ipsecme
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

D. Schinazi
Apple Inc.

March 5, 2018

Privacy Addition to the Internet Key Exchange Protocol Version 2 (IKEv2) IKE_SA_INIT Exchange
draft-dschinazi-ipsecme-sa-init-privacy-addition-00

Abstract

The Internet Key Exchange Protocol version 2 (IKEv2) provides strong security and privacy properties to both endpoints once they have authenticated each other. However, before an endpoint has validated the peer's AUTH payload, it could be divulging information to an untrusted host. An example of such information is the Identification payload of the initiator. Another example is the fact that a host is running an IKEv2 responder. This document introduces a new "Initialization Authentication Code" notify payload that can be included in IKE_SA_INIT messages to increase their trustworthiness. This new protection is meant to be used in addition to current IKEv2 mechanisms and is not meant to replace the AUTH payload in any way.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Table of Contents

1. Introduction

The Internet Key Exchange Protocol version 2 (IKEv2) [RFC7296] provides strong security and privacy properties to both endpoints once they have authenticated each other. However, before an endpoint has validated the peer's AUTH payload, it could be divulging information to an untrusted host. Examples include:

- The Identification payload of the initiator is sent with the initiator's first IKE_AUTH request. This payload can be used to track the owner of the device initiating IKE.

- Some IKEv2 servers may wish to hide their very existence to avoid being blacklisted by entities that resent the privacy properties an IKEv2/IPsec tunnel can provide to users. If the IKEv2 server is accessible over TLS on a TCP port [RFC8229] that is shared with another protocol, responding to the initiator's IKE_SA_INIT can disclose the server's existence.

This document introduces a new "Initialization Authentication Code" (IAC) notify payload that can be included in IKE_SA_INIT messages to increase their trustworthiness. This new protection is meant to be used in addition to current IKEv2 mechanisms and is not meant to replace the AUTH payload in any way.

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 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Terminology

This document uses the following terms:

Endpoint One of the two hosts that are involved in an IKE exchange.

Initiator The endpoint that sends the first IKE_SA_INIT request of the IKE exchange being discussed.

Responder The endpoint that is not the initiator.

Peer When discussing an endpoint, its peer is the other endpoint participating in the IKE exchange.

IASS Initialization Authentication Shared Secret, a shared secret included in the IKEv2 configuration. It is separate from any shared secret used for computation of the AUTH payload.

MAC Message Authentication Code, a cryptographic means of ensuring integrity and authenticity of a message.

IAC Initialization Authentication Code, a MAC of IKE_SA_INIT nonces with the IASS.

PRF Pseudo-Random Function, a function used to compute the IAC.

2. Attack Vectors

This document only attempts to address the following attack vectors.

2.1. On-Path Attacker Targeting Initiator

This attack vector assumes the presence of an active on-path attacker that can block and forge any packets between both endpoints. Without the mechanism described in this document, the attacker can forge an IKE_SA_INIT reply and get the initiator to send it its IKE_AUTH request encrypted with the ephemeral shared secret computed between the initiator and the attacker. This leaks the identity of the initiator (IDi) and can leak the identity of the responder (IDr) if the initiator also sent it.

2.2. Off-Path Attacker Targeting Responder

Some network middleboxes may wish block to block IKEv2 negotiation. This is often done by blocking UDP traffic which can be worked around using IKEv2 TCP encapsulation [RFC8229]. This obfuscation can even be improved by encapsulating IKEv2 and IPsec inside TLS. However, a more persevering middlebox can establish a TLS connection to the responder and try to send an IKE_SA_INIT to probe the server for IKEv2 support. Without the mechanism described in this document, the responder has to send an IKE_SA_INIT reply before it’s established any initiator identity, leaking the presence of the IKEv2 server.

3. Initialization Authentication

3.1. Computing Initialization Authentication

Each endpoint configuration will include both an IASS and a PRF for this endpoint, and also IASS and PRF of the peer. It will commonly be the case (but it is not required) that the same IASS and the same PRF is used in both directions.

The peers authenticate the IKE_SA_INIT messages by having each MAC nonces using a padded shared secret as the key. The IAC is computed as follows:

IAC_i = prf_i(prf_i(IASS_i, "Initialization Authentication Key Pad for IKEv2 Initiator"), Ni)

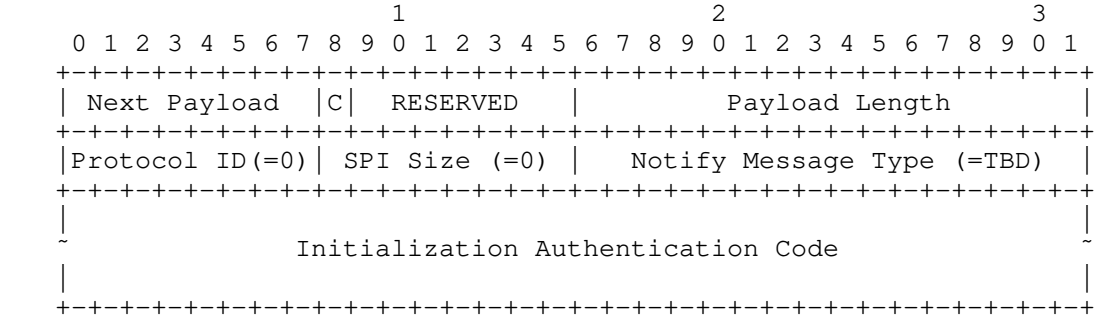
IAC_r = prf_r(prf_r(IASS_r, "Initialization Authentication Key Pad for IKEv2 Responder"), Ni | Nr)

Where IAC_i and IAC_r are the Initialization Authentication Codes of the initiator and responder respectively. Ni and Nr are the nonces sent in the IKE_SA_INIT messages that contain the IAC. The strings are 57 ASCII characters without null termination. prf_i() and prf_r() denote the PRFs selected in the initiator and responder configurations respectively. IASS_i and IASS_r denote the initialization authentication shared secret in the initiator and responder configurations respectively.

The pad strings are added so that if the IASS are derived from a password, the IKE implementation need not store the password in cleartext, which could not be used as a password equivalent for protocols other than IKEv2. Using different pad strings for each direction limits the information leakage about the IASS if IASS_i and IASS_r are equal. IAC_r is based on both Ni and Nr to prevent replay attacks on the IKE_SA_INIT reply while also preventing a MAC oracle on the responder, since the responder controls the random generation of Nr.

3.2. Initialization Authentication Notify Payload

The Initialization Authentication Notify Payload is defined as follows:



The 'Next Payload', 'C', 'RESERVED', 'Payload Length', 'Protocol ID', 'SPI Size', and 'Notify Message Type' fields are the same as described in Section 3 of [RFC7296]. The Critical ('C') bit MUST be set to 0. The 'SPI Size' field MUST be set to 0 to indicate that the SPI is not present in this message. The 'Protocol ID' MUST be set to 0, since the notification is specific to this IKE_SA_INIT message. The 'Payload Length' field is set to the length in octets of the entire payload, including the generic payload header. The 'Notify Message Type' field is set to indicate INITIALIZATION_AUTHENTICATION_CODE (TBD). The Initialization Authentication Code field has a variable length, and is computed according to Section 3.1.

3.3. Receiving Initialization Authentication

When the responder receives the initiator's IKE_SA_INIT request, it has not yet established the identity of the initiator, as the identity payload will come later. If the responder has distributed the same initialization authentication shared secret for all of its clients, it can easily verify that incoming IKE_SA_INIT requests come from clients that possess the shared secret. If the responder uses different initialization authentication shared secrets per client, it will have to iterate all of them to find a match since there is no identity sent with the IKE_SA_INIT request. Care should be taken with regards to the timing of the IKE_SA_INIT reply to avoid leaking information. If the responder cannot find a (IASS, PRF) combination in its configuration that matches the IAC in the incoming IKE_SA_INIT request, it MUST silently ignore the incoming packet. Not responding at all is crucial to hiding the fact that the responder is running an IKEv2 server. The responder SHOULD log the failure to facilitate debugging.

When the initiator receives the responder's IKE_SA_INIT reply, it knows the identity of the responder it is trying to establish a security association with. It can therefore use the (IASS, PRF) from its configuration to validate the IAC on the reply. If the IAC in the reply does not match what was computed from the configuration, the initiator treats this similarly to receiving and error on the reply and MUST fail the exchange and MUST NOT send the IKE_AUTH message it would have normally sent. This is crucial to protect the initiator identity (IDi) from an active on-path attacker. The initiator SHOULD log the failure to facilitate debugging.

4. Security Considerations

This document attempts to resolve the attacks described in Section 2 and no other attacks on IKEv2.

4.1. Timing Attacks

An IKEv2 responder wishing to stay hidden needs to ensure it doesn't leak information via the timing of its responses. In general if it receives an IKE_SA_INIT message whose IAC does not match, it simply does not respond. However if IKEv2 is running over TCP, the timing of when the responder closes the TCP connection can leak information. Implementors of hidden IKEv2 responders should ensure that they reply to bad input and to invalid IAC in similar time. In particular, if the server is also running another application protocol on the same port, it SHOULD reply to an invalid or missing IAC the same way as it would reply to an invalid request on that other protocol.

4.2. Replay Attacks

The initiator's IKE_SA_INIT message is sent unencrypted and can be replayed. The mechanism described in this document is still vulnerable to replays of the IKE_SA_INIT message. Note however that an obfuscated IKEv2 server running over TLS can leverage TLS to ensure the absence of on-path attackers inside the TLS channel between both endpoints.

The responder's IKE_SA_INIT message is also sent unencrypted and can also be replayed. However, the Initialization Authentication Code takes Ni as input so replaying a previous responder IKE_SA_INIT for a different IKEv2 exchange will have a different IAC and will be ignored.

4.3. Denial of Service Attacks

An IKEv2 responder implementing this specification opens themselves to computing more MACs for IKE_SA_INIT messages. We believe that downside is negligible compared to other DOS attacks on IKEv2.

5. IANA Considerations

If approved, this document defines a new payload in the IANA "IKEv2 Notify Message Types - Status Types" registry [IKEV2IANA]:

NOTIFY messages: status types	Value
INITIALIZATION_AUTHENTICATION_CODE	TBD

6. Normative References

[IKEV2IANA] "IANA, Internet Key Exchange Version 2 (IKEv2) Parameters", <https://www.iana.org/assignments/ikev2-parameters/>.

[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>.

[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>.

Author's Address

David Schinazi
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
US

Email: dschinazi@apple.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.

Table of Contents

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

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

draft-ietf-ipsecme-qr-ikev2-10

- Addresses issues raised during IETF LC.

draft-ietf-ipsecme-qr-ikev2-09

- Addresses issues raised in AD review.

draft-ietf-ipsecme-qr-ikev2-08

- Editorial changes.

draft-ietf-ipsecme-qr-ikev2-07

- Editorial changes.

draft-ietf-ipsecme-qr-ikev2-06

- Editorial changes.

draft-ietf-ipsecme-qr-ikev2-05

- Addressed comments received during WGLC.

draft-ietf-ipsecme-qr-ikev2-04

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

draft-ietf-ipsecme-qr-ikev2-03

- Editorial changes and minor text nit fixes.
- Integrated Tommy P. text suggestions.

draft-ietf-ipsecme-qr-ikev2-02

- Added note that the PPK is stirred in the initial IKE SA setup only.
- Added note about the initiator ignoring any content in the PPK_IDENTITY notification from the responder.
- fixed Tero's suggestions from 2/6/1028
- Added IANA assigned message types where necessary.
- fixed minor text nits

draft-ietf-ipsecme-qr-ikev2-01

- Nits and minor fixes.
- prf is replaced with prf+ for the SK_d and SK_pi/r calculations.
- Clarified using PPK in case of EAP authentication.
- PPK_SUPPORT notification is changed to USE_PPK to better reflect its purpose.

draft-ietf-ipsecme-qr-ikev2-00

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

draft-fluhrer-qr-ikev2-05

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

draft-fluhrer-qr-ikev2-03

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

draft-fluhrer-qr-ikev2-02

- 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.
- Removed positive endorsements of various algorithms. Retained warnings about algorithms known to be weak against a quantum computer.

draft-fluhrer-qr-ikev2-01

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

draft-fluhrer-qr-ikev2-00

- We switched from using vendor ID's to transmit the additional data to notifications.

- We added a mandatory cookie exchange to allow the server to communicate to the client before the initial exchange.

- We added algorithm agility by having the server tell the client what algorithm to use in the cookie exchange.

- 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.

- 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:

```
Initiator                          Responder
-------------------------------------------------------------------
HDR, SAi1, KEi, Ni, N(USE_PPK)  --->
```

N(USE_PPK) is a status notification payload with the type 16435; it has a protocol ID of 0, no SPI and no notification data associated with it.

If the initiator needs to resend this initial message with a COOKIE notification, then the resend would include the USE_PPK notification if the original message did (see Section 2.6 of [RFC7296]).

If the responder does not support this specification or does not have any PPK configured, then it ignores the received notification (as defined in [RFC7296] for unknown status notifications) and continues with the IKEv2 protocol as normal. Otherwise the responder replies with the IKE_SA_INIT message including a USE_PPK notification in the response:

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

When the initiator receives this reply, it checks whether the responder included the USE_PPK notification. If the responder did not and the flag mandatory_or_not indicates that using PPKs is mandatory for communication with this responder, then the initiator MUST abort the exchange. This situation may happen in case of misconfiguration, when the initiator believes it has a mandatory-to-use PPK for the responder, while the responder either doesn't support PPKs at all or doesn't have any PPK configured for the initiator. See Section 6 for discussion of the possible impacts of this situation.

If the responder did not include the USE_PPK notification and using a PPK for this particular responder is optional, then the initiator continues with the IKEv2 protocol as normal, without using PPKs.

If the responder did include the USE_PPK notification, then the initiator selects a PPK, along with its identifier PPK_ID. Then, it computes this modification of the standard IKEv2 key derivation from Section 2.14 of [RFC7296]:

```
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 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:

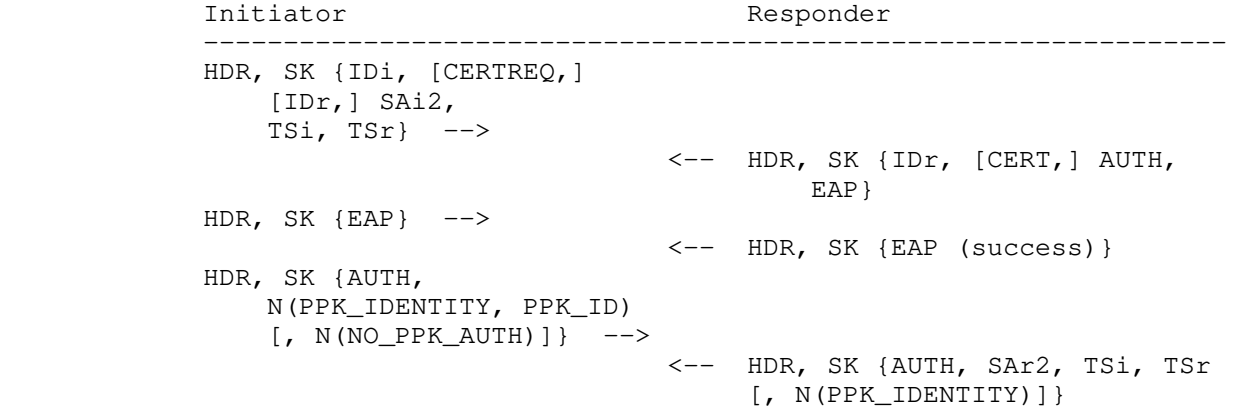
```
Initiator                       Responder
-----------------------------------------------------------------
                          <--   HDR, SK {IDr, [CERT,]
                                AUTH, SAr2,
                                TSi, TSr, N(PPK_IDENTITY)}
```

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.



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:

- The set of PPKs (and corresponding PPK_IDs) that this node would need to know.

- For each peer that this node would initiate to, which PPK will be used.

- 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:

- 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.

- 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 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

- Any IKEv2 Encryption algorithm, PRF or Integrity algorithm with key size less than 256 bits.

- Any ESP Transform with key size less than 256 bits.

- 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 Fluhrer
Cisco Systems

Email: sfluhrer@cisco.com

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

David McGrew
Cisco Systems

Email: mcgrew@cisco.com

Valery Smyslov
ELVIS-PLUS

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 6, 2019

V. Smyslov
ELVIS-PLUS

December 3, 2018

Intermediate Exchange in the IKEv2 Protocol
draft-smyslov-ipsecme-ikev2-aux-02

Abstract

This documents defines a new exchange, called Intermediate Exchange, for the Internet Key Exchange protocol Version 2 (IKEv2). This exchange can be used for transferring large amount of data in the process of IKEv2 Security Association (SA) establishment. Introducing Intermediate Exchange allows re-using existing IKE Fragmentation mechanism, that helps to avoid IP fragmentation of large IKE messages, but cannot be used in the initial IKEv2 exchange.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 6, 2019.

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 size of the messages is large enough, IP fragmentation takes place, that may interfere badly with some network devices. The problem is described in more detail in [RFC7383], which also defines an extension to the IKEv2 called IKE Fragmentation. This extension allows IKE messages to be fragmented at IKE level, eliminating possible issues caused by IP fragmentation. However, the IKE Fragmentation cannot be used in the initial IKEv2 exchange, IKE_SA_INIT. This limitation in most cases is not a problem, since the IKE_SA_INIT messages used to be small enough not to cause IP fragmentation.

Recent progress in Quantum Computing has brought a concern that classical Diffie-Hellman key exchange methods will become insecure in a relatively near future and should be replaced with Quantum Computer (QC) resistant ones. Currently most of QC-resistant key exchange methods have large public keys. If these keys are exchanged in the IKE_SA_INIT, then most probably IP fragmentation will take place, therefore all the problems caused by it will become inevitable.

A possible solution to the problem would be to use TCP as a transport for IKEv2, as defined in [RFC8229]. However this approach has

significant drawbacks and is intended to be a "last resort" when UDP transport is completely blocked by intermediate network devices.

This document defines a new exchange for the IKEv2 protocol, called Intermediate Exchange or INTERMEDIATE. One or more these exchanges may take place right after the IKE_SA_INIT exchange and prior to the IKE_AUTH exchange. The INTERMEDIATE exchange messages can be fragmented using IKE Fragmentation mechanism, so these exchanges may be used to transfer large amounts of data which don't fit into the IKE_SA_INIT exchange without causing IP fragmentation.

While ability to transfer large public keys of QC-resistant key exchange methods is a primary motivation for introducing of the Intermediate Exchange, its application is not limited to this use case. This exchange may be used whenever some data need to be transferred before the IKE_AUTH exchange and for some reason the IKE_SA_INIT exchange is not suited for this purpose. This document defines the INTERMEDIATE exchange without tying it to any specific use case. It is expected that separate specifications will define for which purposes and how the INTERMEDIATE exchange is used in the IKEv2.

2. Terminology and Notation

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

3. Intermediate Exchange Details

3.1. Support for Intermediate Exchange Negotiation

The initiator indicates its support for Intermediate Exchange by including a notification of type INTERMEDIATE_EXCHANGE_SUPPORTED in the IKE_SA_INIT request message. If the responder also supports this exchange, it includes this notification in the response message.

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

The INTERMEDIATE_EXCHANGE_SUPPORTED is a Status Type IKEv2 notification. Its Notify Message Type is <TBA by IANA>. Protocol ID

and SPI Size are both set to 0. This specification doesn't define any data this notification may contain, so the Notification Data is left empty. However, future enhancements of this specification may override this. Implementations MUST ignore the non-empty Notification Data if they don't understand its purpose.

3.2. Using Intermediate Exchange

If both peers indicated their support for the Intermediate Exchange, the initiator may use one or more these exchanges to transfer additional data. Using the INTERMEDIATE exchange is optional, the initiator may find it unnecessary after completing the IKE_SA_INIT exchange.

The Intermediate Exchange is denoted as INTERMEDIATE, its Exchange Type is <TBA by IANA>.

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

The initiator may use several INTERMEDIATE exchanges if necessary. Since initiator's Window Size is initially set to one (Section 2.3 of [RFC7296]), these exchanges MUST follow each other and MUST all be completed before the IKE_AUTH exchange is initiated. The IKE SA MUST NOT be considered as established until the IKE_AUTH exchange is successfully completed.

The Message IDs for the INTERMEDIATE exchanges MUST be chosen according to the standard IKEv2 rule, described in the Section 2.2. of [RFC7296], i.e. it is set to 1 for the first INTERMEDIATE exchange, 2 for the next (if any) and so on. The message ID for the first pair of the IKE_AUTH messages is one more than the one that was used in the last INTERMEDIATE exchange.

If the presence of NAT is detected in the IKE_SA_INIT exchange via NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP notifications, then the peers MUST switch to port 4500 immediately once this exchange is completed, i.e. in the first INTERMEDIATE exchange.

The content of the INTERMEDIATE exchange messages depends on the data being transferred and will be defined by specifications utilizing this exchange. However, since the main motivation for the INTERMEDIATE exchange is to avoid IP fragmentation when large amount of data need to be transferred prior to IKE_AUTH, the Encrypted payload MUST be present in the INTERMEDIATE exchange messages and

payloads containing large data MUST be placed inside. This will allow IKE Fragmentation [RFC7383] to take place, provided it is supported by the peers and negotiated in the initial exchange.

3.3. The INTERMEDIATE Exchange Protection and Authentication

3.3.1. Protection of the INTERMEDIATE Messages

The keys SK_e[i/r] and SK_a[i/r] for the Encrypted payload in the INTERMEDIATE exchanges are computed in a standard fashion, as defined in the Section 2.14 of [RFC7296]. Every subsequent INTERMEDIATE exchange uses the most recently calculated keys before this exchange is started. The first INTERMEDIATE exchange always uses SK_e[i/r] and SK_a[i/r] keys that were computed as result the IKE_SA_INIT exchange. If this INTERMEDIATE exchange performs additional key exchange resulting in the update of SK_e[i/r] and SK_a[i/r], then these updated keys are used for encryption and authentication of next INTERMEDIATE exchange, otherwise the current keys are used, and so on.

3.3.2. Authentication of the INTERMEDIATE Exchanges

The data transferred in the INTERMEDIATE exchanges must be authenticated in the IKE_AUTH exchange. For this purpose the definition of the blob to be signed (or MAC'ed) from the Section 2.15 of [RFC7296] is modified as follows:

```
InitiatorSignedOctets = RealMsg1 | NonceRData | MACedIDForI [| IntAuth]
ResponderSignedOctets = RealMsg2 | NonceIData | MACedIDForR [| IntAuth]

IntAuth =  IntAuth_1 | [| IntAuth_2 [| IntAuth_3]] ...

IntAuth_1 = IntAuth_1_I | IntAuth_1_R
IntAuth_2 = IntAuth_2_I | IntAuth_2_R
IntAuth_3 = IntAuth_3_I | IntAuth_3_R
...

IntAuth_1_I = prf(SK_pi_1, [IntAuth_1_I_P |] IntAuth_1_I_A)
IntAuth_2_I = prf(SK_pi_2, [IntAuth_2_I_P |] IntAuth_2_I_A)
IntAuth_3_I = prf(SK_pi_3, [IntAuth_3_I_P |] IntAuth_3_I_A)
...

IntAuth_1_R = prf(SK_pr_1, [IntAuth_1_R_P |] IntAuth_1_R_A)
IntAuth_2_R = prf(SK_pr_2, [IntAuth_2_R_P |] IntAuth_2_R_A)
IntAuth_3_R = prf(SK_pr_3, [IntAuth_3_R_P |] IntAuth_3_R_A)
...
```

IntAuth_1_I/IntAuth_1_R, IntAuth_2_I/IntAuth_2_R, IntAuth_3_I/IntAuth_3_R, etc. represent the results of applying the negotiated prf to the content of the INTERMEDIATE messages sent by the initiator (IntAuth_*_I) and by the responder (IntAuth_*_R) in an order of increasing Message IDs (i.e. in an order the INTERMEDIATE exchanges took place). The prf is applied to the two chunks of data: optional IntAuth_*_[I/R]_P and mandatory IntAuth_*_[I/R]_A. The IntAuth_*_[I/R]_A chunk lasts from the first octet of the IKE Header (not including prepended four octets of zeros, if port 4500 is used) to the last octet of the Encrypted Payload header. The IntAuth_*_[I/R]_P chunk is present if the Encrypted payload is not empty. It consists of the not yet encrypted content of the Encrypted payload, excluding Initialization Vector, Padding, Pad Length and Integrity Checksum Data fields (see 3.14 of [RFC7296] for description of the Encrypted payload). In other words, the IntAuth_*_[I/R]_P chunk is the inner payloads of the Encrypted payload in plaintext form.

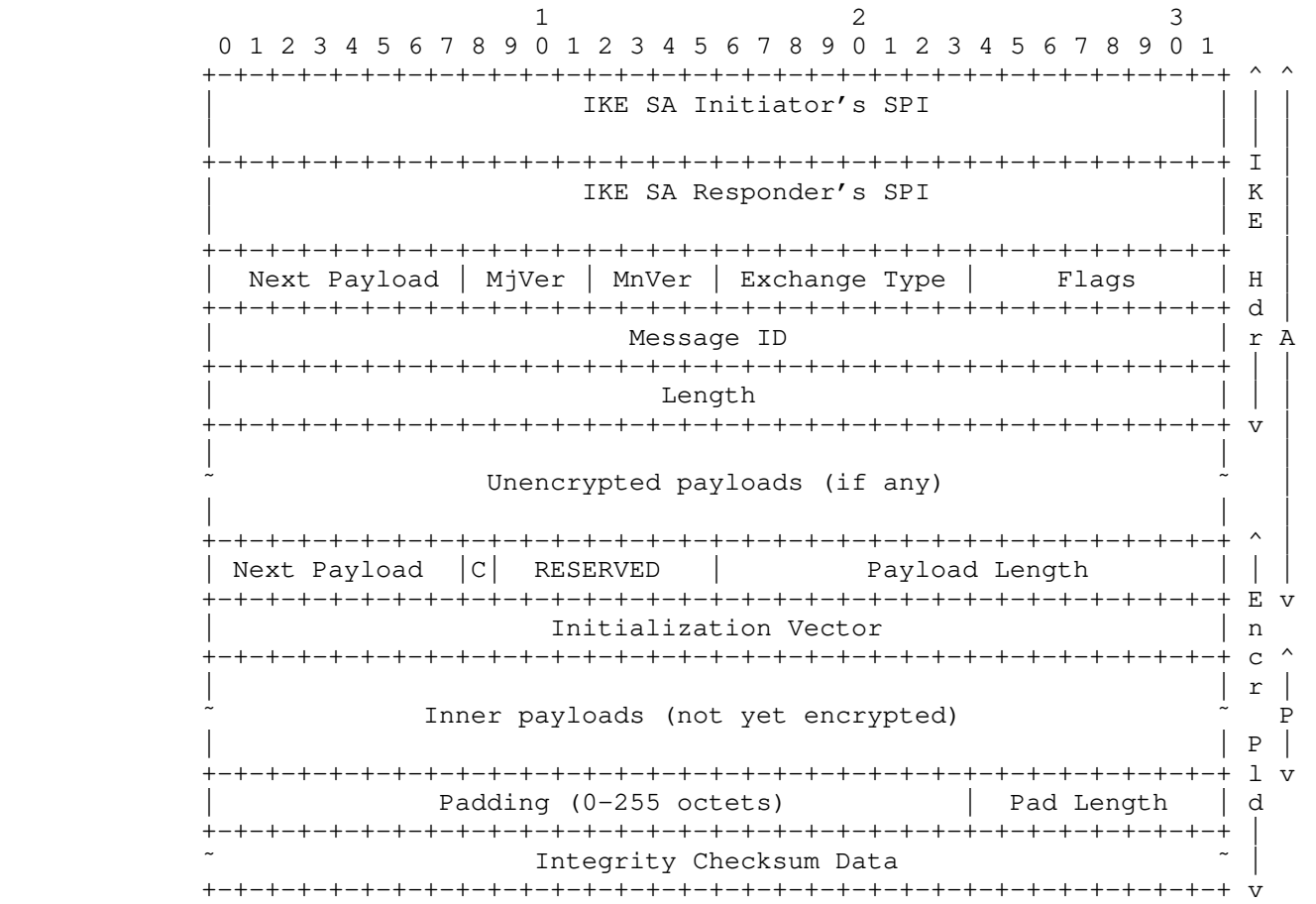


Figure 1: Data to Authenticate in the INTERMEDIATE Exchange Messages

Figure 1 illustrates the layout of the IntAuth_*_[I/R]_P (denoted as P) and the IntAuth_*_[I/R]_A (denoted as A) chunks in case the Encrypted payload is not empty.

The calculations are applied to whole messages only, before possible fragmentation. This ensures that the IntAuth will be the same regardless of whether fragmentation takes place or not ([RFC7383] allows sending first unfragmented message and then trying fragmentation in case of no reply).

Each calculation of IntAuth_*_[I/R] uses its own key SK_p[i/r]_*, which is the most recently updated SK_p[i/r] key available before the corresponded INTERMEDIATE exchange is started. The first INTERMEDIATE exchange always uses SK_p[i/r] key that was computed in

the IKE_SA_INIT as SK_p[i/r]_1. If the first INTERMEDIATE exchange performs additional key exchange resulting in SK_p[i/r] update, then this updated SK_p[i/r] is used as SK_p[i/r]_2, otherwise the original SK_p[i/r] is used, and so on. Note, that if keys are updated then for any given INTERMEDIATE exchange the keys SK_e[i/r] and SK_a[i/r] used for its messages protection (see Section 3.3.1) and the keys SK_p[i/r] for its authentication are always from the same generation.

3.4. Error Handling in the INTERMEDIATE Exchange

Since messages of the INTERMEDIATE exchange are not authenticated until the IKE_AUTH exchange successfully completes, possible errors need to be handled carefully. There is a trade-off between providing a better diagnostics of the problem and a risk to become a part of DoS attack. See Section 2.21.1 and 2.21.2 of [RFC7296] describe how errors are handled in initial IKEv2 exchanges, these considerations are applied to the INTERMEDIATE exchange too.

4. Interaction with other IKEv2 Extensions

The INTERMEDIATE exchanges MAY be used in the IKEv2 Session Resumption [RFC5723] between the IKE_SESSION_RESUME and the IKE_AUTH exchanges.

5. Security Considerations

The data that is transferred by means of the INTERMEDIATE exchanges is not authenticated until the subsequent IKE_AUTH exchange is completed. However, if the data is placed inside the Encrypted payload, then it is protected from passive eavesdroppers. In addition the peers can be certain that they receives messages from the party he/she performed the IKE_SA_INIT with if they can successfully verify the Integrity Checksum Data of the Encrypted payload.

The main application for Intermediate Exchange is to transfer large amount of data before IKE SA is set up without causing IP fragmentation. For that reason it is expected that in most cases IKE Fragmentation will be employed in the INTERMEDIATE exchanges. Section 5 of [RFC7383] contains security considerations for IKE Fragmentation.

Note, that if an attacker was able to break key exchange in real time (e.g. by means of Quantum Computer), then the security of the INTERMEDIATE exchange would degrade. In particular, such an attacker would be able both to read data contained in the Encrypted payload and to forge it. The forgery would become evident in the IKE_AUTH exchange (provided the attacker cannot break employed authentication

mechanism), but the ability to inject forged the INTERMEDIATE exchange messages with valid ICV would allow the attacker to mount Denial-of-Service attack. Moreover, if in this situation the negotiated prf was not secure against preimage attack with known key, then the attacker could forge the INTERMEDIATE exchange messages without later being detected in the IKE_AUTH exchange. To do this the attacker should find the same IntAuth_*_[I|R] value for the forged message as for original.

6. IANA Considerations

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

<TBA>	INTERMEDIATE

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

<TBA>	INTERMEDIATE_EXCHANGE_SUPPORTED

7. Acknowledgements

The idea to use an intermediate exchange between IKE_SA_INIT and IKE_AUTH was first suggested by Tero Kivinen. Scott Fluhrer and Daniel Van Geest identified a possible problem with authentication of the INTERMEDIATE exchange and helped to resolve it.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <https://www.rfc-editor.org/info/rfc7383>.

8.2. Informative References

[RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <https://www.rfc-editor.org/info/rfc8229>.

[RFC5723] Sheffer, Y. and H. Tschofenig, "Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption", RFC 5723, DOI 10.17487/RFC5723, January 2010, <https://www.rfc-editor.org/info/rfc5723>.

Author's Address

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

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

IPSECME Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2018

S. Piriyath
U. Mangla
N. Melam
R. Bonica
Juniper Networks

February 28, 2018

Packetization Layer Path Maximum Transmission Unit Discovery (PLPMTUD) For IPsec Tunnels
draft-spiriyath-ipsecme-dynamic-ipsec-pmtu-01

Abstract

This document describes Packetization Layer PMTU Discovery (PLPMTUD) procedures for IPSec tunnels. In these procedures, the encrypting node discovers and maintains a running estimate of the tunnel MTU. In order to do this, the encrypting nodes sends Probe Packets of various size through the IPSec tunnel. If the size of Probe Packet exceeds the tunnel MTU, a downstream node discards the packet and sends an ICMP PTB message to the encrypting node. The encrypting node ignores the ICMP PTB message. If the size of the Probe Packet does not exceed the tunnel MTU and the decrypting node receives the Probe Packet, the decrypting node sends an Acknowledgement Packet to encrypting node through the IPSec tunnel. The Acknowledgement Packet indicates the size of the Probe Packet.

The procedures described in this document are applicable to IPSec tunnels that are signaled by IKEv2 and provide authentication services.

Table of Contents

1. Introduction

IPsec [RFC4301] tunnels provides private and/or authenticated connectivity between an encrypting node and a decrypting node. An IPsec tunnel is constrained by the number of bytes that it can convey in a single packet, without fragmentation of any kind. This constraint is called the tunnel Maximum Transmission Unit (MTU). An IPSec tunnel's MTU can be calculated as its Path MTU (PMTU) minus IPSec tunnel overhead, where:

- PMTU is the smallest MTU of all the links forming a path between the encrypting node and the decrypting node.

- IPSec tunnel overhead is the maximum number of bytes required for padding (by the encryption algorithm) plus the number of bytes required for IPSec encapsulation.

When forwarding a packet through an IPSec tunnel, the encrypting node compares the packet's length to the tunnel MTU. If the packet length is less than or equal to the tunnel MTU, the encrypting node encrypts the packet, encapsulates it and forwards it through the IPSec tunnel.

If the packet length is greater than the tunnel MTU and the packet cannot be fragmented, the encrypting node discards the packet and sends an ICMP [RFC0792] [RFC4443] Packet Too Big (PTB) message to the packet's source.

If the packet length is greater than the tunnel MTU and the packet can be fragmented, the encrypting node can execute either of the following procedures:

- Fragment, encrypt and encapsulate (FEE)

- Encrypt, encapsulate and fragment (EEF)

If the encrypting node executes FEE procedures, it fragments the packet first. Then it encrypts, encapsulates and forwards each fragment. When a fragment arrives at the decrypting node, the decrypting node decapsulates and decrypts the fragment. Finally, the decrypting node forwards the fragment to its ultimate destination, where it can be reassembled.

If the encrypting node executes EEF procedures, it encrypts and encapsulates the packet first. Then it fragments the resulting packet and forwards each fragment to the decrypting node. When the decrypting node has received all fragments, it reassembles the packet, decapsulates and decrypts it. Finally, it forwards the packet, in one piece, to its ultimate destination.

In the paragraphs above, IPv4 [RFC0791] packets with the Don't Fragment (DF) bit set to zero can be fragmented. IPv6 [RFC8200] packets and IPv4 packets with the DF bit set to one cannot be fragmented.

In the above-described procedure, the encrypting node maintains an estimate of the tunnel MTU. Network operators can configure the tunnel MTU on the encrypting node. Alternatively, they can configure the encrypting node to automatically discover and maintain a running estimate of the tunnel MTU. Today, when a encrypting node is configured to automatically discover the tunnel MTU, it executes ICMP-based PMTU Discovery (PMTUD) [RFC1191] [RFC8201] procedures. Having discovered the PMTU, it calculates the tunnel MTU by subtracting the IPSec tunnel overhead from the PMTU.

The above-mentioned ICMP-based PMTUD procedures are susceptible to attack [I-D.roca-ipsecme-ptb-pts-attack]. An attacker can forge an ICMP PTB message, setting the MTU to a low value. When the encrypting node receives the forged ICMP PTB message, it decreases its estimate of tunnel MTU, causing unnecessary fragmentation. Therefore, many IPsec implementations do not implement tunnel MTU discovery at all.

This document describes Packetization Layer PMTU Discovery (PLPMTUD) procedures for IPSec tunnels. In these procedures, the encrypting node discovers and maintains a running estimate of the tunnel MTU. In order to do this, the encrypting nodes sends Probe Packets of various size through the IPSec tunnel. If the size of Probe Packet exceeds the tunnel MTU, a downstream node discards the packet and sends an ICMP PTB message to the encrypting node. The encrypting node ignores the ICMP PTB message. If the size of the Probe Packet does not exceed the tunnel MTU and the decrypting node receives the Probe Packet, the decrypting node sends an Acknowledgement Packet to encrypting node through the IPSec tunnel. The Acknowledgement Packet indicates the size of the Probe Packet. Unlike ICMP PTB messages, this Acknowledgement Packet cannot be forged.

The procedures described in this document are applicable to IPSec tunnels that are signaled by Internet Key Exchange version 2 (IKEv2) [RFC7296] and provide authentication services.

2. Requirements Language

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

3. PLPMTU Discovery Procedures

3.1. Method of Operation

A special loopback interface is configured on the encrypting and decrypting nodes. In this document, these loopback interfaces are called the IPSec PLPMTUD interfaces.

The encrypting node executes IKEv2 procedures to signal an IPSec tunnel between itself and a decrypting node. The IPSec tunnel MUST provide authentication services. It MAY also provide privacy services. If the outermost header of the IPSec tunnel is an IPv4 header, the DF bit must be set. IKEv2 endpoints MUST exchange traffic selectors advertising their IPSec PLPMTUD interface

addresses. Implementations MUST ensure that traffic from one IPSec PLPMTUD address to another traverses the appropriate tunnel using the correct security association.

As part of the tunnel establishment process, the encrypting node produces an initial estimate of the tunnel MTU. The encrypting node's initial estimate of the tunnel MTU is equal to its initial PMTU estimate minus IPSec tunnel overhead, where:

- The initial PMTU estimate is equal to the MTU of the first link along the path between the encrypting node and the decrypting node.

- IPSec tunnel overhead is the maximum number of bytes required for padding (by the encryption algorithm) plus the number of bytes required for IPSec encapsulation.

This initial estimate may be greater than the actual tunnel MTU.

Having established the IPSec tunnel, the ingress node begins to refine its estimate of the tunnel MTU. It MAY pass traffic through the tunnel as it refines the tunnel MTU estimate.

In order to refine its estimate of the tunnel MTU, the ingress node executes the Packetization Layer PMTU Discovery (PLPMTUD) procedures described in Section 4 of [I-D.fairhurst-tsvwg-datagram-plpmtud]. When applied to IPSec tunnels, these procedures can be summarized as follows:

- The encrypting node sends Probe Packets of various size through the IPSec tunnel.

- If the size of the Probe Packet exceeds the tunnel MTU, a downstream device drops the packet and sends an ICMP Packet Too Big (PTB) message to the encrypting node. The encrypting node ignores the ICMP PTB message.

- If the Probe Packet reaches the decrypting node, the decrypting node acknowledges receipt of the Probe Packet.

Section 3.2 of this document describes the Probe Packet. Section 3.3 of this document describes how the decrypting node acknowledges receipt of the Probe Packet.

3.2. PLPMTUD Probe

The encrypting node can probe the IPSec tunnel using an IPv4 packet or an IPv6 packet. Figure 1 depicts the IPv4 Probe Packet while Figure 2 depicts the IPv6 Probe Packet. In either case, the encrypting node forwards the Probe Packet through the IPSec tunnel.

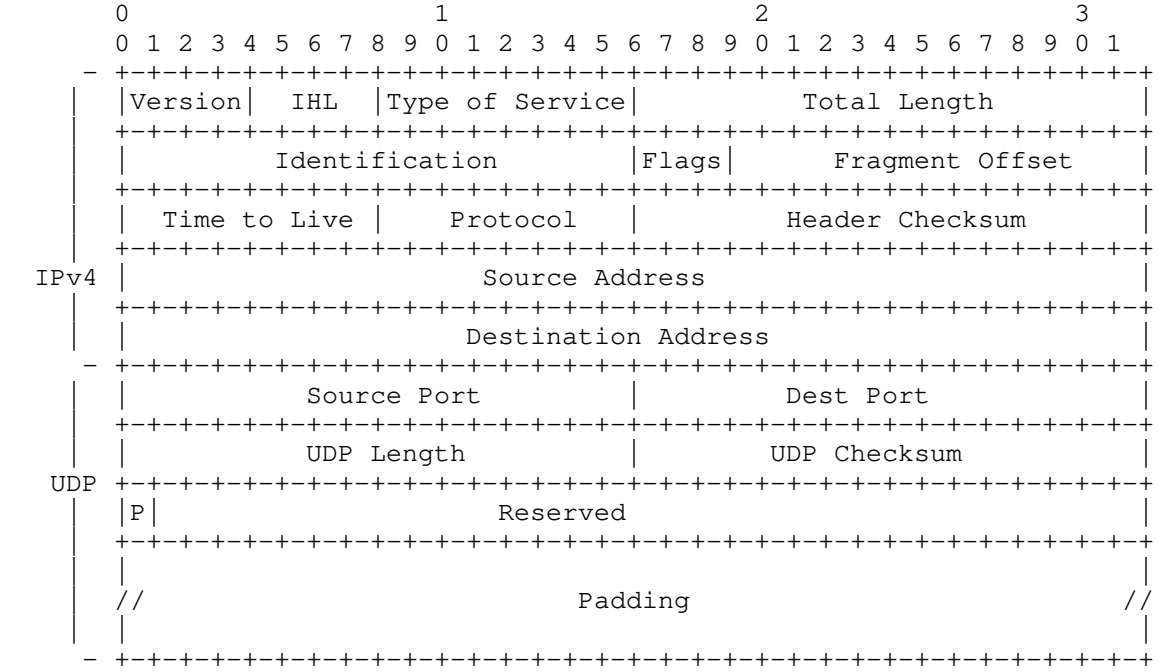


Figure 1: IPv4 Probe Packet

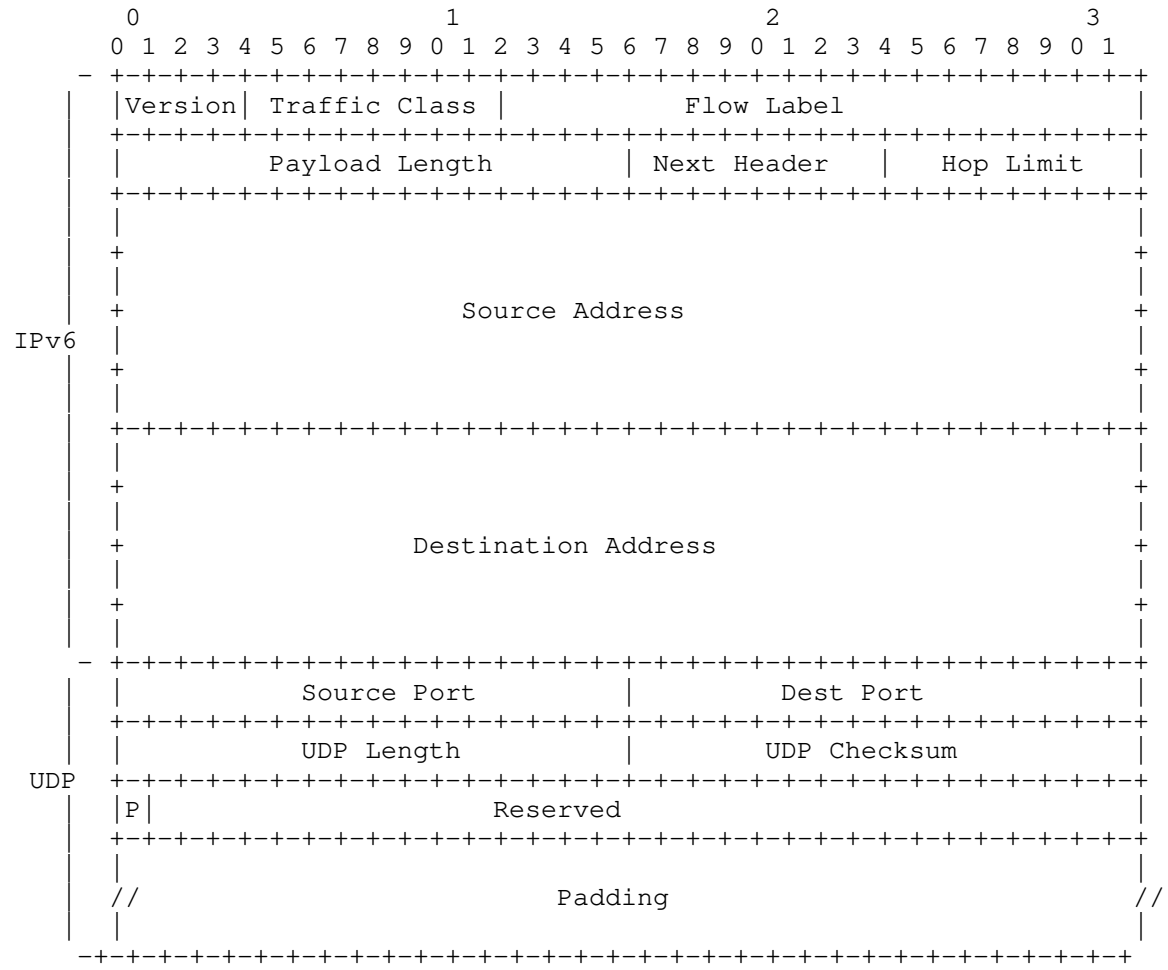


Figure 2: IPv6 Probe Packet

Regardless of whether the Probe Packet is IPv4 or IPv6:

- The Source Address is the encrypting node's IPSec PLPMTUD interface address.
- The Destination Address is the decrypting node's IPSec PLPMTUD interface address.
- The Source Port is chosen from the dynamic port range (49152-65535) [RFC6335]

- The Destination Port is equal to IPSec PLPMTUD. (Value TBD by IANA).
- The P-bit is set to indicate that this is a Probe Packet.
- The Reserved Field MUST be set to zero and MUST be ignored upon receipt.
- The Padding field is used to vary the size of the packet.

3.3. PLPMTUD Acknowledgement

When the decrypting node receives a Probe Packet, it returns an Acknowledgment Packet. The Acknowledgment Packet can be an IPv4 packet or an IPv6 packet. Figure 3 depicts the IPv4 Acknowledgment Packet while Figure 4 depicts the IPv6 Acknowledgment Packet. In either case, the decrypting node forwards the Acknowledgment Packet through the IPSec tunnel that connects it to the encrypting node.

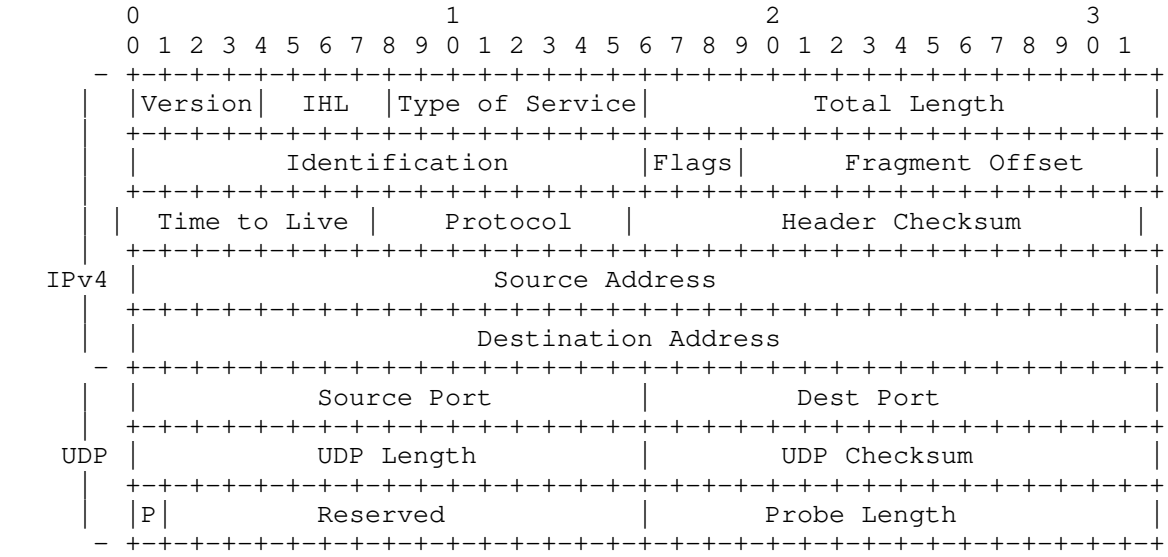


Figure 3: IPv4 Acknowledgement Packet

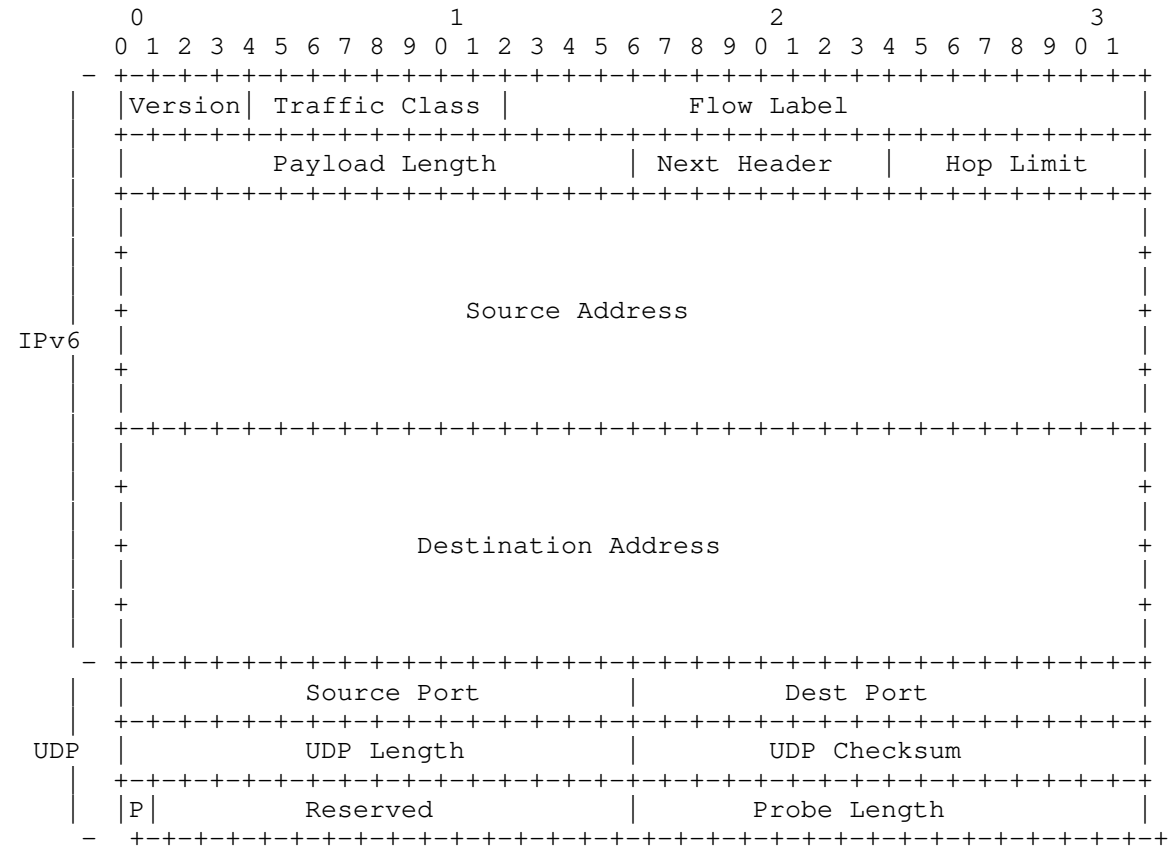


Figure 4: IPv6 Acknowledgement Packet

Regardless of whether the Acknowledgment Packet is IPv4 or IPv6:

- The Source Address is copied from the Destination Address of the corresponding Probe Packet
- The Destination Address is copied from the Source Address of the corresponding Probe Packet
- The Source Port is copied from the Destination Port of the corresponding Probe Packet
- The Destination Port copied from the Source Port of the corresponding Probe Packet
- The P-bit is clear to indicate that this is an Acknowledgement Packet.

- The Reserved Field MUST be set to zero and MUST be ignored upon receipt.

- The Probe length represents the total length of the corresponding Probe Packet, measured in bytes and not counting IPSec overhead

4. Security Considerations

The procedures described herein are an improvement upon ICMP-based PMTUD procedures because unlike ICMP PTB messages, the Acknowledgement Packets described herein cannot be forged.

The decrypting node MUST protect the encrypting node from forged Acknowledgement Packets. Therefore, the decrypting MAY originate packets whose source address is its PLPMTUD interface address. However, it MUST NOT forward packets whose source address is its PLPMTUD interface address.

5. ECMP Considerations

Packets traversing a network, with multi paths (ECMP), would end up picking the lowest MTU available in any of the ECMP paths, when the proposed solution is employed (assuming that paths have different MTU values for the sake of analysis). This might cause some additional load on the encryption end, due to the lower MTU level fragmentation. This wouldn't be a major issue, as even otherwise, these loads would have got processed on the receiving side (decryption side) for reassembly and holding the packets in memory. It is worth noting that, at the encryption side it is more of 'stateless' action in terms of packet fragmentation is concerned as compared to at decryption side it is more of a 'stateful' action, where in, it need to maintain the fragments queue for reassembly. Moreover, reassembly node has no control over arrival of the fragments. So, when a choice has to be made for loading the end between encryption and decryption end, it is always better to load the encryption side due to the fact that the operation is stateless and less costly to perform comparatively.

6. IANA Considerations

IANA is request to allocate a UDP port called "IPSec PLPMTUD" from the Registered Port Range (1024 to 49151).

7. Acknowledgements

Thanks to Yoav Nir, Joe Touch, and Dan Wing for their review and comments.

8. References

8.1. Normative References

[I-D.fairhurst-tsvwg-datagram-plpmtud]
Fairhurst, G., Jones, T., Tuexen, M., and I. Ruengeler, "Packetization Layer Path MTU Discovery for Datagram Transports", draft-fairhurst-tsvwg-datagram-plpmtud-02 (work in progress), December 2017.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[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>.

[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>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[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>.

[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>.

8.2. Informative References

[I-D.roca-ipsecme-ptb-pts-attack] Roca, V. and S. Fall, "Too Big or Too Small? The PTB-PTS ICMP-based Attack against IPsec Gateways", draft-roca-ipsecme-ptb-pts-attack-00 (work in progress), July 2015.

Authors' Addresses

Shibu Piriyath
Juniper Networks
Elnath-Exora Business Park
Bangalore, KA 93117
India

Email: spiriyath@juniper.net

Umesh Mangla
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
USA

Email: umangla@juniper.net

Nagavenkata Suresh Melam
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
USA

Email: nmelam@juniper.net

Ron Bonica
Juniper Networks
2251 Corporate Park Drive
Herndon, Virginia 20171
USA

Email: rbonica@juniper.net

Network
Internet-Draft
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: September 5, 2018

S. Prasad
Technical University of Munich
P. Wouters
Red Hat

March 4, 2018

Labeled IPsec Traffic Selector support for IKEv2
draft-sprasad-ipsecme-labeled-ipsec-00

Abstract

Some IPsec implementations support Security Labels otherwise known as Security Contexts, to be configured as a selector within the Security Policy Database (SPD) for IPsec SAs. This document adds support to IKEv2 to negotiate these Security Labels or Contexts using a new Traffic Selector (TS) Type TS_SECLABEL. The approach is named "Labeled IPsec". It assumes that the SPD processing of RFC 4303 is already extended to support Security Labels. This document only adds the ability for IKE to negotiate the Security Labels used with the SPD.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 5, 2018.

Table of Contents

1. Introduction

A Security Context or Security Label is a mechanism used to classify resources. It allows the enforcement of rules on how or by whom these resources are accessable. This document introduces a mechanism to negotiate these values using IKE. This negotiation is done via a new Traffic Selector (TS) Type in the TSi/TSr Payloads of the IKE_AUTH Exchange.

Traffic Selector (TS) payloads allow endpoints to communicate some of the information from their SPD to their peers. Section 2.9 in the Internet Key Exchange protocol version 2 [RFC7296] illustrates the Traffic selector negotiation procedure.

Two or more TS payloads appear in each of the messages in the exchange that creates a Child SA pair. Each TS payload contains one or more Traffic Selectors. The Traffic Selector types TS_IPV4_ADDR_RANGE and TS_IPV6_ADDR_RANGE consists of an address range, a port range, and an IP protocol ID. [RFC4595] defines Traffic Selector type TS_FC_ADDR_RANGE to denote a list of Fibre Channel (FC) addresses and protocol ranges. This document extends the above set by adding a new TS Type that allows endpoints to agree on assigning a Security Label or Context to the IPsec SA.

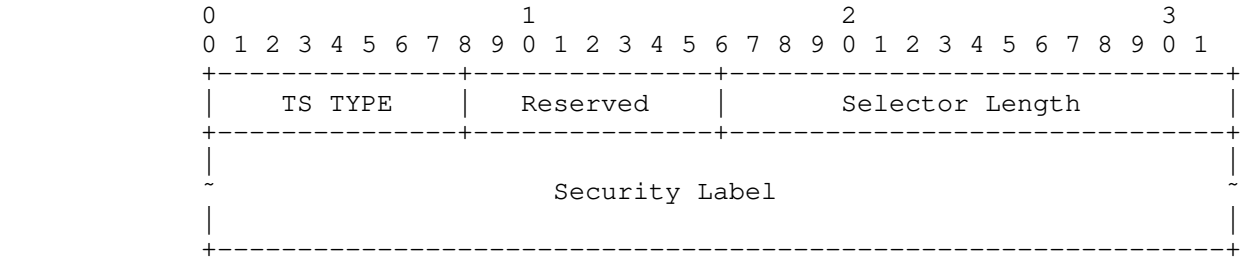
Negotiating and applying the Security Label or Context in the new TS Type will act as an additional selector criterium that has to match along with any other existing Traffic Selectors.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Labeled IPsec Traffic Selector

Labeled IPsec Traffic Selectors allow endpoints to negotiate the Security Label using the Selector Type TS_SECLABEL. In addition to the regular prcessing of Traffic Selectors as described in Section 3.13.1 of [RFC7296], the context or label of an IP packet now also has to match the Security Label Traffic Selector.



- TS TYPE (one octet) - Specifies the type of Traffic Selector.
- Selector Length (2 octets, network byte order) - Specifies the length of Security Label including the header.
- Security Label - This field contains the opaque payload.

The following table lists the assigned value for the Labeled IPsec Traffic Selector Type field:

TS Type	Value
TS_SECLABEL	[TBD]

To indicate support and a requirement for agreeing on a specific security context, the initiator MUST include the security context or label via TS_SECLABEL in the TSi (Traffic Selector-initiator) and TSr (Traffic Selector-responder) Payloads. On reception of TS_SECLABEL, the responder MUST find a matching Security Policy Database (SPD) entry that contains a Security Label and match the proposed label. Assuming that this proposal was acceptable to the responder, it would send the same or narrowed TS payloads in the IKE_AUTH reply. If the

Security Label was not found in the SPD by the responder, it MUST respond with a TS_UNACCEPTABLE Notify message.

As per section 2.9.2 in [RFC7296], TS sets MUST BE kept identical during rekey. If a Security Label needs to change, the IPsec SA must be torn down and a new one must be negotiated with the updated TS_SECLABEL values.

2.1. Narrowing of Labeled IPsec Traffic Selector

The IKE daemon might or might not be able to interpret the Security Label beyond an exact match. It might be possible for the IKE daemon to apply narrowing to the Security Labels. For example, a Security Label "Top Secret" could mean that this IPsec SA may also transport traffic with label "Secret". An initiator requesting "Top Secret" might be willing to be narrowed down to a "Secret" security context. Or a security context of "*" might mean "any security context". If the daemon does not interpret the Security Label, then it can only support an exact match of the raw data of the TS.

The rules of responder narrowing as explained in section 2.9 in [RFC7296] are applicable to TS_SECLABEL.

The TS_SECLABEL Traffic Selector Type if present MUST be mutually agreed upon. If one side includes a TS_SECLABEL and the other sides does not, the IPsec SA negotiation MUST fail with TS_UNAVAILABLE. If a responder insists on a TS_SECLABEL security context and receives a TSi/TSr set that does not contain a TS_SECLABEL Traffic Selector, it MUST fail the negotiation with TS_UNAVAILABLE.

DISCUSS: Should a TS_SECLABEL of length 0 be allowed? If so, should it mean "any label" ?

3. Security Considerations

While matching the Security Label on the endpoints, an assumption that the Security label will contain only ASCII text MUST NOT be made. If the Security Label is handed off to a helper routine for interpretation, it MUST be assumed that the content can be malicious. While Security Labels might look like text, there is no guarantee this text is null terminated.

Any errors in handling the SPD entry, such as failing to add the SPD entry with the negotiated Security Label, MUST be abled as any other failure of SPD processing as defined in [RFC4303].

4. IANA Considerations

This document defines one new Traffic Selector Type in the IKEv2 Traffic Selector Types Registry namespace.

TS Type	Value
TS_SECLABEL	[TBD]

Figure 1

5. References

5.1. 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>.

[RFC4595] Maino, F. and D. Black, "Use of IKEv2 in the Fibre Channel Security Association Management Protocol", RFC 4595, DOI 10.17487/RFC4595, July 2006, <https://www.rfc-editor.org/info/rfc4595>.

[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>.

5.2. Informative References

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

Authors' Addresses

Sahana Prasad
Technical University of Munich

Email: sahana.prasad07@gmail.com

Paul Wouters
Red Hat

Email: pwouters@redhat.com

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.

Table of Contents

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. In 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

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

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

- Using multiple key exchanges CREATE_CHILD_SA is defined.

draft-tjhai-ipsecme-hybrid-qske-ikev2-02

- 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

- Use IKE_INTERMEDIATE to perform multiple key exchanges in succession.
- 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.

- Simplify the negotiation of the 'extra' key exchanges.

draft-tjhai-ipsecme-hybrid-qske-ikev2-00

- 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.

- 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:

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

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) | SK_ai(n) | SK_ar(n) | SK_ei(n) | SK_er(n) | SK_pi(n) |
 SK_pr(n)} = prf+ (SKEYSEED(n), Ni(n) | Nr(n) | SPIi | 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 CRETE_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:

```
SKEYSEED = prf(SK_d, KE  | Ni | Nr | KE(1) | Ni(1) | Nr(1) ...
                                   | KE(n) | Ni(n) | Nr(n))
```

In case of Child SA creating or rekey:

```
KEYMAT = prf+ (SK_d, KE  | Ni | Nr | KE(1) | Ni(1) | Nr(1) ...
                                   | KE(n) | Ni(n) | Nr(n))
```

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, KEx, using an algorithm defined in Transform Type 2. While a quantum attacker may learn the value of KEx', if this value is obtained by means of a classical key exchange, other KEx values generated by means of a quantum-resistant algorithm ensure that the final SKEYSEED is not compromised. This assumes that the algorithm defined in the Transform Type 2 is post-quantum.

The main focus of this document is to prevent a passive attacker performing a "harvest and decrypt" attack. In other words, an attacker that records messages exchanges today and proceeds to decrypt them once he owns a quantum computer. This attack is prevented due to the hybrid nature of the key exchange. Other attacks involving an active attacker using a quantum-computer are not completely solved by this document. This is for two reasons.

The first reason is because the authentication step remains classical. In particular, the authenticity of the SAs established under IKEv2 is protected using a pre-shared key, RSA, DSA, or ECDSA algorithms. Whilst the pre-shared key option, provided the key is long enough, is post-quantum, the other algorithms are not. Moreover, in implementations where scalability is a requirement, the pre-shared key method may not be suitable. Quantum-safe authenticity may be provided by using a quantum-safe digital signature and several quantum-safe digital signature methods are being explored by IETF. For example, if the implementation is able to reliably track state, the hash based method, XMSS has the status of an RFC, see [RFC8391]. Currently, quantum-safe authentication methods are not specified in this document, but are planned to be incorporated in due course.

It should be noted that the purpose of post-quantum algorithms is to provide resistance to attacks mounted in the future. The current threat is that encrypted sessions are subject to eavesdropping and archived with decryption by quantum computers taking place at some point in the future. Until quantum computers become available there is no point in attacking the authenticity of a connection because there are no possibilities for exploitation. These only occur at the time of the connection, for example by mounting a MitM attack. Consequently there is not such a pressing need for quantum-safe authenticity.

This draft does not attempt to address key exchanges with KE payloads longer than 64k; the current IKE payload format does not allow that as a possibility. If such huge KE payloads are required, a work around (such as making the KE payload a URL and a hash of the real

payload) would be needed. At the current time, it appears likely that there will be plenty of key exchanges available that would not require such a workaround.

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:

- 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.

- Sending post-quantum proposals and policies in KE payload only

  With the objective of not introducing unnecessary notify payloads, we considered communicating the hybrid post-quantum proposal in the KE payload during the first pass of the protocol exchange. Unfortunately, this design is susceptible to the following downgrade attack. Consider the scenario where there is an MitM attacker sitting between an initiator and a responder. The initiator proposes, through SAi payload, to use a hybrid post-quantum group and as a backup a Diffie-Hellman group, and through KEi payload, the initiator proposes a list of hybrid post-quantum

proposals and policies. The MitM attacker intercepts this traffic and replies with N(INVALID_KE_PAYLOAD) suggesting to downgrade to the backup Diffie-Hellman group instead. The initiator then resends the same SAi payload and the KEi payload containing the public value of the backup Diffie-Hellman group. Note that the attacker may forward the second IKE_SA_INIT message only to the responder, and therefore at this point in time, the responder will not have the information that the initiator prefers the hybrid group. Of course, it is possible for the responder to have a policy to reject an IKE_SA_INIT message that (a) offers a hybrid group but not offering the corresponding public value in the KEi payload; and (b) the responder has not specifically acknowledged that it does not supported the requested hybrid group. However, the checking of this policy introduces unnecessary protocol complexity. Therefore, in order to fully prevent any downgrade attacks, using KE payload alone is not sufficient and that the initiator MUST always indicate its preferred post-quantum proposals and policies in a notify payload in the subsequent IKE_SA_INIT messages following a N(INVALID_KE_PAYLOAD) response.

- 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.

- 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

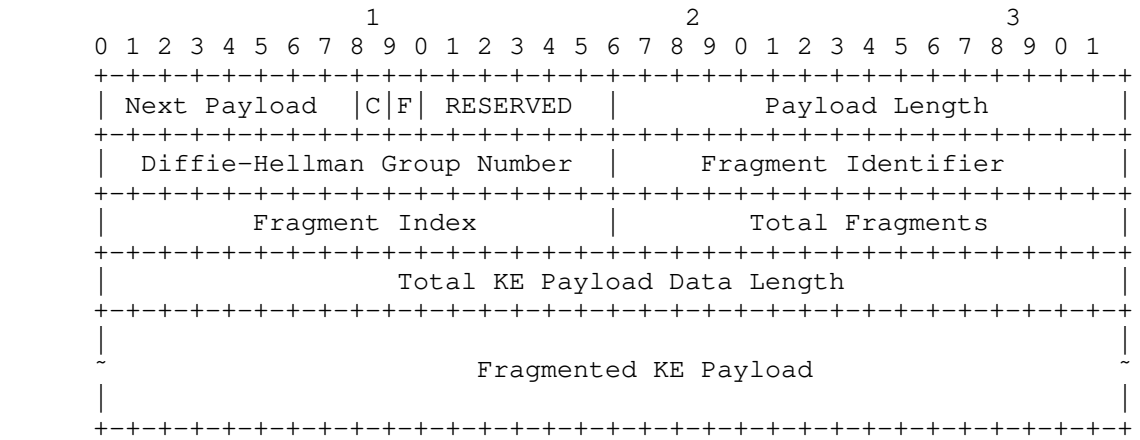
reasoning as above, this may not be a good idea from a backward compatibility perspective. Using a single KE payload would require an encoding or formatting to be defined so that both peers are able to compose and extract the individual public values. However, we believe that it is cleaner to send the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

- 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.

- 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

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

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.)

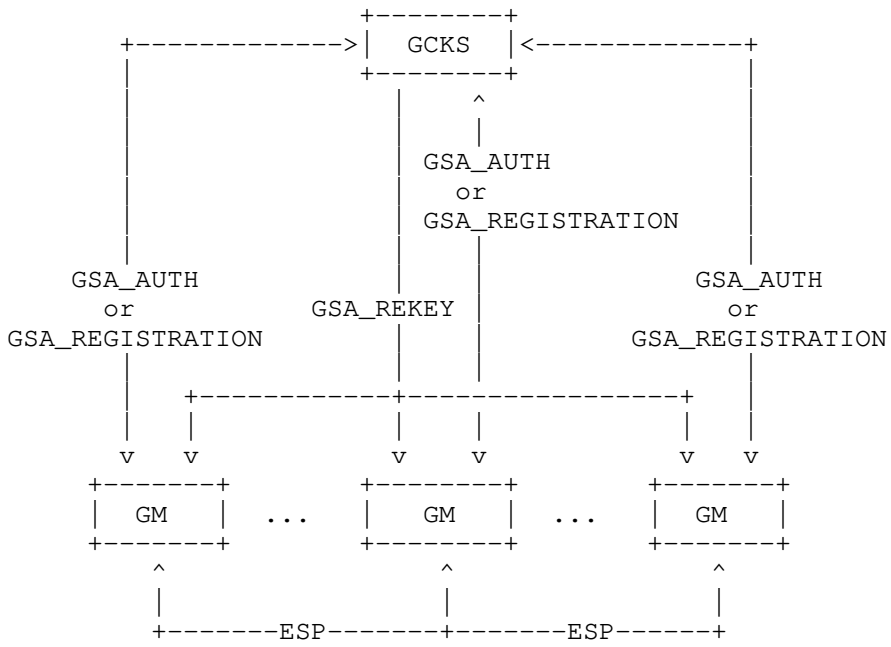


Figure 1: G-IKEv2 used in large groups

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.)

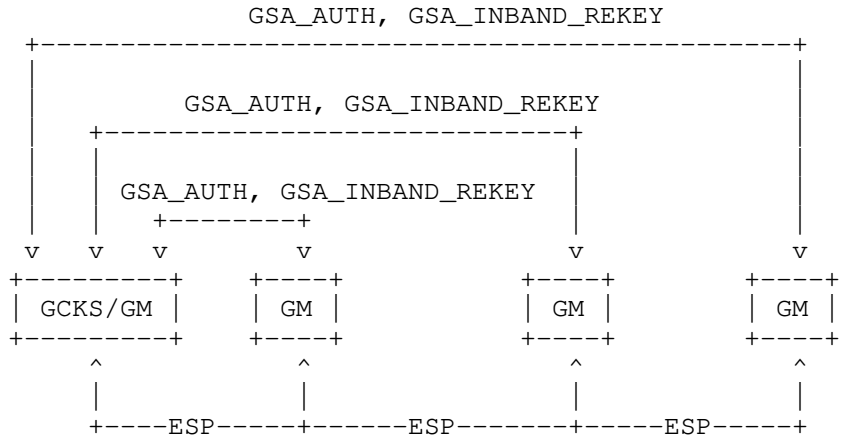


Figure 2: G-IKEv2 used in small groups

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:

- IDg (Group ID) - The GM requests the GCKS for membership into the group by sending its IDg payload.
- GSA (Group Security Association) - The GCKS sends the group policy to the GM using this payload.
- 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)
   --------------------              ------------------
    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)
   --------------------              ------------------
    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)
   --------------------              ------------------
    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.

- 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.

- 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.

- 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 doesen'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 Digiatal 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 not 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.

- 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.

- 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.

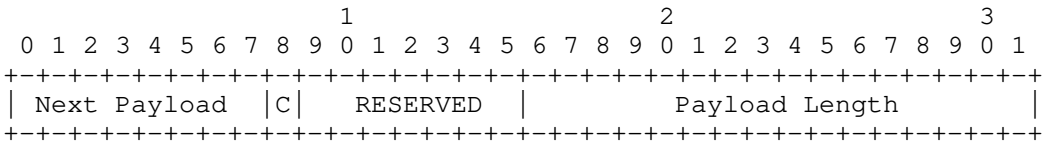


Figure 18: GSA Payload Format

The Security Association Payload fields are defined as follows:

- Next Payload (1 octet) -- Identifies the next payload type for the G-IKEv2 registration or the G-IKEv2 rekey message.
- Critical (1 bit) -- Set according to [RFC7296].
- RESERVED (7 bits) -- Must be zero.
- 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.

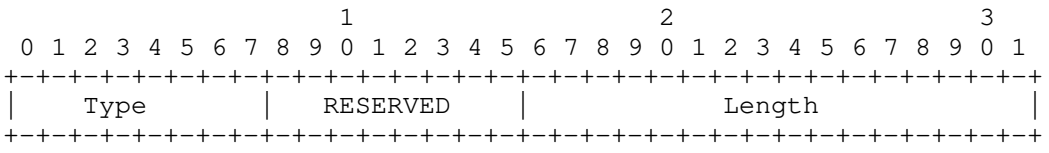


Figure 19: GSA Policy Generic Header Format

The payload fields are defined as follows:

- 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

- RESERVED (1 octet) -- Unused, set to zero.

- 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.

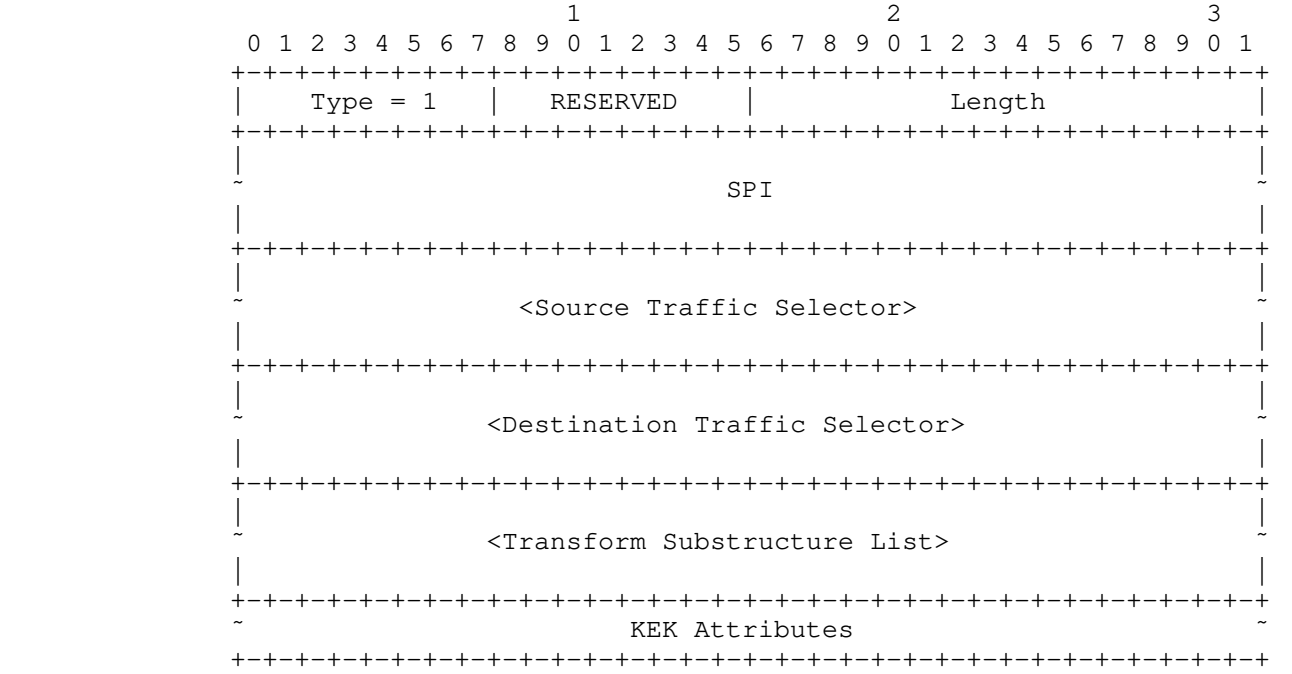


Figure 20: KEK Policy Format

The GSA KEK Payload fields are defined as follows:

- Type = 1 (1 octet) -- Identifies the GSA payload type as KEK in the G-IKEv2 registration or the G-IKEv2 rekey message.
- RESERVED (1 octet) -- Must be zero.
- Length (2 octets) -- Length of this structure including KEK attributes.
- 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.

- 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.

- 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.

- 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.

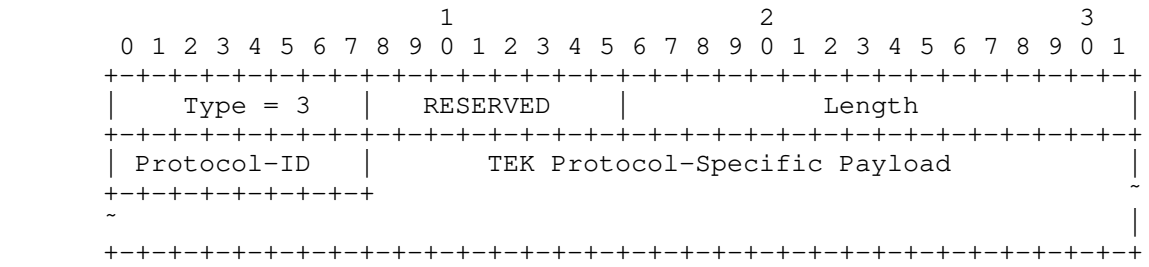


Figure 21: TEK Policy Generic Header Format

The GSA TEK Payload fields are defined as follows:

- Type = 3 (1 octet) -- Identifies the GSA payload type as TEK in the G-IKEv2 registration or the G-IKEv2 rekey message.

- RESERVED (1 octet) -- Must be zero.

- Length (2 octets) -- Length of this structure, including the TEK Protocol-Specific Payload.

- 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

- 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:

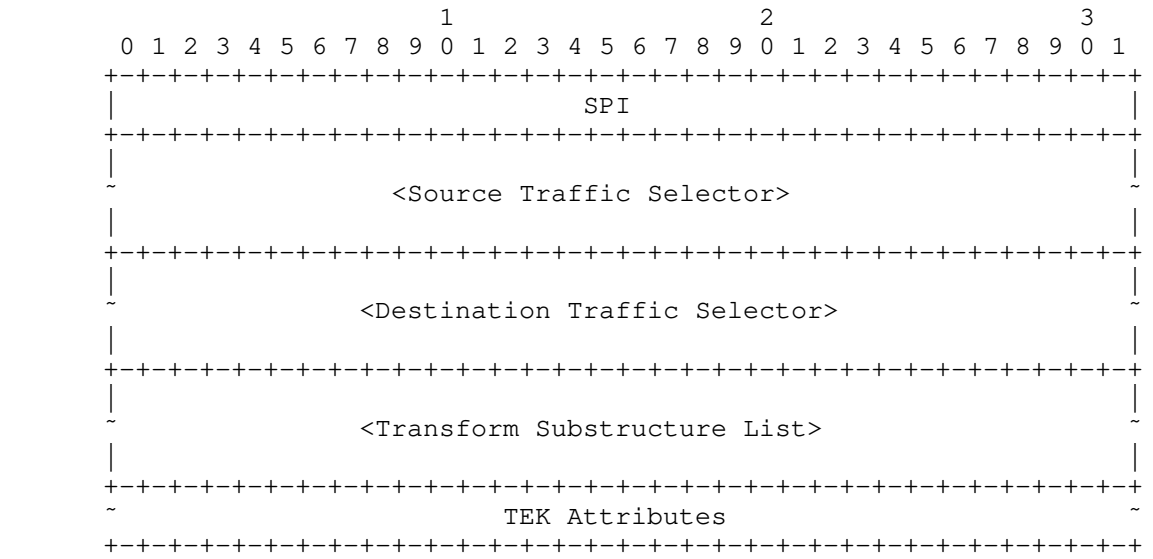


Figure 22: AH and ESP TEK Policy Format

The GSA TEK Policy fields are defined as follows:

- SPI (4 octets) -- Security Parameter Index.

- 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.

- 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.

- 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:

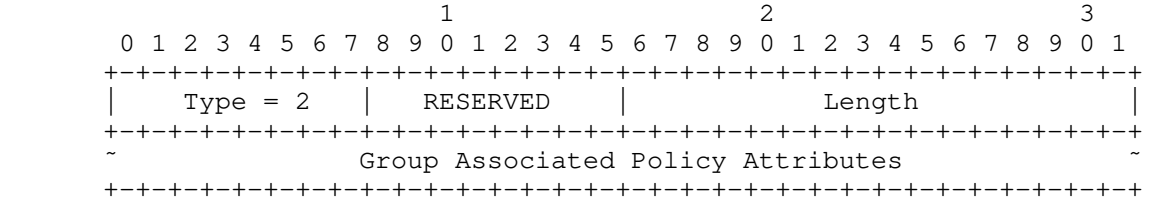


Figure 23: GAP Policy Format

The GSA GAP payload fields are defined as follows:

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

- RESERVED (1 octet) -- Must be zero.

- Length (2 octets) -- Length of this structure, including the GSA GAP header and Attributes.

- 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.

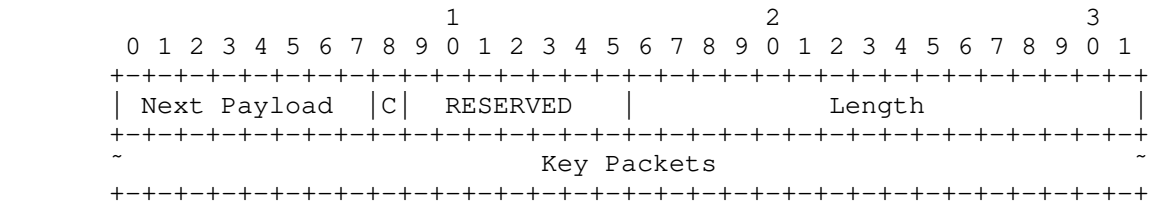


Figure 24: Key Download Payload Format

The Key Download Payload fields are defined as follows:

- 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.

- Critical (1 bit) -- Set according to [RFC7296].

- RESERVED (7 bits) -- Unused, set to zero.

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

- Key Packets (variable) -- Contains Key Packets. Several types of key packets are defined. Each Key Packet has the following format.

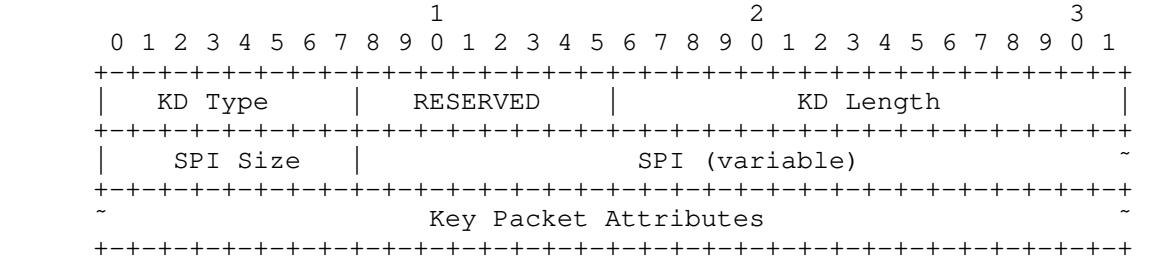


Figure 25: Key Packet Format

- 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

- RESERVED (1 octet) -- Unused, set to zero.

- Key Download Length (2 octets) -- Length in octets of the Key Packet data, including the Key Packet header.

- SPI Size (1 octet) -- Value specifying the length in octets of the SPI as defined by the Protocol-Id.

- SPI (variable length) -- Security Parameter Index which matches a SPI previously sent in an GSA KEK or GSA TEK Payload.

- 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.

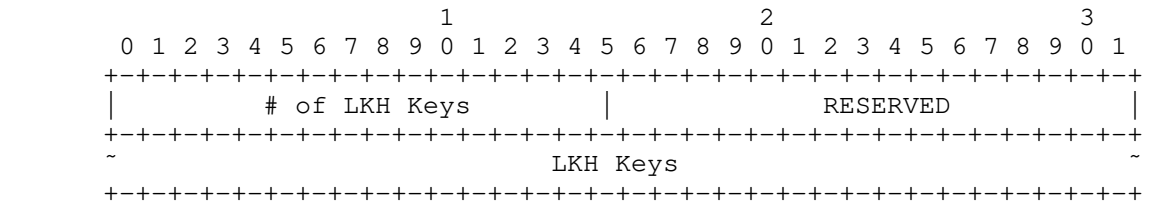


Figure 26: LKH_DOWNLOAD_ARRAY Format

The KEK_LKH attribute fields are defined as follows:

- Number of LKH Keys (2 octets) -- This value is the number of distinct LKH keys in this sequence.

- RESERVED (2 octets) -- Unused, set to zero.

Each LKH Key is defined as follows:

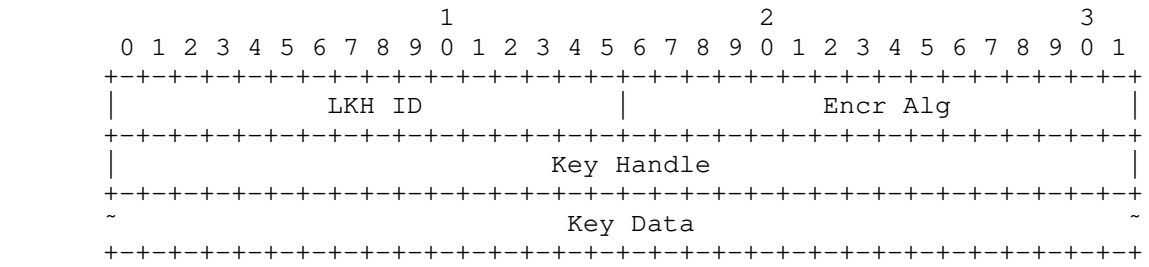


Figure 27: LKH Key Format

- LKH ID (2 octets) -- This is the position of this key in the binary tree structure used by LKH.

- 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.

- Key Handle (4 octets) -- This is a randomly generated value to uniquely identify a key within an LKH ID.

- 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.

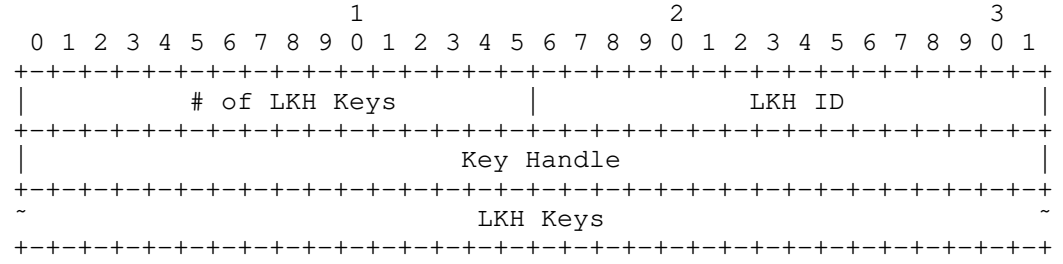


Figure 28: LKH_UPDATE_ARRAY Format

- Number of LKH Keys (2 octets) -- This value is the number of distinct LKH keys in this sequence.

- LKH ID (2 octets) -- This is the node identifier associated with the key used to encrypt the first LKH Key.

- 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 GKCS 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>.

[NNL] Naor, D., Noal, M., and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", Advances in Cryptology, 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>.

[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.

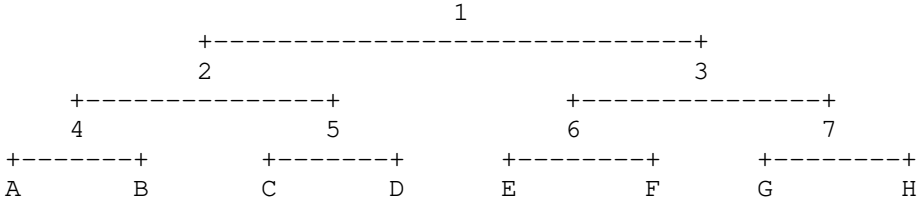


Figure 29: Initial LKH tree

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 GCKS 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 GCKS 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.

- One will provide GM "A" with new logical keys that are shared with B: {4'}A, {2'}4', {1'}2'
- One will provide all GMs holding key "5" with new logical keys: {2'}5, {1'}2'
- 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 would be shown in Figure 30.

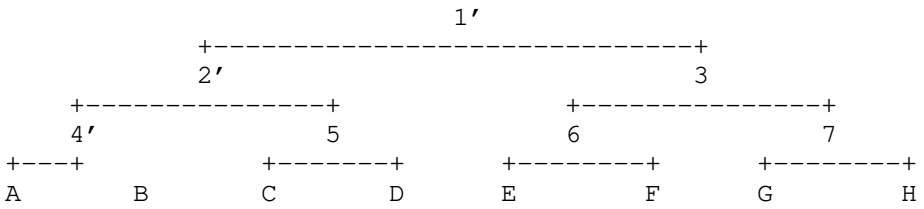


Figure 30: LKH tree after B has been excluded

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