Internet Engineering Task Force (IETF)
Internet-Draft
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: July 11, 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

January 8, 2020

Multiple Key Exchanges in IKEv2
draft-ietf-ipsecme-ikev2-multiple-ke-00

Abstract

This document describes how to extend the Internet Key Exchange Protocol Version 2 (IKEv2) to allow multiple key exchanges to take place while computing of a shared secret during a Security Association (SA) setup. The primary application of this feature in IKEv2 is the ability to perform one or more post-quantum key exchanges in conjunction with the classical (Elliptic Curve) Diffie-Hellman key exchange, so that the resulting shared key is resistant against quantum computer attacks. Another possible application is the ability to combine several key exchanges in situations when no single key exchange algorithm is trusted by both initiator and responder.

This document updates RFC7296 by renaming a tranform type 4 from "Diffie-Hellman Group (D-H)" to "Key Exchange Method (KE)" and renaming a field in the Key Exchange Payload from "Diffie-Hellman Group Num" to "Key Exchange Method". It also renames an IANA registry for this transform type from "Transform Type 4 - Diffie-Hellman Group Transform IDs" to "Transform Type 4 - Key Exchange Method Transform IDs". These changes generalize key exchange algorithms that can be used in IKEv2.

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 11, 2020.

Table of Contents

1. Introduction

1.1. Problem Description

Internet Key Exchange Protocol (IKEv2) as specified in [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 method to perform multiple successive key exchanges in IKEv2. It allows integration of QSC in IKEv2, while maintaining backwards compatibility, to derive a set of IKE keys that is resistant to quantum computer attacks. This extension 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 extension 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, this specification relies on 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 contains an additional 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 SA keys 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 defined in this document as providing a long term solution in upgrading the IKEv2 protocol to support post-quantum algorithms. A short term solution to make IKEv2 key exchange quantum secure is to use post-quantum pre-shared keys as discussed in [I-D.ietf-ipsecme-qr-ikev2].

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

1.3. Changes

RFC EDITOR PLEASE DELETE THIS SECTION.

Changes in this draft in each version iterations.

draft-ietf-ipsecme-ikev2-multiple-ke-00

- Draft name changed as result of WG adoption and generalization of the approach.
- New exchange IKE_FOLLOWUP_KE is defined for additional key exchanges performed after CREATE_CHILD_SA.
- Nonces are removed from all additional key exchanges.
- Clarification that IKE_INTERMEDIATE must be negotiated is added.

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 multiple key exchanges are 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 extension is driven by the following criteria:

1) Need for post-quantum cryptography in IPsec. Quantum computers might become feasible in the near future. 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 post-quantum algorithms to be introduced 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 extension 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 keys, 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) Federal Information Processing Standards (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.

12) Ability to use this method with multiple classical (EC)DH key exchanges. In some situations peers have no single mutually trusted key exchange algorithm (e.g., due to local policy restrictions). The ability to combine two (or more) key exchange methods in such a way that the resulting shared key depends on all of them allows peers to communicate in this situation.

3. Multiple Key Exchanges

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 be more specific, this document renames Transform Type 4 from "Diffie-Hellman Group (D-H)" to "Key Exchange Method (KE)" and renames a field in the Key Exchange Payload from "Diffie-Hellman Group Num" to "Key Exchange Method". The corresponding IANA registry is also renamed from "Diffie-Hellman Group Transform IDs" to "Key Exchange Method Transform IDs".

In order to support IKE fragmentation for additional key exchanges that may have long public keys, the proposed framework utilizes the IKE_INTERMEDIATE exchange defined 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 additional key exchanges may 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. However, it is also possible to use classical (EC)DH primitives for non post-quantum requirements.

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 exchange 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 additional key exchanges 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.

The fact that all Additional Key Exchange Transform Types share the same registry with Transform Type 4 allows additional key exchanges to be of any type - either post-quantum ones or classical (EC)DH ones. This approach allows any combination of defined key exchange methods to take place. This also allows performing a single post-quantum key exchange in the IKE_SA_INIT without additional key exchanges, provided that IP fragmentation is not an issue and that hybrid key exchange is not needed.

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 it is not concerned with possible fragmentation of the IKE_SA_INIT messages (either because the key exchange it selects is small enough not to fragment, or the initiator is confident that fragmentation will be handled either by IP fragmentation, or transport via TCP).

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, it responds with an IKE_SA_INIT response, and IKE continues as usual.

If the initiator desires to negotiate multiple key exchanges, or it needs IKE to handle any possible fragmentation, then the initiator 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 post-quantum) 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 (DH or ECDH), 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.

If the initiator includes any Additional Key Exchanges transform types into SA payload, it MUST also negotiate using IKE_INTERMEDIATE exchange as described in [I-D.ietf-ipsecme-ikev2-intermediate], by including INTERMEDIATE_EXCHANGE_SUPPORTED notification in the IKE_SA_INIT request message. If the responder agrees to use additional key exchanges, it MUST also return back this notification, thus confirming that IKE_INTERMEDIATE exchange is supported and will be used for transferring additional key exchange data. Presence of Additional Key Exchanges transform types in SA payload without negotiation of using IKE_INTERMEDIATE exchange MUST be treated as protocol error by both initiator and responder.

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 IKE_INTERMEDIATE exchange, as described in [I-D.ietf-ipsecme-ikev2-intermediate].

These exchanges are as follows:

```
Initiator                              Responder
-----------------------------------------------------------------------
HDR, SK {KEi(n)}     -->
                              <--      HDR, SK {KEr(n)}
```

The initiator sends key exchange data in the KEi(n) payload. This packet is protected with the current SK_ei/SK_ai keys.

On receiving this, the responder sends back key exchange payload KEr(n); again, this packet is protected with the current SK_er/SK_ar keys.

The former "Diffie-Hellman Group Num" (now called "Key Exchange Method") field in the KEi(n) and KEr(n) payloads MUST match the n-th negotiated additional key exchange. Note that the negotiated transform types (the encryption type, integrity 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) | Ni | Nr)
```

where KE(n) is the resulting shared secret of this key exchange, Ni and Nr are nonces from the IKE_SA_INIT 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 | Nr | 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].

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 to 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 new IKE_FOLLOWUP_KE exchanges that follows the CREATE_CHILD_SA exchange. The IKE_FOLLOWUP_KE exchange is introduced as a dedicated exchange type to transfer data of additional key exchanges following the key exchange performed in the CREATE_CHILD_SA. Its Exchange Type is <TBA by IANA>.

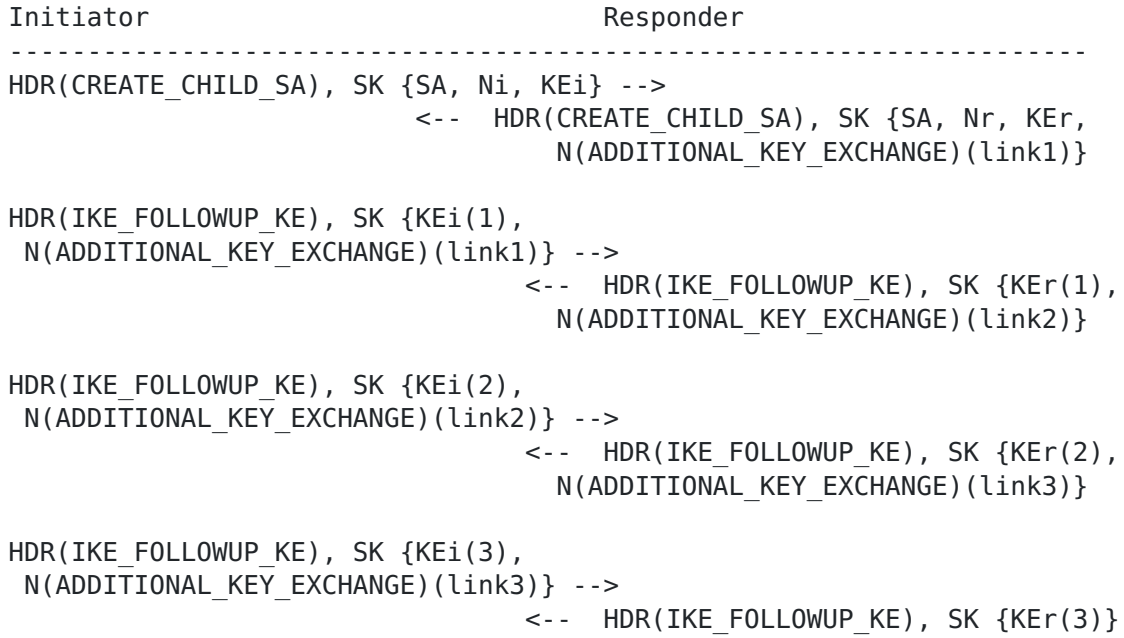
These key exchanges are performed in an order of the values of their transform types, so that key exchange negotiated using Transform Type n always precedes key exchange negotiated using Transform Type n + 1. Each IKE_FOLLOWUP_KE exchange MUST bear exactly one key exchange

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 in progress, it is essential to link the IKE_FOLLOWUP_KE 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 IKE_FOLLOWUP_KE 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.



The former "Diffie-Hellman Group Num" (now called "Key Exchange Method") field in the KEi(n) and KEr(n) payloads MUST match the n-th negotiated additional key exchange.

It is possible that due to some unexpected events (e.g. reboot) the initiator could forget that it is in the process of performing

additional key exchanges and never starts next IKE_FOLLOWUP_KE exchanges. The responder MUST handle this situation gracefully and delete the associated state if it doesn't receive the next expected IKE_FOLLOWUP_KE request after some reasonable period of time.

If responder receives IKE_FOLLOWUP_KE request containing ADDITIONAL_KEY_EXCHANGE notification and the content of this notify doesn't correspond to any active key exchange state the responder has, it MUST send back a new error type notification STATE_NOT_FOUND. This is a non-fatal error 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 the initiator receives this notification in response to IKE_FOLLOWUP_KE exchange performing additional key exchange, it 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 IKE_FOLLOWUP_KE 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 IKE_FOLLOWUP_KE exchanges) to retry the failed attempt. If the initiator continues to receive STATE_NOT_FOUND notifications after several retries, it MUST treat this situation as fatal error and delete IKE SA by sending a 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 when the CRETE_CHILD_SA is completed, they would be created only after the series of IKE_FOLLOWUP_KE 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 - this side MUST not initiate IKE_FOLLOWUP_KE 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 IKE_FOLLOWUP_KE exchanges is in progress). In this case, a 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) | ... KE(n))
```

In case of Child SA creation or rekey:

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

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

4. IANA Considerations

This document adds new exchange type into the "IKEv2 Exchange Types" registry:

<TBA> IKE_FOLLOWUP_KE

This document renames Transform Type 4 defined in "Transform Type Values" registry from "Diffie-Hellman Group (D-H)" to "Key Exchange Method (KE)".

This document renames IKEv2 registry "Transform Type 4 - Diffie-Hellman Group Transform IDs" to "Transform Type 4 - Key Exchange Method Transform IDs"

This document adds the following Transform Types to the "Transform Type Values" registry:

Type	Description	Used In
<TBA>	Additional Key Exchange 1	(optional in IKE, AH, ESP)
<TBA>	Additional Key Exchange 2	(optional in IKE, AH, ESP)
<TBA>	Additional Key Exchange 3	(optional in IKE, AH, ESP)
<TBA>	Additional Key Exchange 4	(optional in IKE, AH, ESP)
<TBA>	Additional Key Exchange 5	(optional in IKE, AH, ESP)
<TBA>	Additional Key Exchange 6	(optional in IKE, AH, ESP)
<TBA>	Additional Key Exchange 7	(optional in IKE, AH, ESP)

This document 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 KE(x) using an algorithm defined in Transform Type 2. While a quantum attacker may learn the value of KE(x), if this value is obtained by means of a classical key exchange, other KE(x) 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 man-in-the-middle (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-03 (work in progress), December 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, "Mixing Preshared Keys in IKEv2 for Post-quantum Resistance", draft-ietf-ipsecme-qr-ikev2-10 (work in progress), December 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.

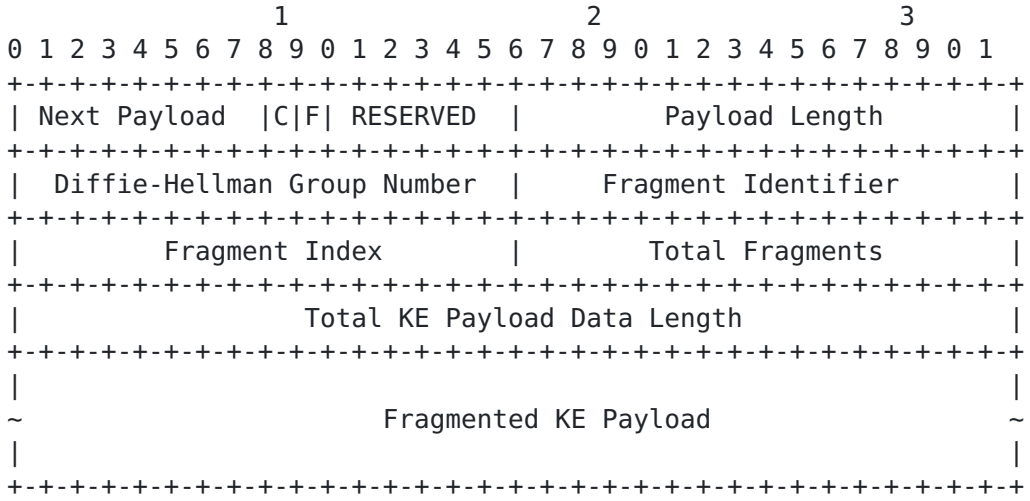
- Fragmentation

  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