need for the IKE SA (e.g., it does not plan to initiate an GSA_INBAND_REKEY exchange). If the GM does not initiate another registration exchange or Notify (e.g., NO_PROPOSAL_CHOSEN), and also does not close the IKE SA and the GCKS is not intended to use the SA, then after a short period of time the GCKS SHOULD close the IKEv2 SA. The delay before closing provides for receipt of a GM's error notification in the event of packet loss.

1.4.5. Group Maintenance Channel

The GCKS is responsible for rekeying the secure group per the group policy. Rekeying is an operation whereby the GCKS provides replacement TEKs and KEK, deleting TEKs, and/or excluding group members. The GCKS may initiate a rekey message if group membership and/or policy has changed, or if the keys are about to expire. Two forms of group maintenance channels are provided in G-IKEv2 to push new policy to group members.

GSA_REKEY The GSA_REKEY exchange is an exchange initiated by the GCKS, where the rekey policy is usually delivered to group members using IP multicast as a transport. This is valuable for large and dynamic groups, and where policy may change frequently and an scalable rekeying method is required. When the GSA_REKEY exchange is used, the IKEv2 SA protecting the member registration exchanges is terminated, and group members await policy changes from the GCKS via the GSA_REKEY exchange.

GSA_INBAND_REKEY The GSA_INBAND_REKEY exchange is a rekey method using the IKEv2 SA that was setup to protecting the member registration exchange. This exchange allows the GCKS to rekey without using an independent GSA_REKEY exchange. The GSA_INBAND_REKEY exchange is useful when G-IKEv2 is used with a small group of cooperating devices.

1.4.5.1. GSA_REKEY Exchange

The GCKS initiates the G-IKEv2 Rekey securely, usually using IP multicast. Since this rekey does not require a response and it sends to multiple GMs, G-IKEv2 rekeying **MUST NOT** support IKE SA windowing. The GCKS rekey message replaces the rekey GSA KEK or KEK array, and/or creates a new Data-Security GSA TEK. The SID Download attribute in the Key Download payload (defined in Section 2.5.4) **MUST NOT** be part of the Rekey Exchange as this is sender specific information and the Rekey Exchange is group specific. The GCKS initiates the GSA_REKEY exchange as following:

Members (Responder)	GCKS (Initiator)
-----	-----
	<-- HDR, SK { GSA, KD, [D,] [AUTH] }

Figure 9: GSA_REKEY Exchange

HDR is defined in Section 2.1. The Message ID in this message will start with the same value the GCKS sent to the group members in the KEK attribute KEK_MESSAGE_ID during registration; this Message ID will be increased each time a new GSA_REKEY message is sent to the group members.

The GSA payload contains the current rekey and data security SAs. The GSA may contain a new rekey SA and/or a new data security SA, which, optionally contains an LKH rekey SA, Section 2.4.

The KD payload contains the keys for the policy included in the GSA. If the data security SA is being refreshed in this rekey message, the IPsec keys are updated in the KD, and/or if the rekey SA is being

refreshed in this rekey message, the rekey Key or the LKH KEK array is updated in the KD payload.

A Delete payload MAY be included to instruct the GM to delete existing SAs.

The AUTH payload MUST be included to authenticate the GSA_REKEY message if the authentication method is based on public key signatures and MUST NOT be included if it is based on shared secret. In a latter case, the fact that a GM can decrypt the GSA_REKEY message and verify its ICV proves that the sender of this message knows the current KEK, thus authenticating that the sender is a member of the group. Shared secret authentication doesn't provide source origin authentication. For this reason using it as authentication method for multicast Rekey is NOT RECOMMENDED unless source origin authentication is not required (for example, in a small group of highly trusted GMs). If AUTH payload is included then the Auth Method field MUST be one specifying using digital signatures.

During group member registration, the GCKS sends the authentication key in the GSA KEK payload, KEK_AUTH_KEY attribute, which the group member uses to authenticate the key server. Before the current Authentication Key expires, the GCKS will send a new KEK_AUTH_KEY to the group members in a GSA_REKEY message. The AUTH key that is used in the rekey message may be not the same as the authentication key used in GSA_AUTH.

1.4.5.1.1. GSA_REKEY GCKS Operations

The GCKS builds the rekey message with a Message ID value that is one greater than the value included in the previous rekey. If the message is using a new KEK attribute, the Message ID is reset to 1 in this message. The GSA, KD, and D payloads follow with the same characteristics as in the GSA Registration exchange.

If present the AUTH payload is created as follows. First the message is prepared, all payloads are formed and included in the message, but the content of the Encrypted payload is not yet encrypted. However, the Encrypted payload must be fully formed, including correct values in IV, Padding and Pad Length and fields. The AUTH payload is included in the message with the correct values in the Payload Header (including Next Payload, Payload Length and Auth Method fields). The Authentication Data field is zeroed for the purposes of signature calculation, but if Digital Signature authentication method is in use, then the ASN.1 Length and the AlgorithmIdentifier fields must be properly filled in, see [RFC7427]. The signature is computed using the signature algorithm from the KEK_AUTH_METHOD attribute (along with the KEK_AUTH_HASH if KEK_AUTH_METHOD is not Digital Signature)

and the private key corresponding to the public key from the KEK_AUTH_KEY attribute. It is computed over the block of data starting from the first octet of IKE Header (but non including non-ESP marker if it is present) to the last octet of the (not yet encrypted) Encrypted Payload (i.e. up to and including Pad Length field). Then the signature is placed into the Signature Value of the AUTH payload, the content of the Encrypted payload is encrypted and the ICV is computed using current KEK keys.

Because GSA_REKEY messages are not acknowledged and could be discarded by the network, one or more GMs may not receive the message. To mitigate such lost messages, during a rekey event the GCKS may transmit several GSA_REKEY messages with the new policy. The retransmitted messages MUST be bitwise identical and SHOULD be sent within a short time interval (a few seconds) to ensure that time-to-live would not be substantially skewed for the GMs that would receive different copies of the messages.

GCKS may also include one or several KEK_NEXT_SPI/TEK_NEXT_SPI attributes specifying SPIs for the prospected rekeys, so that listening GMs are able to detect lost rekey messages and recover from this situation. See Sections Section 2.4.2.1.6 and Section 2.4.3.1.4 for more detail.

1.4.5.1.2. GSA_REKEY GM Operations

When a group member receives the Rekey Message from the GCKS it decrypts the message using the current KEK, validates the signature using the public key retrieved in a previous G-IKEv2 exchange if AUTH payload is present, verifies the Message ID, and processes the GSA and KD payloads. The group member then downloads the new data security SA and/or new Rekey SA. The parsing of the payloads is identical to the parsing done in the registration exchange.

Replay protection is achieved by a group member rejecting a GSA_REKEY message which has a Message ID smaller than the current Message ID that the GM is expecting. The GM expects the Message ID in the first GSA_REKEY message it receives to be equal or greater than the message id it receives in the KEK_MESSAGE_ID attribute. The GM expects the message ID in subsequent GSA_REKEY messages to be greater than the last valid GSA_REKEY message ID it received.

If the GSA payload includes a Data-Security SA including a counter-modes of operation and the receiving group member is a sender for that SA, the group member uses its current SID value with the Data-Security SAs to create counter-mode nonces. If it is a sender and does not hold a current SID value, it MUST NOT install the Data-Security SAs. It MAY initiate a GSA_REGISTRATION exchange to the

GCKS in order to obtain an SID value (along with current group policy).

Once a new Rekey SA is installed as a result of GSA_REKEY message, the current Rekey SA (over which the message was received) MUST be silently deleted after waiting DEACTIVATION_TIME_DELAY interval regardless of its expiration time. If the GSA TEK payload includes TEK_REKEY_SPI attribute then after installing a new Data-Security SA the old one, identified by the SPI in this attribute, MUST be silently deleted after waiting DEACTIVATION_TIME_DELAY interval regardless of its expiration time.

If a Data-Security SA is not rekeyed yet and is about to expire (a "soft lifetime" expiration is described in Section 4.4.2.1 of [RFC4301]), the GM SHOULD initiate a registration to the GCKS. This registration serves as a request for current SAs, and will result in the download of replacement SAs, assuming the GCKS policy has created them. A GM SHOULD also initiate a registration request if a Rekey SA is about to expire and not yet replaced with a new one.

1.4.5.1.3. Forward and Backward Access Control

Through the G-IKEv2 rekey, G-IKEv2 supports algorithms such as LKH that have the property of denying access to a new group key by a member removed from the group (forward access control) and to an old group key by a member added to the group (backward access control). An unrelated notion to PFS, "forward access control" and "backward access control" have been called "perfect forward security" and "perfect backward security" in the literature [RFC2627].

Group management algorithms providing forward and backward access control other than LKH have been proposed in the literature, including OFT [OFT] and Subset Difference [NNL]. These algorithms could be used with G-IKEv2, but are not specified as a part of this document.

Support for group management algorithms are supported via the KEY_MANAGEMENT_ALGORITHM attribute which is sent in the GSA KEK policy. G-IKEv2 specifies one method by which LKH can be used for forward and backward access control. Other methods of using LKH, as well as other group management algorithms such as OFT or Subset Difference may be added to G-IKEv2 as part of a later document.

1.4.5.1.3.1. Forward Access Control Requirements

When group membership is altered using a group management algorithm new GSA TEKs (and their associated keys) are usually also needed.

New GSAs and keys ensure that members who were denied access can no longer participate in the group.

If forward access control is a desired property of the group, new GSA TEKs and the associated key packets in the KD payload MUST NOT be included in a G-IKEv2 rekey message which changes group membership. This is required because the GSA TEK policy and the associated key packets in the KD payload are not protected with the new KEK. A second G-IKEv2 rekey message can deliver the new GSA TEKs and their associated key packets because it will be protected with the new KEK, and thus will not be visible to the members who were denied access.

If forward access control policy for the group includes keeping group policy changes from members that are denied access to the group, then two sequential G-IKEv2 rekey messages changing the group KEK MUST be sent by the GCKS. The first G-IKEv2 rekey message creates a new KEK for the group. Group members, which are denied access, will not be able to access the new KEK, but will see the group policy since the G-IKEv2 rekey message is protected under the current KEK. A subsequent G-IKEv2 rekey message containing the changed group policy and again changing the KEK allows complete forward access control. A G-IKEv2 rekey message MUST NOT change the policy without creating a new KEK.

If other methods of using LKH or other group management algorithms are added to G-IKEv2, those methods MAY remove the above restrictions requiring multiple G-IKEv2 rekey messages, providing those methods specify how the forward access control policy is maintained within a single G-IKEv2 rekey message.

1.4.5.1.4. Fragmentation

IKE fragmentation [RFC7383] can be used to perform fragmentation of large GSA_REKEY messages, however when the GSA_REKEY message is emitted as an IP multicast packet there is a lack of response from the GMs. This has the following implications.

- o Policy regarding the use of IKE fragmentation is implicit. If a GCKS detects that all GMs have negotiated support of IKE fragmentation in IKE_SA_INIT, then it MAY use IKE fragmentation on large GSA_REKEY exchange messages.
- o The GCKS must always use IKE fragmentation based on a known fragmentation threshold (unspecified in this memo), as there is no way to check if fragmentation is needed by first sending unfragmented messages and waiting for response.

- o PMTU probing cannot be performed due to lack of GSA_REKEY response message.

1.4.5.2. GSA_INBAND_REKEY Exchange

When the IKEv2 SA protecting the member registration exchange is maintained while group member participates in the group, the GCKS can use the GSA_INBAND_REKEY exchange to individually provide policy updates to the group member.

Member (Responder)		GCKS (Initiator)
-----		-----
	<--	HDR, SK { GSA, KD, [D,] }
HDR, SK { }	-->	

Figure 10: GSA_INBAND_REKEY Exchange

Because this is an IKEv2 exchange, the HDR is treated as defined in [RFC7296].

1.4.5.2.1. GSA_INBAND_REKEY GCKS Operations

The GSA, KD, and D payloads are built in the same manner as in a registration exchange.

1.4.5.2.2. GSA_INBAND_REKEY GM Operations

The GM processes the GSA, KD, and D payloads in the same manner as if they were received in a registration exchange.

1.4.5.3. Deletion of SAs

There are occasions when the GCKS may want to signal to group members to delete policy at the end of a broadcast, or if group policy has changed. Deletion of keys MAY be accomplished by sending the G-IKEv2 Delete Payload [RFC7296], section 3.11 as part of the GSA_REKEY Exchange as shown below.

Members (Responder)		GCKS (Initiator)
-----		-----
	<--	HDR, SK { [GSA], [KD], [D,] [AUTH] }

Figure 11: SA Deletion in GSA_REKEY

The GSA MAY specify the remaining active time of the remaining policy by using the DTD attribute in the GSA GAP. If a GCKS has no further SAs to send to group members, the GSA and KD payloads MUST be omitted from the message. There may be circumstances where the GCKS may want

to start over with a clean slate. If the administrator is no longer confident in the integrity of the group, the GCKS can signal deletion of all the policies of a particular TEK protocol by sending a TEK with a SPI value equal to zero in the delete payload. For example, if the GCKS wishes to remove all the KEKs and all the TEKs in the group, the GCKS SHOULD send a Delete payload with a SPI of zero and a protocol_id of a TEK protocol_id value defined in Section 2.4.3, followed by another Delete payload with a SPI of zero and protocol_id of zero, indicating that the KEK SA should be deleted.

1.4.6. Counter-based modes of operation

Several new counter-based modes of operation have been specified for ESP (e.g., AES-CTR [RFC3686], AES-GCM [RFC4106], AES-CCM [RFC4309], AES-GMAC [RFC4543]) and AH (e.g., AES-GMAC [RFC4543]). These counter-based modes require that no two senders in the group ever send a packet with the same Initialization Vector (IV) using the same cipher key and mode. This requirement is met in G-IKEv2 when the following requirements are met:

- o The GCKS distributes a unique key for each Data-Security SA.
- o The GCKS uses the method described in [RFC6054], which assigns each sender a portion of the IV space by provisioning each sender with one or more unique SID values.

1.4.6.1. Allocation of SIDs

When at least one Data-Security SA included in the group policy includes a counter-based mode of operation, the GCKS automatically allocates and distributes one SID to each group member acting in the role of sender on the Data-Security SA. The SID value is used exclusively by the group member to which it was allocated. The group member uses the same SID for each Data-Security SA specifying the use of a counter-based mode of operation. A GCKS MUST distribute unique keys for each Data-Security SA including a counter-based mode of operation in order to maintain unique key and nonce usage.

During registration, the group member can choose to request one or more SID values. Requesting a value of 1 is not necessary since the GCKS will automatically allocate exactly one to the group member. A group member MUST request as many SIDs matching the number of encryption modules in which it will be installing the TEKs in the outbound direction. Alternatively, a group member MAY request more than one SID and use them serially. This could be useful when it is anticipated that the group member will exhaust their range of Data-Security SA nonces using a single SID too quickly (e.g., before the time-based policy in the TEK expires).

When the group policy includes a counter-based mode of operation, a GCKS SHOULD use the following method to allocate SID values, which ensures that each SID will be allocated to just one group member.

1. A GCKS maintains an SID-counter, which records the SIDs that have been allocated. SIDs are allocated sequentially, with zero as the first allocated SID.
2. Each time an SID is allocated, the current value of the counter is saved and allocated to the group member. The SID-counter is then incremented in preparation for the next allocation.
3. When the GCKS specifies a counter-based mode of operation in the Data Security SA a group member may request a count of SIDs during registration in a Notify payload information of type SENDER. When the GCKS receives this request, it increments the SID-counter once for each requested SID, and distributes each SID value to the group member. The GCKS SHOULD have a policy-defined upper bound for the number of SIDs that it will return irrespective of the number requested by the GM.
4. A GCKS allocates new SID values for each GSA_REGISTRATION exchange originated by a sender, regardless of whether a group member had previously contacted the GCKS. In this way, the GCKS is not required to maintaining a record of which SID values it had previously allocated to each group member. More importantly, since the GCKS cannot reliably detect whether the group member had sent data on the current group Data-Security SAs it does not know what Data-Security counter-mode nonce values that a group member has used. By distributing new SID values, the key server ensures that each time a conforming group member installs a Data-Security SA it will use a unique set of counter-based mode nonces.
5. When the SID-counter maintained by the GCKS reaches its final SID value, no more SID values can be distributed. Before distributing any new SID values, the GCKS MUST delete the Data-Security SAs for the group, followed by creation of new Data-Security SAs, and resetting the SID-counter to its initial value.
6. The GCKS SHOULD send a GSA_REKEY message deleting all Data-Security SAs and the Rekey SA for the group. This will result in the group members initiating a new GSA_REGISTRATION exchange, in which they will receive both new SID values and new Data-Security SAs. The new SID values can safely be used because they are only used with the new Data-Security SAs. Note that deletion of the Rekey SA is necessary to ensure that group members receiving a GSA_REKEY exchange before the re-register do not inadvertently use their old SIDs with the new Data-Security SAs. Using the method above, at no time can

two group members use the same IV values with the same Data-Security SA key.

1.4.6.2. GM Usage of SIDs

A GM applies the SID to Data Security SA as follows.

1. The most significant bits `NUMBER_OF_SID_BITS` of the IV are taken to be the SID field of the IV.
2. The SID is placed in the least significant bits of the SID field, where any unused most significant bits are set to zero. If the SID value doesn't fit into the `NUMBER_OF_SID_BITS` bits, then the GM MUST treat this as a fatal error and re-register to the group.

1.5. Interaction with IKEv2 Protocol Extensions

IKEv2 defines a number of extensions that can be used to extend protocol functionality. G-IKEv2 is compatible with most of such extensions. In particular, EAP authentication defined in [RFC7296] can be used to establish registration IKE SA, as well as Secure Password authentication ([RFC6467]). G-IKEv2 is compatible with and can use IKEv2 Session Resumption [RFC5723] except that a GM would include the initial ticket request in a `GSA_AUTH` exchange instead of an `IKE_AUTH` exchange. G-IKEv2 is also compatible with Quantum Safe Key Exchange framework, defined in [I-D.tjhai-ipsecme-hybrid-qske-ikev2].

Some IKEv2 extensions however require special handling if used in G-IKEv2.

1.5.1. Postquantum Preshared Keys for IKEv2

G-IKEv2 can take advantage of the protection provided by Postquantum Preshared Keys (PPK) for IKEv2 [I-D.ietf-ipsecme-qr-ikev2]. However, the use of PPK leaves the initial IKE SA susceptible to quantum computer (QC) attacks. So, if PPK was used for IKE SA setup, the GCKS MUST NOT send GSA and KD payloads in the `GSA_AUTH` response message. Instead, the GCKS MUST return a new notification `REKEY_IS_NEEDED`. Upon receiving this notification in the `GSA_AUTH` response the GM MUST perform an IKE SA rekey and then initiate a new `GSA_REGISTRATION` request for the same group. Below are possible scenarios involving using PPK.

GM begins `IKE_SA_INIT` requesting PPK, and GCKS responds with willingness to do it, or aborts according to its "mandatory_or_not" flag:

```

Initiator (Member)                      Responder (GCKS)
-----
HDR, SAi1, KEi, Ni, N(USE_PPK) --->
<--- HDR, SAR1, KEr, Nr, [CERTREQ],
      N(USE_PPK)

```

Figure 12: IKE_SA_INIT Exchange requesting using PPK

GM begins GSA_AUTH with PPK_ID; if using PPK is not mandatory for the GM, N(NO_PPK_AUTH) is included too:

```

Initiator (Member)                      Responder (GCKS)
-----
HDR, SK {IDi, AUTH, IDg,
N(PPK_IDENTITY), N(NO_PPK_AUTH) } --->

```

Figure 13: GSA_AUTH Request using PPK

If GCKS has no such PPK and using PPK is not mandatory for it and N(NO_PPK_AUTH) is included, then the GCKS continues w/o PPK; in this case no rekey is needed:

```

Initiator (Member)                      Responder (GCKS)
-----
<--- HDR, SK { IDr, AUTH, GSA, KD }

```

Figure 14: GSA_AUTH Response using no PPK

If GCKS has no such PPK and either N(NO_PPK_AUTH) is missing or using PPK is mandatory for GCKS, the GCKS aborts the exchange:

```

Initiator (Member)                      Responder (GCKS)
-----
<--- HDR, SK { N(AUTHENTICATION_FAILED) }

```

Figure 15: GSA_AUTH Error Response

Assuming GCKS has a proper PPK the GCKS continues with request to GM to immediately perform a rekey:

```

Initiator (Member)                      Responder (GCKS)
-----
<--- HDR, SK{IDr, AUTH, N(PPK_IDENTITY),
      N(REKEY_IS_NEEDED) }

```

Figure 16: GSA_AUTH Response using PPK

GM initiates CREATE_CHILD_SA to rekey IKE SA and then makes a new registration request for the same group over the new IKE SA:

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK {SA, Ni, KEi } --->	
	<--- HDR, SK {SA, Nr, KEr }
HDR, SK {IDg } --->	
	<--- HDR, SK { GSA, KD }

Figure 17: Rekeying IKE SA followed by GSA_REGISTRATION Exchange

2. Header and Payload Formats

Refer to IKEv2 [RFC7296] for existing payloads. Some payloads used in G-IKEv2 exchanges are not aligned to 4-octet boundaries, which is also the case for some IKEv2 payloads (see Section 3.2 of [RFC7296]).

2.1. The G-IKEv2 Header

G-IKEv2 uses the same IKE header format as specified in [RFC7296] section 3.1.

Several new payload formats are required in the group security exchanges.

Next Payload Type	Value
-----	-----
Group Identification (IDg)	50
Group Security Association (GSA)	51
Key Download (KD)	52

New exchange types GSA_AUTH, GSA_REGISTRATION and GSA_REKEY are added to the IKEv2 [RFC7296] protocol.

Exchange Type	Value
-----	-----
GSA_AUTH	39
GSA_REGISTRATION	40
GSA_REKEY	41
GSA_INBAND_REKEY	TBD

Major Version is 2 and Minor Version is 0 as in IKEv2 [RFC7296]. IKE SA Initiator's SPI, IKE SA Responder's SPI, Flags, Message ID, and Length are as specified in [RFC7296].

2.2. Group Identification (IDg) Payload

The IDg Payload allows the group member to indicate which group it wants to join. The payload is constructed by using the IKEv2 Identification Payload (section 3.5 of [RFC7296]). ID type ID_KEY_ID MUST be supported. ID types ID_IPV4_ADDR, ID_FQDN, ID_RFC822_ADDR, ID_IPV6_ADDR SHOULD be supported. ID types ID_DER_ASN1_DN and ID_DER_ASN1_GN are not expected to be used.

2.3. Security Association - GM Supported Transforms (Sag)

The Sag payload declares which Transforms a GM is willing to accept. The payload is constructed using the format of the IKEv2 Security Association payload (section 3.3 of [RFC7296]). The Payload Type for Sag is identical to the SA Payload Type (33).

2.4. Group Security Association Payload

The Group Security Association payload is used by the GCKS to assert security attributes for both Rekey and Data-security SAs.

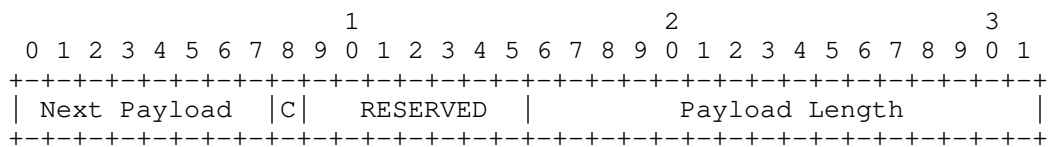


Figure 18: GSA Payload Format

The Security Association Payload fields are defined as follows:

- o Next Payload (1 octet) -- Identifies the next payload type for the G-IKEv2 registration or the G-IKEv2 rekey message.
- o Critical (1 bit) -- Set according to [RFC7296].
- o RESERVED (7 bits) -- Must be zero.
- o Payload Length (2 octets) -- Is the octet length of the current payload including the generic header and all TEK and KEK policies.

2.4.1. GSA Policy

Following the GSA generic payload header are GSA policies for group rekeying (KEK), data traffic SAs (TEK) and/or Group Associated Policy (GAP). There may be zero or one GSA KEK policy, zero or one GAP policies, and zero or more GSA TEK policies, where either one GSA KEK or GSA TEK payload MUST be present.

This latitude allows various group policies to be accommodated. For example if the group policy does not require the use of a Rekey SA, the GCKS would not need to send a GSA KEK attribute to the group member since all SA updates would be performed using the Registration SA. Alternatively, group policy might use a Rekey SA but choose to download a KEK to the group member only as part of the Registration SA. Therefore, the GSA KEK policy would not be necessary as part of the GSA_REKEY message.

Specifying multiple GSA TEKs allows multiple related data streams (e.g., video, audio, and text) to be associated with a session, but each protected with an individual security association policy.

A GAP payload allows for the distribution of group-wise policy, such as instructions for when to activate and de-activate SAs.

Policies are distributed in substructures to the GSA payload, and include the following header.

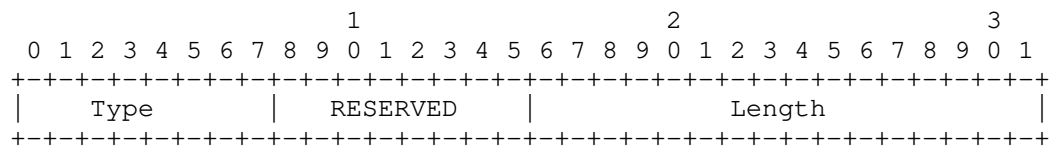


Figure 19: GSA Policy Generic Header Format

The payload fields are defined as follows:

- o Type (1 octet) -- Identifies the substructure type. In the following table the terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

Type	Value
-----	-----
Reserved	0
KEK	1
GAP	2
TEK	3
Unassigned	4-127
Private Use	128-255

- o RESERVED (1 octet) -- Unused, set to zero.
- o Length (2 octets) -- Length in octets of the substructure, including its header.

2.4.2. KEK Policy

The GSA KEK policy contains security attributes for the KEK method for a group and parameters specific to the G-IKEv2 registration operation. The source and destination traffic selectors describe the network identities used for the rekey messages.

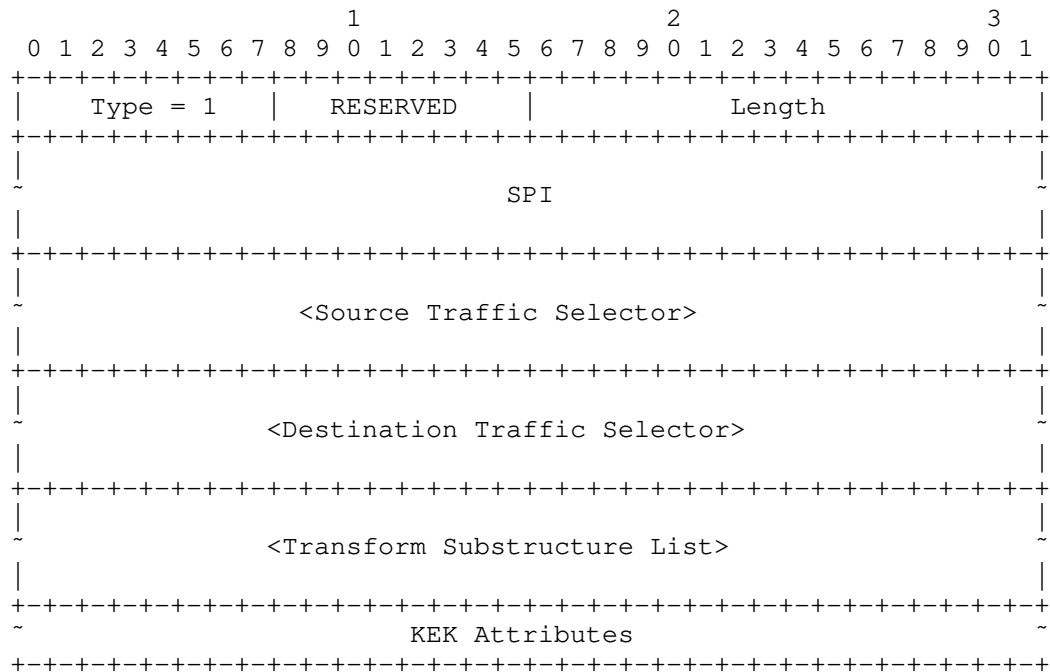


Figure 20: KEK Policy Format

The GSA KEK Payload fields are defined as follows:

- o Type = 1 (1 octet) -- Identifies the GSA payload type as KEK in the G-IKEv2 registration or the G-IKEv2 rekey message.
- o RESERVED (1 octet) -- Must be zero.
- o Length (2 octets) -- Length of this structure including KEK attributes.
- o SPI (16 octets) -- Security Parameter Index for the rekey message. The SPI must be the IKEv2 Header SPI pair where the first 8 octets become the "Initiator's SPI" field in the G-IKEv2 rekey message IKEv2 HDR, and the second 8 octets become the "Responder's SPI" in the same HDR. As described above, these SPIs are assigned by the

GCKS. When selecting SPI the GCKS MUST make sure that the sole first 8 octets (corresponding to "Initiator's SPI" field in the IKEv2 header) uniquely identify the Rekey SA.

- o Source & Destination Traffic Selectors -- Substructures describing the source and destination of the network identities. These identities refer to the source and destination of the next KEK rekey SA. Defined format and values are specified by IKEv2 [RFC7296], section 3.13.1.
- o Transform Substructure List -- A list of Transform Substructures specifies the transform information. The format is defined in IKEv2 [RFC7296], section 3.3.2, and values are described in the IKEv2 registries [IKEV2-IANA]. Valid Transform Types are ENCR, INTEG. The Last Substruc value in each Transform Substructure will be set to 3 except for the last one in the list, which is set to 0.
- o KEK Attributes -- Contains KEK policy attributes associated with the group. The following sections describe the possible attributes. Any or all attributes may be optional, depending on the group policy.

2.4.2.1. KEK Attributes

The following attributes may be present in a GSA KEK policy. The attributes must follow the format defined in the IKEv2 [RFC7296] section 3.3.5. In the table, attributes that are defined as TV are marked as Basic (B); attributes that are defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

KEK Attributes	Value	Type	Mandatory
-----	----	----	-----
Reserved	0		
KEK_MANAGEMENT_ALGORITHM	1	B	N
Reserved	2		
Reserved	3		
KEK_KEY_LIFETIME	4	V	Y
Reserved	5		
KEK_AUTH_METHOD	6	B	Y
KEK_AUTH_HASH	7	B	N
KEK_MESSAGE_ID	8	V	Y (*)
KEK_NEXT_SPI	9	V	N
Unassigned	10-16383		
Private Use	16384-32767		

(*) the KEK_MESSAGE_ID MUST be included in a G-IKEv2 registration message and MUST NOT be included in rekey messages.

The following attributes may only be included in a G-IKEv2 registration message: KEK_MANAGEMENT_ALGORITHM, KEK_MESSAGE_ID.

2.4.2.1.1. KEK_MANAGEMENT_ALGORITHM

The KEK_MANAGEMENT_ALGORITHM attribute specifies the group KEK management algorithm used to provide forward or backward access control (i.e., used to exclude group members). Defined values are specified in the following table. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

KEK Management Type	Value
-----	-----
Reserved	0
LKH	1
Unassigned	2-16383
Private Use	16384-32767

2.4.2.1.2. KEK_KEY_LIFETIME

The KEK_KEY_LIFETIME attribute specifies the maximum time for which the KEK is valid. The GCKS may refresh the KEK at any time before the end of the valid period. The value is a four (4) octet number defining a valid time period in seconds.

2.4.2.1.3. KEK_AUTH_METHOD

The KEK_AUTH_METHOD attribute specifies the method of authentication used. This value is from the IKEv2 Authentication Method registry [IKEV2-IANA]. The method must either specify using some public key signatures or Shared Key Message Integrity Code. Other authentication methods MUST NOT be used.

2.4.2.1.4. KEK_AUTH_HASH

The KEK_AUTH_HASH attribute specifies the hash algorithm used to generate the AUTH key to authenticate GSA_REKEY messages. Hash algorithms are defined in IANA registry IKEv2 Hash Algorithms [IKEV2-IANA].

This attribute SHOULD NOT be sent if the KEK_AUTH_METHOD implies a particular hash algorithm (e.g., for DSA-based algorithms). Furthermore, it is not necessary for the GCKS to send it if the GM is known to support the algorithm because it declared it in a

SIGNATURE_HASH_ALGORITHMS notification during registration (see [RFC7427]).

2.4.2.1.5. KEK_MESSAGE_ID

The KEK_MESSAGE_ID attribute defines the initial Message ID to be used by the GCKS in the GSA_REKEY messages. The Message ID is a 4 octet unsigned integer in network byte order.

2.4.2.1.6. KEK_NEXT_SPI

The KEK_NEXT_SPI attribute may optionally be included by GCKS in GSA_REKEY message, indicating what IKE SPIs are intended be used for the next rekey SA. The attribute data MUST be 16 octets in length specifying the pair of IKE SPIs as they appear in the IKE header. Multiple attributes of this type MAY be included, meaning that any of the supplied SPIs can be used for the next rekey.

The GM may save these values and if later the GM starts receiving IKE messages with one of these SPIs without seeing a rekey message over the current rekey SA, this may be used as an indication, that the rekey message was lost on its way to this GM. In this case the GM SHOULD re-register to the group.

Note, that this method of detecting missed rekeys can only be used by passive GMs, i.e. those, that only listen and don't send data. It's also no point to include this attribute in the GSA_INBAND_REKEY messages, since they use reliable transport. Note also, that the GCKS is free to forget its promises and not to use the SPIs it sent in the KEK_NEXT_SPI attributes before (e.g. in case of GCKS reboot), so the GM must only treat these information as a "best effort" made by GCKS to prepare for future rekeys.

2.4.3. GSA TEK Policy

The GSA TEK policy contains security attributes for a single TEK associated with a group.

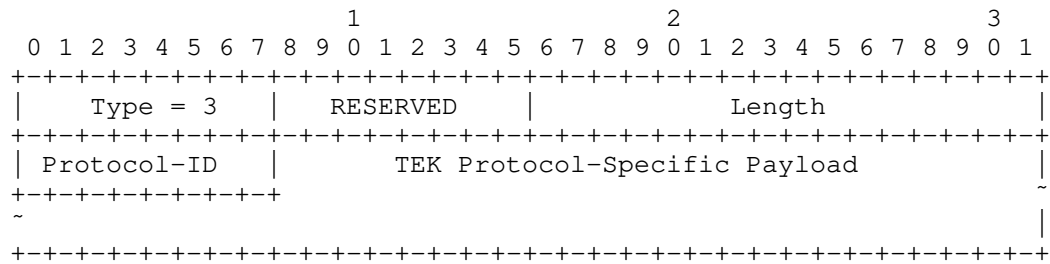


Figure 21: TEK Policy Generic Header Format

The GSA TEK Payload fields are defined as follows:

- o Type = 3 (1 octet) -- Identifies the GSA payload type as TEK in the G-IKEv2 registration or the G-IKEv2 rekey message.
- o RESERVED (1 octet) -- Must be zero.
- o Length (2 octets) -- Length of this structure, including the TEK Protocol-Specific Payload.
- o Protocol-ID (1 octet) -- Value specifying the Security Protocol. The following table defines values for the Security Protocol. Support for the GSA_PROTO_IPSEC_AH GSA TEK is OPTIONAL. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

Protocol ID	Value
-----	-----
Reserved	0
GSA_PROTO_IPSEC_ESP	1
GSA_PROTO_IPSEC_AH	2
Unassigned	3-127
Private Use	128-255

- o TEK Protocol-Specific Payload (variable) -- Payload which describes the attributes specific for the Protocol-ID.

2.4.3.1. TEK ESP and AH Protocol-Specific Policy

The TEK Protocol-Specific policy contains two traffic selectors one for the source and one for the destination of the protected traffic, SPI, Transforms, and Attributes.

The TEK Protocol-Specific policy for ESP and AH is as follows:

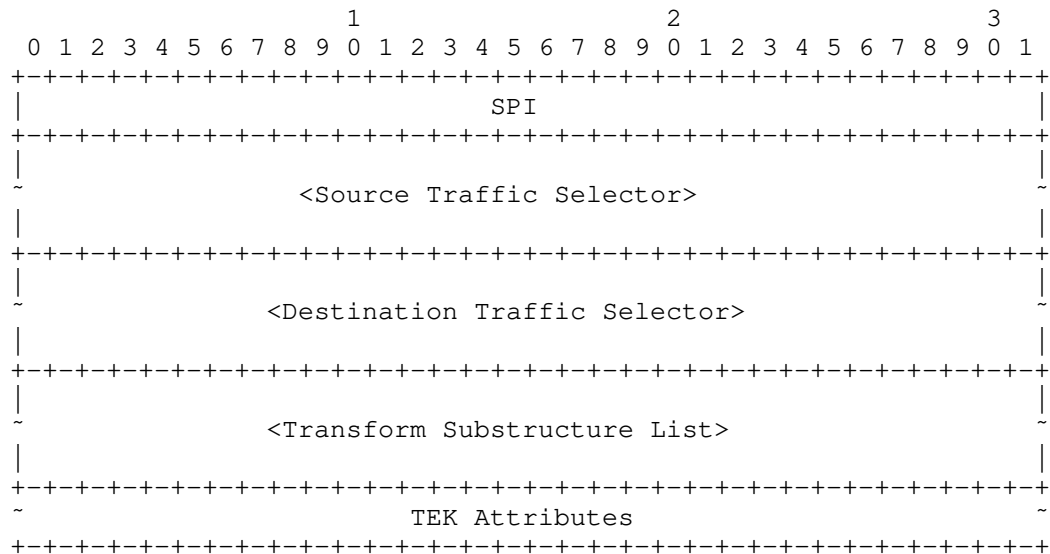


Figure 22: AH and ESP TEK Policy Format

The GSA TEK Policy fields are defined as follows:

- o SPI (4 octets) -- Security Parameter Index.
- o Source & Destination Traffic Selectors - The traffic selectors describe the source and the destination of the protected traffic. The format and values are defined in IKEv2 [RFC7296], section 3.13.1.
- o Transform Substructure List -- A list of Transform Substructures specifies the transform information. The format is defined in IKEv2 [RFC7296], section 3.3.2, and values are described in the IKEv2 registries [IKEV2-IANA]. Valid Transform Types for ESP are ENCR, INTEG, and ESN. Valid Transform Types for AH are INTEG and ESN. The Last Substruc value in each Transform Substructure will be set to 3 except for the last one in the list, which is set to 0. A Transform Substructure with attributes (e.g., the ENCR Key Length), they are included within the Transform Substructure as usual.
- o TEK Attributes -- Contains the TEK policy attributes associated with the group, in the format defined in Section 3.3.5 of [RFC7296]. All attributes are optional, depending on the group policy.

Attribute Types are as follows. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

TEK Attributes -----	Value -----	Type ----	Mandatory -----
Reserved	0		
TEK_KEY_LIFETIME	1	V	N
TEK_MODE	2	B	Y
TEK_REKEY_SPI	3	V	N
TEK_NEXT_SPI	4	V	N
Unassigned	5-16383		
Private Use	16384-32767		

It is NOT RECOMMENDED that the GCKS distribute both ESP and AH Protocol-Specific Policies for the same set of Traffic Selectors.

2.4.3.1.1. TEK_KEY_LIFETIME

The TEK_KEY_LIFETIME attribute specifies the maximum time for which the TEK is valid. When the TEK expires, the AH or ESP security association and all keys downloaded under the security association are discarded. The GCKS may refresh the TEK at any time before the end of the valid period.

The value is a four (4) octet number defining a valid time period in seconds. If unspecified the default value of 28800 seconds (8 hours) shall be assumed.

2.4.3.1.2. TEK_MODE

The value of 0 is used for tunnel mode and 1 for transport mode. In the absence of this attribute tunnel mode will be used.

2.4.3.1.3. TEK_REKEY_SPI

This attribute contains an SPI for the SA that is being rekeyed. The size of SPI depends on the protocol, for ESP and AH it is 4 octets, so the size of the data MUST be 4 octets for AH and ESP.

If this attribute is included in the rekey message, the GM SHOULD delete the SA corresponding to this SPI once the new SA is installed and regardless of the expiration time of the SA to be deleted (but after waiting DEACTIVATION_TIME_DELAY time period).

2.4.3.1.4. TEK_NEXT_SPI

This attribute contains an SPI that the GCKS reserved for the next rekey. The size of SPI depends on the protocol, for ESP and AH it is 4 octets, so the size of the data MUST be 4 octets for AH and ESP. Multiple attributes of this type MAY be included, which means that any of the provided SPIs can be used in the next rekey.

The GM may save these values and if later the GM starts receiving IPsec messages with one of these SPIs without seeing a rekey message for it, this may be used as an indication, that the rekey message was lost on its way to this GM. In this case the GM SHOULD re-register to the group.

Note, that this method of detecting missed rekey messages can only be used by passive GMs, i.e. those, that only listen and don't send data. It's also no point to include this attribute in the GSA_INBAND_REKEY messages, since they use reliable transport. Note also, that the GCKS is free to forget its promises and not to use the SPIs it sent in the TEK_NEXT_SPI attributes before (e.g. in case of GCKS reboot), so the GM must only treat these information as a "best effort" made by GCKS to prepare for future rekeys.

2.4.4. GSA Group Associated Policy

Group specific policy that does not belong to rekey policy (GSA KEK) or traffic encryption policy (GSA TEK) can be distributed to all group member using GSA GAP (Group Associated Policy).

The GSA GAP payload is defined as follows:

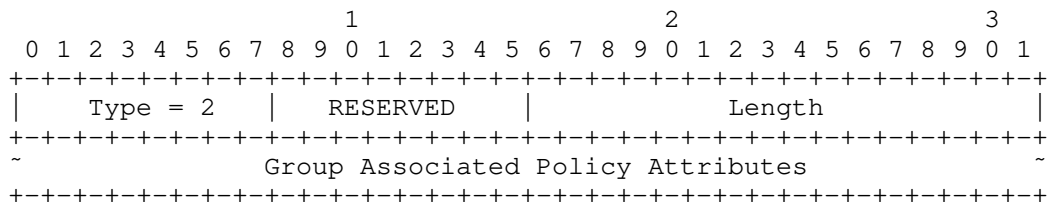


Figure 23: GAP Policy Format

The GSA GAP payload fields are defined as follows:

- o Type = 2 (1 octet) -- Identifies the GSA payload type as GAP in the G-IKEv2 registration or the G-IKEv2 rekey message.
- o RESERVED (1 octet) -- Must be zero.

- o Length (2 octets) -- Length of this structure, including the GSA GAP header and Attributes.
- o Group Associated Policy Attributes (variable) -- Contains attributes following the format defined in Section 3.3.5 of [RFC7296].

Attribute Types are as follows. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

Attribute Type	Value	Type
-----	-----	----
Reserved	0	
ACTIVATION_TIME_DELAY	1	B
DEACTIVATION_TIME_DELAY	2	B
Unassigned	3-16383	
Private Use	16384-32767	

2.4.4.1. ACTIVATION_TIME_DELAY/DEACTIVATION_TIME_DELAY

Section 4.2.1 of [RFC5374] specifies a key rollover method that requires two values be provided to group members. The ACTIVATION_TIME_DELAY attribute allows a GCKS to set the Activation Time Delay (ATD) for SAs generated from TEKs. The ATD defines how long after receiving new SAs that they are to be activated by the GM. The ATD value is in seconds.

The DEACTIVATION_TIME_DELAY allows the GCKS to set the Deactivation Time Delay (DTD) for previously distributed SAs. The DTD defines how long after receiving new SAs it should deactivate SAs that are destroyed by the rekey event. The value is in seconds.

The values of ATD and DTD are independent. However, the DTD value should be larger, which allows new SAs to be activated before older SAs are deactivated. Such a policy ensures that protected group traffic will always flow without interruption.

2.5. Key Download Payload

The Key Download Payload contains the group keys for the group specified in the GSA Payload. These key download payloads can have several security attributes applied to them based upon the security policy of the group as defined by the associated GSA Payload.

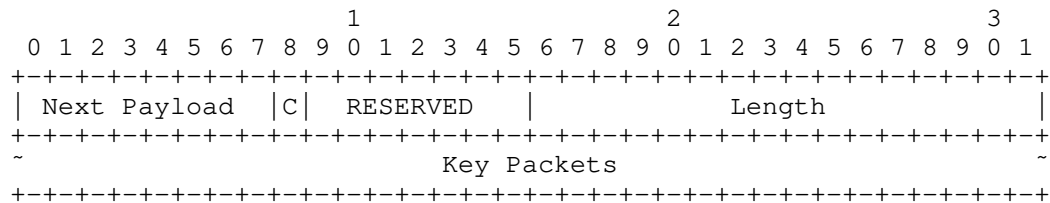


Figure 24: Key Download Payload Format

The Key Download Payload fields are defined as follows:

- o Next Payload (1 octet) -- Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be zero.
- o Critical (1 bit) -- Set according to [RFC7296].
- o RESERVED (7 bits) -- Unused, set to zero.
- o Payload Length (2 octets) -- Length in octets of the current payload, including the generic payload header.
- o Key Packets (variable) -- Contains Key Packets. Several types of key packets are defined. Each Key Packet has the following format.

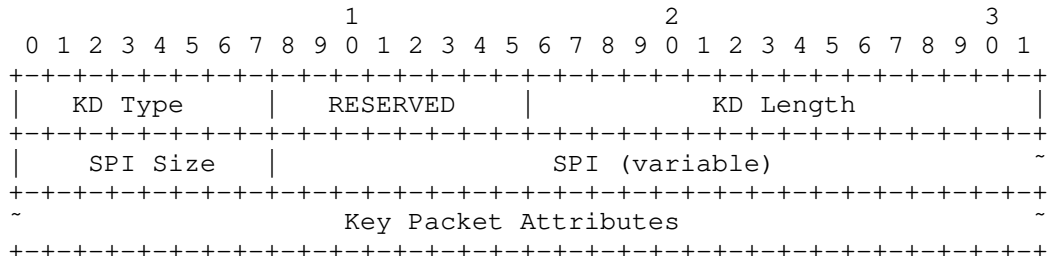


Figure 25: Key Packet Format

- o Key Download (KD) Type (1 octet) -- Identifier for the Key Data field of this Key Packet. In the following table the terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

Key Download Type	Value
-----	-----
Reserved	0
TEK	1
KEK	2
LKH	3
SID	4
Unassigned	5-127
Private Use	128-255

- o RESERVED (1 octet) -- Unused, set to zero.
- o Key Download Length (2 octets) -- Length in octets of the Key Packet data, including the Key Packet header.
- o SPI Size (1 octet) -- Value specifying the length in octets of the SPI as defined by the Protocol-Id.
- o SPI (variable length) -- Security Parameter Index which matches a SPI previously sent in an GSA KEK or GSA TEK Payload.
- o Key Packet Attributes (variable length) -- Contains Key information. The format of this field is specific to the value of the KD Type field. The following sections describe the format of each KD Type.

2.5.1. TEK Download Type

The following attributes may be present in a TEK Download Type. Exactly one attribute matching each type sent in the GSA TEK payload MUST be present. The attributes must follow the format defined in IKEv2 (Section 3.3.5 of [RFC7296]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

TEK KD Attributes	Value	Type	Mandatory
-----	-----	-----	-----
Reserved	0-2		
TEK_KEYMAT	3	V	Y
Unassigned	4-16383		
Private Use	16384-32767		

It is possible that the GCKS will send no TEK key packets in a Registration KD payload (as well as no corresponding GSA TEK payloads in the GSA payload), after which the TEK payloads will be sent in a rekey message.

2.5.1.1. TEK_KEYMAT

The TEK_KEYMAT attribute contains keying material for the corresponding SPI. This keying material will be used with the transform specified in the GSA TEK payload. The keying material is treated equivalent to IKEv2 KEYMAT derived for that IPsec transform.

2.5.2. KEK Download Type

The following attributes may be present in a KEK Download Type. Exactly one attribute matching each type sent in the GSA KEK payload MUST be present. The attributes must follow the format defined in IKEv2 (Section 3.3.5 of [RFC7296]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

KEK KD Attributes -----	Value -----	Type -----	Mandatory -----
Reserved	0		
KEK_ENCR_KEY	1	V	Y
KEK_INTEGRITY_KEY	2	V	N
KEK_AUTH_KEY	3	V	N
Unassigned	4-16383		
Private Use	16384-32767		

If the KEK Key Packet is included, there MUST be only one present in the KD payload.

2.5.2.1. KEK_ENCR_KEY

The KEK_ENCR_KEY attribute type declares that the encryption key for the corresponding SPI is contained in the Key Packet Attribute. The encryption algorithm that will use this key was specified in the GSA KEK payload.

2.5.2.2. KEK_INTEGRITY_KEY

The KEK_INTEGRITY_KEY attribute type declares the integrity key for this SPI is contained in the Key Packet Attribute. The integrity algorithm that will use this key was specified in the GSA KEK payload.

2.5.2.3. KEK_AUTH_KEY

The KEK_AUTH_KEY attribute type declares that the authentication key for this SPI is contained in the Key Packet Attribute. The signature algorithm that will use this key was specified in the GSA KEK payload. An RSA public key format is defined in [RFC3447], Section A.1.1. DSS public key format is defined in [RFC3279] Section 2.3.2. For ECDSA Public keys, use format described in [RFC5480] Section 2.2. Other algorithms added to the IKEv2 Authentication Method registry are also expected to include a format of the public key included in the algorithm specification.

2.5.3. LKH Download Type

The LKH key packet is comprised of attributes representing different leaves in the LKH key tree.

The following attributes are used to pass an LKH KEK array in the KD payload. The attributes must follow the format defined in IKEv2 (Section 3.3.5 of [RFC7296]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

LKH KD Attributes	Value	Type
-----	-----	----
Reserved	0	
LKH_DOWNLOAD_ARRAY	1	V
LKH_UPDATE_ARRAY	2	V
Unassigned	3-16383	
Private Use	16384-32767	

If an LKH key packet is included in the KD payload, there MUST be only one present.

2.5.3.1. LKH_DOWNLOAD_ARRAY

The LKH_DOWNLOAD_ARRAY attribute type is used to download a set of LKH keys to a group member. It MUST NOT be included in a IKEv2 rekey message KD payload if the IKEv2 rekey is sent to more than one group member. If an LKH_DOWNLOAD_ARRAY attribute is included in a KD payload, there MUST be only one present.

This attribute consists of a header block, followed by one or more LKH keys.

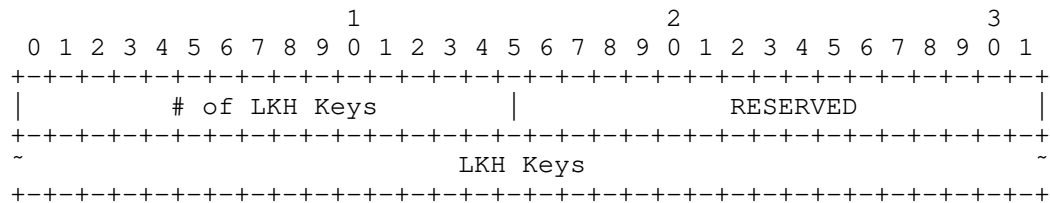


Figure 26: LKH_DOWNLOAD_ARRAY Format

The KEK_LKH attribute fields are defined as follows:

- o Number of LKH Keys (2 octets) -- This value is the number of distinct LKH keys in this sequence.
- o RESERVED (2 octets) -- Unused, set to zero.

Each LKH Key is defined as follows:

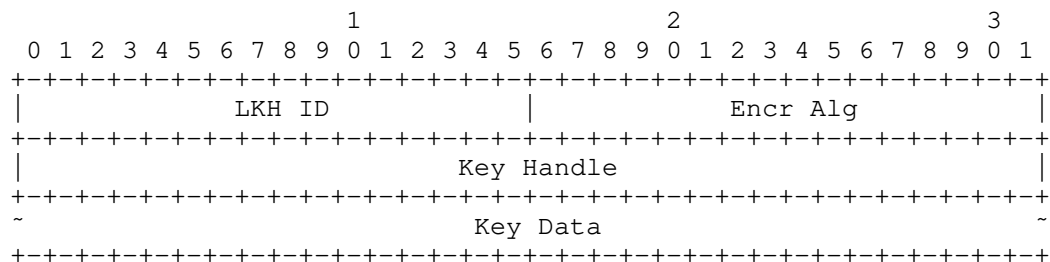


Figure 27: LKH Key Format

- o LKH ID (2 octets) -- This is the position of this key in the binary tree structure used by LKH.
- o Encr Alg (2 octets) -- This is the encryption algorithm for which this key data is to be used. This value is specified in the ENCR transform in the GSA payload.
- o Key Handle (4 octets) -- This is a randomly generated value to uniquely identify a key within an LKH ID.
- o Key Data (variable length) -- This is the actual encryption key data, which is dependent on the Encr Alg algorithm for its format.

The first LKH Key structure in an LKH_DOWNLOAD_ARRAY attribute contains the Leaf identifier and key for the group member. The rest of the LKH Key structures contain keys along the path of the key tree in the order starting from the leaf, culminating in the group KEK.

2.5.3.2. LKH_UPDATE_ARRAY

The LKH_UPDATE_ARRAY attribute type is used to update the LKH keys for a group. It is most likely to be included in a G-IKEv2 rekey message KD payload to rekey the entire group. This attribute consists of a header block, followed by one or more LKH keys, as defined in Section 2.5.3.1.

There may be any number of LKH_UPDATE_ARRAY attributes included in a KD payload.

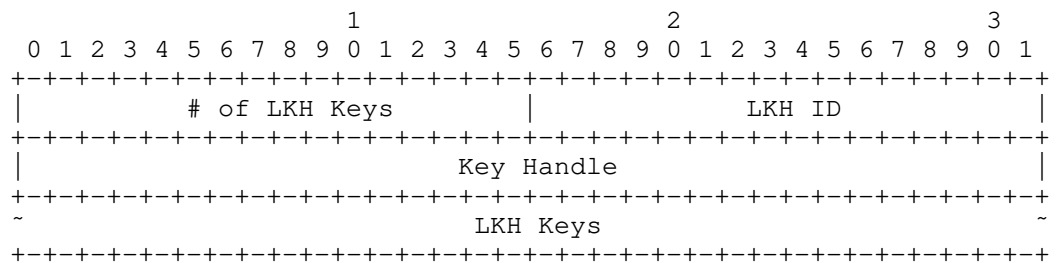


Figure 28: LKH_UPDATE_ARRAY Format

- o Number of LKH Keys (2 octets) -- This value is the number of distinct LKH keys in this sequence.
- o LKH ID (2 octets) -- This is the node identifier associated with the key used to encrypt the first LKH Key.
- o Key Handle (4 octets) -- This is the value that uniquely identifies the key within the LKH ID which was used to encrypt the first LKH key.

The LKH Keys are as defined in Section 2.5.3.1. The LKH Key structures contain keys along the path of the key tree in the order from the LKH ID found in the LKH_UPDATE_ARRAY header, culminating in the group KEK. The Key Data field of each LKH Key is encrypted with the LKH key preceding it in the LKH_UPDATE_ARRAY attribute. The first LKH Key is encrypted under the key defined by the LKH ID and Key Handle found in the LKH_UPDATE_ARRAY header.

2.5.4. SID Download Type

The SID attribute is used to download one or more Sender-ID (SID) values for the exclusive use of a group member. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

SID KD Attributes	Value	Type
-----	-----	-----
Reserved	0	
NUMBER_OF_SID_BITS	1	B
SID_VALUE	2	V
Unassigned	3-16383	
Private Use	16384-32767	

Because a SID value is intended for a single group member, the SID Download type MUST NOT be distributed in a GSA_REKEY message distributed to multiple group members.

2.5.4.1. NUMBER_OF_SID_BITS

The NUMBER_OF_SID_BITS attribute type declares how many bits of the cipher nonce in which to represent an SID value. The bits are applied as the most significant bits of the IV, as shown in Figure 1 of [RFC6054] and specified in Section 1.4.6.2. Guidance for a GCKS choosing the NUMBER_OF_SID_BITS is provided in Section 3 of [RFC6054].

This value is applied to each SID value distributed in the SID Download.

2.5.4.2. SID_VALUE

The SID_VALUE attribute type declares a single SID value for the exclusive use of this group member. Multiple SID_VALUE attributes MAY be included in a SID Download.

2.5.4.3. GM Semantics

The SID_VALUE attribute value distributed to the group member MUST be used by that group member as the SID field portion of the IV for all Data-Security SAs including a counter-based mode of operation distributed by the GCKS as a part of this group. When the Sender-Specific IV (SSIV) field for any Data-Security SA is exhausted, the group member MUST NOT act as a sender on that SA using its active SID. The group member SHOULD re-register, at which time the GCKS will issue a new SID to the group member, along with either the same Data-Security SAs or replacement ones. The new SID replaces the existing SID used by this group member, and also resets the SSIV value to its starting value. A group member MAY re-register prior to the actual exhaustion of the SSIV field to avoid dropping data packets due to the exhaustion of available SSIV values combined with a particular SID value.

A group member MUST ignore an SID Download Type KD payload present in a GSA-REKEY message, otherwise more than one GM may end up using the same SID.

2.5.4.4. GCKS Semantics

If any KD payload includes keying material that is associated with a counter-mode of operation, an SID Download Type KD payload containing at least one SID_VALUE attribute MUST be included. The GCKS MUST NOT send the SID Download Type KD payload as part of a GSA_REKEY message, because distributing the same sender-specific policy to more than one group member will reduce the security of the group.

2.6. Delete Payload

There are occasions when the GCKS may want to signal to group members to delete policy at the end of a broadcast, if group policy has changed, or the GCKS needs to reset the policy and keying material for the group due to an emergency. Deletion of keys MAY be accomplished by sending an IKEv2 Delete Payload, section 3.11 of [RFC7296] as part of a registration or rekey Exchange. Whenever an SA is to be deleted, the GCKS SHOULD send the Delete Payload in both registration and rekey exchanges, because GMs with previous group policy may contact the GCKS using either exchange.

The Protocol ID MUST be 41 for GSA_REKEY Exchange, 2 for AH or 3 for ESP. Note that only one protocol id value can be defined in a Delete payload. If a TEK and a KEK SA for GSA_REKEY Exchange must be deleted, they must be sent in different Delete payloads. Similarly, if a TEK specifying ESP and a TEK specifying AH need to be deleted, they must be sent in different Delete payloads.

There may be circumstances where the GCKS may want to reset the policy and keying material for the group. The GCKS can signal deletion of all policy of a particular TEK by sending a TEK with a SPI value equal to zero in the delete payload. In the event that the administrator is no longer confident in the integrity of the group they may wish to remove all KEK and all the TEKs in the group. This is done by having the GCKS send a delete payload with a SPI of zero and a Protocol-ID of AH or ESP to delete all TEKs, followed by another delete payload with a SPI value of zero and Protocol-ID of KEK SA to delete the KEK SA.

2.7. Notify Payload

G-IKEv2 uses the same Notify payload as specified in [RFC7296], section 3.10.

There are additional Notify Message types introduced by G-IKEv2 to communicate error conditions and status.

NOTIFY messages - error types	Value
-----	-----
INVALID_GROUP_ID -	45
AUTHORIZATION_FAILED -	46
REGISTRATION_FAILED -	TBD

INVALID_GROUP_ID indicates the group id sent during the registration process is invalid.

AUTHORIZATION_FAILED is sent in the response to a GSA_AUTH message when authorization failed.

REGISTRATION_FAILED is sent by the GCKS when the GM registration request cannot be satisfied.

NOTIFY messages - status types	Value
-----	-----
SENDER -	16429
REKEY_IS_NEEDED -	TBD

SENDER notification is sent in GSA_AUTH or GSA_REGISTRATION to indicate that the GM intends to be sender of data traffic. The data includes a count of how many SID values the GM desires. The count MUST be 4 octets long and contain the big endian representation of the number of requested SIDs.

REKEY_IS_NEEDED is sent in GSA_AUTH response message to indicate that the GM must perform an immediate rekey of IKE SA to make it secure against quantum computers and then start a registration request over.

2.8. Authentication Payload

G-IKEv2 uses the same Authentication payload as specified in [RFC7296], section 3.8, to sign the rekey message.

3. Security Considerations

3.1. GSA Registration and Secure Channel

G-IKEv2 registration exchange uses IKEv2 IKE_SA_INIT protocols, inheriting all the security considerations documented in [RFC7296] section 5 Security Considerations, including authentication, confidentiality, protection against man-in-the-middle, protection against replay/reflection attacks, and denial of service protection. The GSA_AUTH and GSA_REGISTRATION exchanges also take advantage of

those protections. In addition, G-IKEv2 brings in the capability to authorize a particular group member regardless of whether they have the IKEv2 credentials.

3.2. GSA Maintenance Channel

The GSA maintenance channel is cryptographically and integrity protected using the cryptographic algorithm and key negotiated in the GSA member registration exchanged.

3.2.1. Authentication/Authorization

Authentication is implicit, the public key of the identity is distributed during the registration, and the receiver of the rekey message uses that public key and identity to verify the message came from the authorized GCKS.

3.2.2. Confidentiality

Confidentiality is provided by distributing a confidentiality key as part of the GSA member registration exchange.

3.2.3. Man-in-the-Middle Attack Protection

GSA maintenance channel is integrity protected by using a digital signature.

3.2.4. Replay/Reflection Attack Protection

The GSA_REKEY message includes a monotonically increasing sequence number to protect against replay and reflection attacks. A group member will recognize a replayed message by comparing the Message ID number to that of the last received rekey message, any rekey message containing a Message ID number less than or equal to the last received value MUST be discarded. Implementations should keep a record of recently received GSA rekey messages for this comparison.

4. IANA Considerations

4.1. New Registries

A new set of registries should be created for G-IKEv2, on a new page titled Group Key Management using IKEv2 (G-IKEv2) Parameters. The following registries should be placed on that page. The terms Reserved, Expert Review and Private Use are to be applied as defined in [RFC8126].

GSA Policy Type Registry, see Section 2.4.1

KEK Attributes Registry, see Section 2.4.2.1

KEK Management Algorithm Registry, see Section 2.4.2.1.1

GSA TEK Payload Protocol ID Type Registry, see Section 2.4.3

TEK Attributes Registry, see Section 2.4.3

Key Download Type Registry, see Section 2.5

TEK Download Type Attributes Registry, see Section 2.5.1

KEK Download Type Attributes Registry, see Section 2.5.2

LKH Download Type Attributes Registry, see Section 2.5.3

SID Download Type Attributes Registry, see Section 2.5.4

4.2. New Payload and Exchange Types Added to the Existing IKEv2 Registry

The following new payloads and exchange types specified in this memo have already been allocated by IANA and require no further action, other than replacing the draft name with an RFC number.

The present document describes new IKEv2 Next Payload types, see Section 2.1

The present document describes new IKEv2 Exchanges types, see Section 2.1

The present document describes new IKEv2 notification types, see Section 2.7

4.3. Changes to Previous Allocations

Section 4.7 indicates an allocation in the IKEv2 Notify Message Types - Status Types registry has been made. This NOTIFY type was allocated earlier in the development of G-IKEv2. The number is 16429, and was allocated with the name SENDER_REQUEST_ID. The name should be changed to SENDER.

5. Acknowledgements

The authors thank Lakshminath Dondeti and Jing Xiang for first exploring the use of IKEv2 for group key management and providing the basis behind the protocol. Mike Sullenberger and Amjad Inamdar were

instrumental in helping resolve many issues in several versions of the document.

6. Contributors

The following individuals made substantial contributions to early versions of this memo.

Sheela Rowles
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-527-7677
Email: sheela@cisco.com

Aldous Yeung
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-853-2032
Email: cyyeung@cisco.com

Paulina Tran
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-526-8902
Email: ptran@cisco.com

Yoav Nir
Dell EMC
9 Andrei Sakharov St
Haifa 3190500
Israel

Email: ynir.ietf@gmail.com

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>>.
- [RFC2627] Wallner, D., Harder, E., and R. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, DOI 10.17487/RFC2627, June 1999, <<https://www.rfc-editor.org/info/rfc2627>>.
- [RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, DOI 10.17487/RFC3740, March 2004, <<https://www.rfc-editor.org/info/rfc3740>>.
- [RFC4046] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, DOI 10.17487/RFC4046, April 2005, <<https://www.rfc-editor.org/info/rfc4046>>.
- [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>>.
- [RFC6054] McGrew, D. and B. Weis, "Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic", RFC 6054, DOI 10.17487/RFC6054, November 2010, <<https://www.rfc-editor.org/info/rfc6054>>.
- [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>>.
- [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>>.
- [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

- [I-D.ietf-ipsecme-qr-ikev2]
Fluhrer, S., McGrew, D., Kampanakis, P., and V. Smyslov,
"Postquantum Preshared Keys for IKEv2", draft-ietf-
ipsecme-qr-ikev2-08 (work in progress), March 2019.
- [I-D.tjhai-ipsecme-hybrid-qske-ikev2]
Tjhai, C., Tomlinson, M., grbartle@cisco.com, g., Fluhrer,
S., Geest, D., Garcia-Morchon, O., and V. Smyslov,
"Framework to Integrate Post-quantum Key Exchanges into
Internet Key Exchange Protocol Version 2 (IKEv2)", draft-
tjhai-ipsecme-hybrid-qske-ikev2-03 (work in progress),
January 2019.
- [IKEV2-IANA]
IANA, "Internet Key Exchange Version 2 (IKEv2)
Parameters", February 2016,
<[http://www.iana.org/assignments/ikev2-parameters/
ikev2-parameters.xhtml#ikev2-parameters-7](http://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-7)>.
- [NNL]
Naor, D., Noal, M., and J. Lotspiech, "Revocation and
Tracing Schemes for Stateless Receivers", Advances in
Cryptography, Crypto '01, Springer-Verlag LNCS 2139, 2001,
pp. 41-62, 2001,
<<http://www.wisdom.weizmann.ac.il/~naor/>>.
- [OFT]
McGrew, D. and A. Sherman, "Key Establishment in Large
Dynamic Groups Using One-Way Function Trees", Manuscript,
submitted to IEEE Transactions on Software Engineering,
1998, <[http://download.nai.com/products/media/nai/misc/
oft052098.ps](http://download.nai.com/products/media/nai/misc/oft052098.ps)>.
- [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>>.
- [RFC3279]
Bassham, L., Polk, W., and R. Housley, "Algorithms and
Identifiers for the Internet X.509 Public Key
Infrastructure Certificate and Certificate Revocation List
(CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April
2002, <<https://www.rfc-editor.org/info/rfc3279>>.
- [RFC3447]
Jonsson, J. and B. Kaliski, "Public-Key Cryptography
Standards (PKCS) #1: RSA Cryptography Specifications
Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February
2003, <<https://www.rfc-editor.org/info/rfc3447>>.

- [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>>.
- [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>>.
- [RFC4543] McGrew, D. and J. Viega, "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, DOI 10.17487/RFC4543, May 2006, <<https://www.rfc-editor.org/info/rfc4543>>.
- [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>>.
- [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <<https://www.rfc-editor.org/info/rfc5480>>.
- [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>>.
- [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>>.
- [RFC6467] Kivinen, T., "Secure Password Framework for Internet Key Exchange Version 2 (IKEv2)", RFC 6467, DOI 10.17487/RFC6467, December 2011, <<https://www.rfc-editor.org/info/rfc6467>>.
- [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>>.

- [RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)", RFC 7427, DOI 10.17487/RFC7427, January 2015, <<https://www.rfc-editor.org/info/rfc7427>>.
- [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>>.

Appendix A. Use of LKH in G-IKEv2

Section 5.4 of [RFC2627] describes the LKH architecture, and how a GCKS uses LKH to exclude group members. This section clarifies how the LKH architecture is used with G-IKEv2.

A.1. Group Creation

When a GCKS forms a group, it creates a key tree as shown in the figure below. The key tree contains logical keys (represented as numbers in the figure) and a private key shared with only a single GM (represented as letters in the figure). Note that the use of numbers and letters is used for explanatory purposes; in fact, each key would have an LKH ID, which is two-octet identifier chosen by the GCKS. The GCKS may create a complete tree as shown, or a partial tree which is created on demand as members join the group. The top of the key tree (i.e., "1" in Figure 29) is used as the KEK for the group.

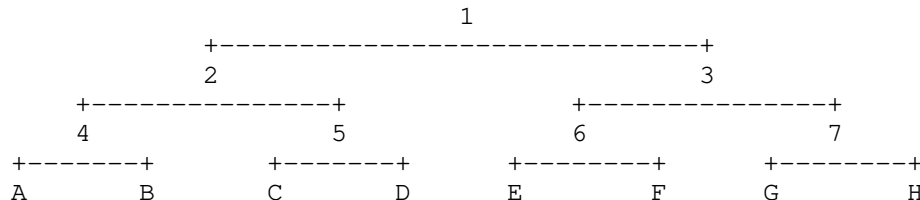


Figure 29: Initial LKH tree

When GM "A" joins the group, the GCKS provides an LKH_DOWNLOAD_ARRAY in the KD payload of the GSA_AUTH or GSA_REGISTRATION exchange. Given the tree shown in figure above, the LKH_DOWNLOAD_ARRAY will contain four LKH Key payloads, each containing an LKH ID and Key Data. If the LKH ID values were chosen as shown in the figure, four LKH Keys would be provided to GM "A", in the following order: A, 4, 2, 1. When GM "B" joins the group, it would also be given four LKH Keys in the following order: B, 4, 2, 1. And so on, until GM "H" joins the group and is given H, 7, 3, 1.

A.2. Group Member Exclusion

If the GKCS has reason to believe that a GM should be excluded, then it can do so by sending a GSA_REKEY exchange that includes a set of LKH_UPDATE_ARRAY attributes in the KD payload. Each LKH_UPDATE_ARRAY contains a set of LKH Key payloads, in which every GM other than the excluded GM will be able to determine a set of new logical keys, which culminate in a new key "1". The excluded GM will observe the set of LKH_UPDATE_ARRAY attributes, but cannot determine the new logical keys because each of the "Key Data" fields is encrypted with a key held by other GMs. The GM will hold no keys to properly decrypt any of the "Key Data" fields, including key "1" (i.e., the new KEK). When a subsequent GSA_REKEY exchange is delivered by the GKCS and protected by the new KEK, the excluded GM will no longer be able to see the contents of the GSA_REKEY, including new TEKs that will be delivered to replace existing TEKs. At this point, the GM will no longer be able to participate in the group.

In the example below, new keys are represented as the number followed by a "prime" symbol (e.g., "1" becomes "1'"). Each key is encrypted by another key. This is represented as "{key1}key2", where key2 encrypts key1. For example, "{1'}2'" states that a new key "1'" is encrypted with a new key "2'".

If GM "B" is to be excluded, the GKCS will need to include three LKH_UPDATE_ARRAY attributes in the GSA_REKEY message. The order of the attributes does not matter; only the order of the keys within each attribute.

- o One will provide GM "A" with new logical keys that are shared with B: {4'}A, {2'}4', {1'}2'
- o One will provide all GMs holding key "5" with new logical keys: {2'}5, {1'}2'
- o One will provide all GMs holding key "3" with a new KEK: {1'}3

Each GM will look at each LKH_UPDATE_ARRAY attribute and observe an LKH ID which is present in an LKH Key delivered to them in the LKH_DOWNLOAD_ARRAY they were given. If they find a matching LKH ID, then they will decrypt the new key with the logical key immediately preceding that LKH Key, and so on until they have received the new 1' key.

The resulting key tree from this rekey event would be shown in Figure 30.

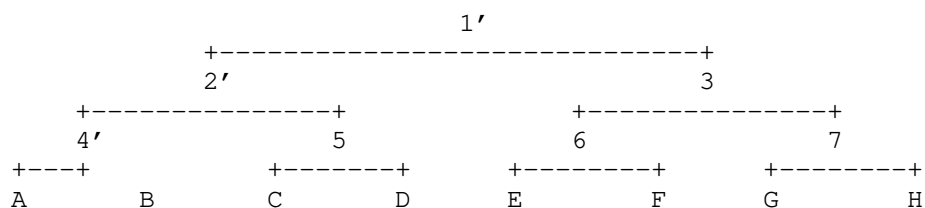


Figure 30: LKH tree after B has been excluded

Authors' Addresses

Brian Weis
Independent
USA

Email: bew.stds@gmail.com

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

Phone: +7 495 276 0211

Email: svan@elvis.ru