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

Copyright Notice

Table of Contents

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
   -----------                               -----------
   HDR, SAi1, KEi, Ni,
   [N(INTERMEDIATE_EXCHANGE_SUPPORTED)] -->
                                  <-- HDR, SAr1, KEr, Nr, [CERTREQ],
                                [N(INTERMEDIATE_EXCHANGE_SUPPORTED)]
```

The INTERMEDIATE_EXCHANGE_SUPPORTED is a Status Type IKEv2 notification. Its Notify Message Type is 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.

```
                        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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ^ ^
   |                       IKE SA Initiator’s SPI                  | | |
   |                                                               | | |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ I |
   |                       IKE SA Responder’s SPI                  | K |
   |                                                               | E |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
   |  Next Payload | MjVer | MnVer | Exchange Type |     Flags     | H |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ d |
   |                          Message ID                           | r A
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | |
   |                       Adjusted Length                         | | |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ v |
   |                                                               |   |
   ~                 Unencrypted payloads (if any)                 ~   |
   |                                                               |   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ^ |
   | Next Payload  |C|  RESERVED   |    Adjusted Payload Length    | | |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | v
   |                                                               | |
   ~                     Initialization Vector                     ~ E
   |                                                               | E
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ c ^
   |                                                               | r |
   ~             Inner payloads (not yet encrypted)                ~   P
   |                                                               | P |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ l v
   |              Padding (0-255 octets)           |  Pad Length   | d
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
   |                                                               | |
   ~                    Integrity Checksum Data                    ~ |
   |                                                               | |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ v
```

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., Dukkipati, 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) {...}  -->

         <Calculate IntAuth_i1 = prf(SK_pi1, ...)>

                             <--  HDR(IKE_INTERMEDIATE,MID=1),
                                  SK(SK_er1,SK_ar1) {...}

         <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) {...}  -->

         <Calculate IntAuth_i2 = prf(SK_pi2, ...)>

                             <--  HDR(IKE_INTERMEDIATE,MID=2),
                                  SK(SK_er2,SK_ar2) {...}

         <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