Workgroup:
Internet Engineering Task Force (IETF)
Internet-Draft:
draft-ietf-ipsecme-ikev2-multiple-ke-06
Updates: 7296 (if approved)
Published: 13 June 2022
Intended Status: Standards Track
Expires: 15 December 2022

Authors:
C. Tjhai
Post-Quantum
M. Tomlinson
Post-Quantum
G. Bartlett
Quantum Secret
S. Fluhrer
Cisco Systems
D. Van Geest
ISARA Corporation
O. Garcia-Morchon
Philips
V. Smyslov
ELVIS-PLUS

Multiple Key Exchanges in IKEv2

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 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 transform 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

Copyright Notice

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, which we shall refer to as (EC)DH collectively, to establish a shared secret between an initiator and a responder. The security of the (EC)DH algorithms relies on the difficulty to solve a discrete logarithm problem in multiplicative (and respectively elliptic curve) groups 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 is 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 PQC 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 PQC algorithm to exchange data, in addition to the existing (EC)DH key exchange data. We believe that the feature of using more than one post-quantum algorithms 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 PQC 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 sizes larger than the standard maximum transmission unit (MTU) size, and therefore there could be issues with fragmentation at the IP layer. In order to allow using those larger payload sizes, this mechanism relies on the IKE_INTERMEDIATE exchange as specified in [RFC9242]. With this mechanism, we do an initial key exchange, using a smaller, possibly classical primitive, such as (EC)DH. 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 specified in [RFC8784].

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 addressing the threat of quantum computer only. It can also be used when all of the performed key exchanges are classical (EC)DH primitives, where for some reasons (e.g. policy requirements) it is essential to perform multiple of them.

This specification does not attempt to address key exchanges with KE payloads longer than 64k; the current IKE payload format does not allow such as possibility. At the time of writing, it appears likely that there are a number of key exchanges available that would not require such a requirement. However, if such a requirement is needed, [I-D.tjhai-ikev2-beyond-64k-limit] discusses approaches that should be taken to exchange huge payloads.

1.3. Changes

RFC EDITOR PLEASE DELETE THIS SECTION.

Changes in this draft in each version iterations.

draft-ietf-ipsecme-ikev2-multiple-ke-05

- Updated the reference to RFC9242.

- Editorial changes.

draft-ietf-ipsecme-ikev2-multiple-ke-04

- Introduction and initial sections are reorganized.

- More clarifications for error handling added.

- ASCII arts displaying SA payload are added.

- Clarification for handling multiple round trips key exchange methods added.

- DoS concerns added into Security Considerations section.

- Explicitly allow scenario when additional key exchanges are performed only after peers are authenticated.

draft-ietf-ipsecme-ikev2-multiple-ke-03

- More clarifications added.

- Figure illustrating initial exchange added.

- Minor editorial changes.

draft-ietf-ipsecme-ikev2-multiple-ke-02

- Added a reference on the handling of KE payloads larger than 64KB.

draft-ietf-ipsecme-ikev2-multiple-ke-01

- References are updated.

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 B. 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 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 (EC)DH is trusted against conventional (non-quantum) adversaries. A hybrid post-quantum algorithm to be introduced next to well-established primitives, since the overall security is at least as strong as each individual primitive.

3) Focus on post-quantum 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 confidentiality due to the urgency of this problem. This document does not address 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" nor does it 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 standard IKEv2 implementations as per [RFC7296] specification 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. Design Overview

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 may not be as secure as expected. A hybrid solution, when multiple key exchanges are performed and the calculated shared key depends on all of them, 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.

In order to be able to use IKE fragmentation [RFC7383] for those key exchanges that may have long public keys, this specification utilizes the IKE_INTERMEDIATE exchange defined in [RFC9242]. The initial IKE_INIT messages do not have any inherent fragmentation support within IKE; however that can include a relatively short KE payload. The additional key exchanges are performed using IKE_INTERMEDIATE messages; because these messages are encrypted, the standard IKE fragmentation mechanism is available.

In order to minimize communication overhead, only the key shares that are agreed to be used are actually exchanged. To negotiate additional key exchanges seven new Transform Types are defined. These transforms and Transform Type 4 share the same Transform IDs.

We assume that new Transform Type 4 identifiers will be assigned later to the various post-quantum key exchanges. We specifically do not make a distinction between classical (EC)DH 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".

The fact, that newly defined transforms share the same registry for possible Transform IDs with Transform Type 4, allows additional key exchanges to be of any type - either post-quantum or classical (EC)DH one. This approach allows any combination of the defined key exchange methods to take place. This also allows IKE peers to perform a single post-quantum key exchange in the IKE_SA_INIT without additional key exchanges, provided that the IP fragmentation is not an issue and that hybrid key exchange is not needed.

The SA payload in the IKE_SA_INIT message includes one or more newly defined transforms which represent the extra key exchange policy required by the initiator. The responder follows the usual IKEv2 negotiation rules: it selects a single transform of each type, and returns all of them in the IKE_SA_INIT response message.

Then, provided that additional key exchanges are negotiated, the initiator and the responder perform one or more IKE_INTERMEDIATE exchanges. Following that, the IKE_AUTH exchange authenticates peers and completes IKE SA establishment.

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

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

                         ...

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

<-- {IKE_AUTH} -->
```

Note that this document assumes, that each key exchange method requires one round trip and consumes exactly one IKE_INTERMEDIATE exchange. This assumption is valid for all classic key exchange methods defined so far and for all post-quantum methods currently known. For hypothetical future key exchange methods requiring multiple round trips to complete, a separate document should define how such methods are splitted into several IKE_INTERMEDIATE exchanges.

3.2. Protocol Details

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 usual, 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, 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 seven new transform types, namely Additional Key Exchange 1 (<TBA by IANA>), Additional Key Exchange 2 (<TBA by IANA>), Additional Key Exchange 3 (<TBA by IANA>), Additional Key Exchange 4 (<TBA by IANA>), Additional Key Exchange 5 (<TBA by IANA>), Additional Key Exchange 6 (<TBA by IANA>) and Additional Key Exchange 7 (<TBA by IANA>) are defined. They are collectively called Additional Key Exchange transforms in this document and have slightly different semantics than the existing IKEv2 transform types. They are interpreted as an indication of additional key exchange methods that peers agree to perform in a series of IKE_INTERMEDIATE exchanges following the

IKE_SA_INIT exchange. The allowed transform IDs for these transform types are the same as the IDs for Transform Type 4, so they all share a single IANA registry for transform IDs.

Key exchange method negotiated via Transform Type 4 always takes place in the IKE_SA_INIT exchange, as defined in [RFC7296]. Additional key exchanges negotiated via newly defined transforms MUST take place in a series of IKE_INTERMEDIATE exchanges following the IKE_SA_INIT exchange, performed in an order of the values of their transform types, so that key exchange negotiated using Additional Key Exchange i always precedes that of Additional Key Exchange i+1. Each additional key exchange method MUST be fully completed before the next one is started.

Note that with this semantics, Additional Key Exchange transforms are not associated with any particular type of key exchange and do not have any specific per transform type transform IDs IANA registry. Instead they all share a single registry for transform IDs, namely "Key Exchange Method Transform IDs", which are also shared by Transform Type 4. All 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 (EC)DH key exchange methods, however it is not a requirement. Additional key exchange methods are proposed using Additional Key Exchange transform types. All of these transform types are optional, the initiator is free to select any of them for proposing additional key exchange methods. Consequently, if none of the Additional Key Exchange transforms is included in the proposal, then this proposal indicates performing standard IKEv2, as defined in [RFC7296]. On the other hand, if the initiator includes any Additional Key Exchange transform in the proposal, the responder MUST select one of the algorithms proposed using this type. Note that this is not a new requirement, but this behaviour is already specified in Section 2.7 of [RFC7296]. A transform ID NONE MAY be added to those transform types which contain key exchange methods that the initiator believes is optional according to its local policy.

The responder performs negotiation using the standard IKEv2 procedure described in Section 3.3 of [RFC7296]. However, for the Additional Key Exchange types, the responder's choice MUST NOT contain duplicated algorithms, except for the transform ID of NONE. An algorithm is represented as a transform, in some cases the transform could include a set of associated attributes that define details of the algorithm. In this case, two transforms can be the same, but the attributes must be different. Additionally, the order of the attributes does not affect the equality of the algorithm, so two transforms (ID=alg1,ATTR1=attr1,ATTR2=attr2) and (ID=alg1,ATTR2=attr2,ATTR1=attr1) define the same algorithm.

If the responder selects NONE for some Additional Key Exchange types (provided they are proposed by the initiator), then the corresponding IKE_INTERMEDIATE exchanges MUST NOT take place. Therefore if the initiator includes NONE in all of the Additional

Key Exchange transforms and the responder selects this value for all of them, then no IKE_INTERMEDIATE messages performing additional key exchanges will take place between the peers. Note that the IKE_INTERMEDIATE exchanges may still take place for other purposes.

The initiator MAY propose non-consecutive Additional Key Exchange transforms, for example proposing Additional Key Exchange 2 and Additional Key Exchange 5 transforms only. The responder MUST treat all of the omitted Additional Key Exchange transforms as if they are proposed with Transform ID NONE.

Below is an example of the SA payload in the initiator's IKE_SA_INIT request message. Here we use an abbreviation AKEi to denote the i-th Additional Key Exchange transform defined in this document, and an abbreviation KE for the Key Exchange transform, that this document renames from the Diffie-Hellman Group transform. We also use some not-yet defined Transform IDs, namely PQ_KEM_1, PQ_KEM_2 and PQ_KEM_3 to denote some popular post-quantum key exchange methods.

```
SA Payload
   |
   +--- Proposal #1 ( Proto ID = IKE(1), SPI size = 8,
         |            9 transforms,      SPI = 0x35a1d6f22564f89d )
         |
         +-- Transform ENCR ( ID = ENCR_AES_GCM_16 )
         |     +-- Attribute ( Key Length = 256 )
         |
         +-- Transform KE ( ID = 4096-bit MODP Group )
         |
         +-- Transform PRF ( ID = PRF_HMAC_SHA2_256 )
         |
         +-- Transform AKE2 ( ID = PQ_KEM_1 )
         |
         +-- Transform AKE2 ( ID = PQ_KEM_2 )
         |
         +-- Transform AKE3 ( ID = PQ_KEM_1 )
         |
         +-- Transform AKE3 ( ID = PQ_KEM_2 )
         |
         +-- Transform AKE5 ( ID = PQ_KEM_3 )
         |
         +-- Transform AKE5 ( ID = NONE )
```

In this example, the initiator proposes to perform initial key exchange using 4096-bit MODP group followed by two mandatory additional key exchanges using PQ_KEM_1 and PQ_KEM_2 methods in any order, then followed by additional key exchange using PQ_KEM_3 method that may be omitted.

The responder might return the following SA payload, indicating that it agrees to perform two additional key exchanges PQ_KEM_2 followed by PQ_KEM_1 and does not want to perform PQ_KEM_3 additionally.

```
SA Payload
   |
   +--- Proposal #1 ( Proto ID = IKE(1), SPI size = 8,
         |            6 transforms,      SPI = 0x8df52b331a196e7b )
         |
         +-- Transform ENCR ( ID = ENCR_AES_GCM_16 )
         |     +-- Attribute ( Key Length = 256 )
         |
         +-- Transform KE ( ID = 4096-bit MODP Group )
         |
         +-- Transform PRF ( ID = PRF_HMAC_SHA2_256 )
         |
         +-- Transform AKE2 ( ID = PQ_KEM_2 )
         |
         +-- Transform AKE3 ( ID = PQ_KEM_1 )
         |
         +-- Transform AKE5 ( ID = NONE )
```

If the initiator includes any Additional Key Exchange transform types into the SA payload in the IKE_SA_INIT exchange request message, then it MUST also negotiate the use of the IKE_INTERMEDIATE exchange as described in [RFC9242], by including INTERMEDIATE_EXCHANGE_SUPPORTED notification in the same message. If the responder agrees to use additional key exchanges while establishing initial IKE SA, it MUST also return this notification in the IKE_SA_INIT response message, thus confirming that IKE_INTERMEDIATE exchange is supported and will be used for transferring additional key exchange data. If the IKE_INTERMEDIATE exchange is not negotiated, then the peers MUST treat any Additional Key Exchange transforms in the IKE_SA_INIT exchange messages as unknown transform types and skip the proposals they appear in. If no other proposals are present in the SA payload, the peers will proceed as if no proposal is chosen (i.e. the responder will send NO_PROPOSAL_CHOSEN notification).

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

3.2.2. IKE_INTERMEDIATE Round: Additional Key Exchanges

For each additional key exchange agreed to in the IKE_SA_INIT exchange, the initiator and the responder perform IKE_INTERMEDIATE exchange, as described in [RFC9242].

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

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

KEi(n) denotes the n-th IKE_INTERMEDIATE KE payload from the initiator and the integer n is sequential starting from 1.

On receiving this, the responder sends back key exchange payload KEr(n), which denotes the n-th IKE_INTERMEDIATE KE payload from the responder. As before, this message 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.

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

```
          SKEYSEED(n) = prf(SK_d(n-1), SK(n) | Ni | Nr)
```

where SK(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 have not already been any IKE_INTERMEDIATE exchanges). The other keying materials 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 these updated key values in the next exchange (IKE_INTERMEDIATE or IKE_AUTH).

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 [RFC9242].

3.2.4. CREATE_CHILD_SA Exchange

The CREATE_CHILD_SA exchange is used in IKEv2 for the purposes 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-Hellman key exchange to add a fresh entropy into the session keys. In case of IKE SA rekey, the key exchange is mandatory. Peers supporting this specification may want to use multiple key exchanges in these situations.

Using multiple key exchanges with CREATE_CHILD_SA exchange is negotiated similarly as in the initial IKE exchange, see Section 3.2.1. If the initiator includes any Additional Key Exchange 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 for transferring data of additional key exchanges following the key exchange performed in the CREATE_CHILD_SA. Its Exchange Type is <TBA by IANA>.

Key exchange negotiated via Transform Type 4 always takes place in the CREATE_CHILD_SA exchange, as per IKEv2 specification. Additional 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 additional key exchange method MUST be fully completed before the next one is started. Note, that this document assumes, that each key exchange method consumes exactly one IKE_FOLLOWUP_KE exchange. For the methods requiring multiple round trips, a separate document should define how such methods are splitted into several IKE_FOLLOWUP_KE exchanges.

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 the next IKE_FOLLOWUP_KE exchange is expected, filling it with some data that would allow linking current exchange to the next one. The initiator MUST send back this notification intact in the request message of the next IKE_FOLLOWUP_KE exchange.

Below is an example of CREATE_CHILD_SA exchange followed by 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(IKE_FOLLOWUP_KE), SK {KEi(1),
 N(ADDITIONAL_KEY_EXCHANGE)(link1)} -->
                               <--  HDR(IKE_FOLLOWUP_KE), SK {KEr(1),
                                   N(ADDITIONAL_KEY_EXCHANGE)(link2)}

HDR(IKE_FOLLOWUP_KE), SK {KEi(2),
 N(ADDITIONAL_KEY_EXCHANGE)(link2)} -->
                               <--  HDR(IKE_FOLLOWUP_KE), SK {KEr(2),
                                   N(ADDITIONAL_KEY_EXCHANGE)(link3)}

HDR(IKE_FOLLOWUP_KE), SK {KEi(3),
 N(ADDITIONAL_KEY_EXCHANGE)(link3)} -->
                               <--  HDR(IKE_FOLLOWUP_KE), SK {KEr(3)}
```

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 may lose its state and forget that it is in the process of performing additional key exchanges and thus never start the remaining IKE_FOLLOWUP_KE exchanges. The responder MUST handle this situation gracefully and delete the associated state if it does not 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 does not 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 the 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 which may eventually be 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 a 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] describe 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 are used in the colliding CREATE_CHILD_SA exchanges. The IKE SA (or pair of Child SAs) that is created by the exchange in which the smallest nonce is used should be deleted by the initiator of this exchange.

With multiple key exchanges, the SAs are not yet created when the CREATE_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 are negotiated in the CREATE_CHILD_SA exchange in which the smallest nonce is used, then because there is nothing to delete yet, the initiator of this exchange just stops the rekeying process and it MUST NOT initiate the IKE_FOLLOWUP_KE 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 the CREATE_CHILD_SA message requesting rekey of 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 a request.

If multiple key exchanges are 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, SK(0) | Ni | Nr | SK(1) | ... SK(n))
```

In case of Child SA creation or rekey:

```
KEYMAT = prf+ (SK_d, SK(0) | Ni | Nr | SK(1) |  ... SK(n))
```

In both cases, SK_d is from the existing IKE SA; SK(0), Ni, Nr are the shared key and nonces from the CREATE_CHILD_SA respectively; SK(1)...SK(n) are the shared keys from additional key exchanges.

3.2.5. Interaction with Childless IKE SA

It is also possible to establish a fully post-quantum secure IKE SAs from additional key exchanges without using IKE_INTERMEDIATE exchanges. In this case, the IKE SA created from IKE_SA_INIT exchange can be immediately rekeyed with CREATE_CHILD_SA using additional key exchanges where IKE_FOLLOWUP_KE messages are used to carry the key exchange payload. If classical key exchange method is used in the IKE_SA_INIT message, the very first Child SA created in IKE_AUTH will offer no resistance against the quantum threats. Consequently, if the peers' local policy requires that all Child SAs should be fully-protected, then the peers can avoid creating the very first Child SA by adopting [RFC6023]. In this case, the peers exchange CHILDLESS_IKEV2_SUPPORTED notification in the IKE_SA_INIT exchange and a fully-protected Child SA can be created with CREATE_CHILD_SA using additional key exchanges.

Note that if the initial IKE SA is used to transfer sensitive information, then this information will not be protected using the additional key exchanges, which may use post-quantum algorithms. Consider G-IKEv2 protocol [I-D.ietf-ipsecme-g-ikev2] where the cryptographic materials are sent from the group controller to the group members when the initial IKE SA is created. In this arrangement, the peers will have to use post-quantum algorithm in Transform Type 4 in order to mitigate the risk of quantum attack.

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

4.1. Additional Changes

RFC EDITOR PLEASE DELETE THIS SECTION.

It is assumed that RFCXXXX refers to this specification.

- Add a reference to RFCXXXX in the "Transform Type 4 - Diffie-Hellman Group Transform IDs" registry.

- Replace the note on "Transform Type 4 - Diffie-Hellman Group Transform IDs" registry with: This registry was originally named "Transform Type 4 - Diffie-Hellman Group Transform IDs" and was renamed to its current name by [RFCXXXX]. It has been referenced in its original name in a number of RFCs prior to [RFCXXXX]. To find out requirement levels for Key Exchange Methods for IKEv2, see [RFC8247].

- Add this note to "Transform Type Values" registry: Transform Type "Transform Type 4 - Key Exchange Method Transform IDs" was originally named "Transform Type 4 - Diffie-Hellman Group Transform IDs" and was renamed to its current name by [RFCXXXX]. It has been referenced in its original name in a number of RFCs prior to [RFCXXXX]. All "Additional Key Exchange" entries use the same "Transform Type 4 - Key Exchange Method Transform IDs" as the "Key Exchange Method (KE)".

- Append this note to "Transform Type 4 - Diffie-Hellman Group Transform IDs" registry: All "Additional Key Exchange" entries use these values as the "Key Exchange Method (KE)".

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 MUST 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 SK(x) using an algorithm defined in Transform Type 2. While a quantum attacker may learn the value of SK(x), if this value is obtained by means of a classical key exchange, other SK(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 exchanged 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 secure, the other algorithms are not. Moreover, in implementations where scalability is a requirement, the pre-shared key method may not be suitable. Post-quantum authenticity may be provided by using a post-quantum digital signature and several of them have been 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, post-quantum authentication methods are not specified in this document, but are planned to be incorporated in due course.

Secondly, 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 less urgency for post-quantum authenticity compared to post-quantum confidentiality.

Performing multiple key exchanges while establishing IKEv2 SA increases the responder's susceptibility to DoS attacks, because of an increased amount of resources needed before the initiator is authenticated. This is especially true for post-quantum key exchange methods, where many of them are more memory and/or CPU intensive than the classical counterparts.

Responders may consider recommendations from [RFC8019] to deal with increased DoS attack susceptibility. It is also possible that the responder only agrees to create initial IKE SA without performing additional key exchanges, provided the initiator includes such an option in its proposals. Then peers immediately rekey the initial IKE SA with the CREATE_CHILD_SA exchange and additional key exchanges performed via the IKE_FOLLOWUP_KE exchanges. In this case, at the point when resource-intensive operations are required, the peers have already authenticated each other. However, in the context of hybrid post-quantum key exchange this scenario would leave the initial IKE SA (and initial Child SA if it is created) unprotected against quantum computers. Nevertheless the rekeyed IKE SA (and Child SAs that will be created over it) will have a full protection. This is similar to the scenario described in [RFC8784]. Depending on peers' policy, this scenario may or may not be appropriate.

6. Acknowledgements

The authors would like to thank 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. Thanks to Paul Wouters for reviewing the document.

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>.

[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>.

[RFC9242] Smyslov, V., "Intermediate Exchange in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 9242, DOI 10.17487/RFC9242, May 2022, <https://www.rfc-editor.org/info/rfc9242>.

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-g-ikev2] Smyslov, V. and B. Weis, "Group Key Management using IKEv2", Work in Progress, Internet-Draft, draft-ietf-ipsecme-g-ikev2-06, 6 April 2022,

<https://www.ietf.org/internet-drafts/draft-ietf-ipsecme-g-ikev2-06.txt>.

[I-D.tjhai-ikev2-beyond-64k-limit] Tjhai, C., Heider, T., and V. Smyslov, "Beyond 64KB Limit of IKEv2 Payloads", Work in Progress, Internet-Draft, draft-tjhai-ikev2-beyond-64k-limit-02, 28 January 2022, <https://www.ietf.org/archive/id/draft-tjhai-ikev2-beyond-64k-limit-02.txt>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC8784] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smyslov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <https://www.rfc-editor.org/info/rfc8784>.

Appendix A. Sample Multiple Key Exchanges

This appendix shows some examples of multiple key exchanges. These examples are purely for information purposes and they describe some message flow scenarios that may occur in establishing an IKE or CHILD SA. Note that some payloads that are not relevant to multiple key exchanges may be omitted for brevity.

A.1. IKE_INTERMEDIATE Exchanges Carrying Additional Key Exchange Payloads

The exchanges below show that the initiator proposes the use of additional key exchanges to establish an IKE SA. The initiator proposes three sets of additional key exchanges and all of which are optional. So the responder can choose NONE for some or all of the additional exchanges if the proposed key exchange methods are not supported or for whatever reasons the responder decides not to perform the additional key exchange.

```
Initiator                     Responder
------------------------------------------------------------------------
HDR(IKE_SA_INIT), SAi1(.. AKE*...), --->
KEi(Curve25519), Ni, N(IKEV2_FRAG_SUPPORTED),
N(INTERMEDIATE_EXCHANGE_SUPPORTED)
    Proposal #1
    Transform ECR (ID = ENCR_AES_GCM_16,
                    256-bit key)
    Transform PRF (ID = PRF_HMAC_SHA2_512)
    Transform KE (ID = Curve25519)
    Transform AKE1 (ID = PQ_KEM_1)
    Transform AKE1 (ID = PQ_KEM_2)
    Transform AKE1 (ID = NONE)
    Transform AKE2 (ID = PQ_KEM_3)
    Transform AKE2 (ID = PQ_KEM_4)
    Transform AKE2 (ID = NONE)
    Transform AKE3 (ID = PQ_KEM_5)
    Transform AKE3 (ID = PQ_KEM_6)
    Transform AKE3 (ID = NONE)
                   <--- HDR(IKE_SA_INIT), SAr1(.. AKE*...),
                        KEr(Curve25519), Nr, N(IKEV2_FRAG_SUPPORTED),
                        N(INTERMEDIATE_EXCHANGE_SUPPORTED)
                        Proposal #1
                          Transform ECR (ID = ENCR_AES_GCM_16,
                                         256-bit key)
                          Transform PRF (ID = PRF_HMAC_SHA2_512)
                          Transform KE (ID = Curve25519)
                          Transform AKE1 (ID = PQ_KEM_2)
                          Transform AKE2 (ID = NONE)
                          Transform AKE3 (ID = PQ_KEM_5)

HDR(IKE_INTERMEDIATE), SK {KEi(1)(PQ_KEM_2)} -->
                   <--- HDR(IKE_INTERMEDIATE), SK {KEr(1)(PQ_KEM_2)}
HDR(IKE_INTERMEDIATE), SK {KEi(2)(PQ_KEM_5)} -->
                   <--- HDR(IKE_INTERMEDIATE), SK {KEr(2)(PQ_KEM_5)}

HDR(IKE_AUTH), SK{ IDi, AUTH, SAi2, TSi, TSr } --->
                      <--- HDR(IKE_AUTH), SK{ IDr, AUTH, SAr2,
                           TSi, TSr }
```

In this particular example, the responder chooses to perform two additional key exchanges. It selects PQ_KEM_2, NONE and PQ_KEM_5 for the first, second and third additional key exchanges respectively. As per [RFC7296] specification, a set of keying materials are derived, in particular SK_d, SK_a[i/r], SK_e[i/r]. Both peers then perform an IKE_INTERMEDIATE exchange carrying PQ_KEM_2 payload which is protected with SK_e[i/r] and SK_a[i/r] keys. After the completion of this IKE_INTERMEDIATE exchange, the SKEYSEED is updated using SK(1), which is the PQ_KEM_2 shared secret, as follows

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

The updated SKEYSEED value is then used to derive the following keying materials

```
{SK_d(1) | SK_ai(1) | SK_ar(1) | SK_ei(1) | SK_er(1) | SK_pi(1) |
 SK_pr(1)} = prf+ (SKEYSEED(1), Ni | Nr | SPIi | SPIr).
```

As per [RFC9242] specification, both peers compute IntAuth_i1 and IntAuth_r1 using the SK_pi(1) and SK_pr(1) keys respectively. These values are required in the IKE_AUTH phase of the exchange.

In the next IKE_INTERMEDIATE exchange, the peers use SK_e[i/r](1) and SK_a[i/r](1) keys to protect the PQ_KEM_5 payload. After completing this exchange, keying materials are updated as below

```
SKEYSEED(2) = prf(SK_d(1), SK(2) | Ni | Nr)
{SK_d(2) | SK_ai(2) | SK_ar(2) | SK_ei(2) | SK_er(2) | SK_pi(2) |
    SK_pr(2)} = prf+ (SKEYSEED(2), Ni | Nr | SPIi | SPIr)
```

where SK(2) is the shared secret from the third additional key exchange, i.e. PQ_KEM_5. Both peers then computes the values of IntAuth_[i/r]2 using the SK_p[i/r](2) keys.

After the completion of the second IKE_INTERMEDIATE exchange, both peers continue to the IKE_AUTH exchange phase. As defined in [RFC9242], the values IntAuth_[i/r]2 are used to compute IntAuth which in turn is used to calculate the payload to be signed or MACed, i.e. InitiatorSignedOctets and ResponderSignedOctets.

A.2. No Additional Key Exchange Used

The initiator proposes two sets of optional additional key exchanges, but the responder does not support any of them. The responder chooses NONE for each set and consequently, IKE_INTERMEDIATE exchange does not takes place and the exchange proceeds to IKE_AUTH phase. The resulting keying materials are the same as those derived with [RFC7296].

```
Initiator                     Responder
-----------------------------------------------------------------------
HDR(IKE_SA_INIT), SAi1(.. AKE*...), --->
KEi(Curve25519), Ni, N(IKEV2_FRAG_SUPPORTED),
N(INTERMEDIATE_EXCHANGE_SUPPORTED)
  Proposal #1
    Transform ECR (ID = ENCR_AES_GCM_16,
                   256-bit key)
    Transform PRF (ID = PRF_HMAC_SHA2_512)
    Transform KE (ID = Curve25519)
    Transform AKE1 (ID = PQ_KEM_1)
    Transform AKE1 (ID = PQ_KEM_2)
    Transform AKE1 (ID = NONE)
    Transform AKE2 (ID = PQ_KEM_3)
    Transform AKE2 (ID = PQ_KEM_4)
    Transform AKE2 (ID = NONE)
                    <--- HDR(IKE_SA_INIT), SAr1(.. AKE*...),
                         KEr(Curve25519), Nr, N(IKEV2_FRAG_SUPPORTED),
                         N(INTERMEDIATE_EXCHANGE_SUPPORTED)
                           Proposal #1
                             Transform ECR (ID = ENCR_AES_GCM_16,
                                            256-bit key)
                             Transform PRF (ID = PRF_HMAC_SHA2_512)
                             Transform KE (ID = Curve25519)
                             Transform AKE1 (ID = NONE)
                             Transform AKE2 (ID = NONE)

HDR(IKE_AUTH), SK{ IDi, AUTH, SAi2, TSi, TSr } --->
                       <--- HDR(IKE_AUTH), SK{ IDr, AUTH, SAr2,
                            TSi, TSr }
```

A.3. Additional Key Exchange in the CREATE_CHILD_SA Exchange only

The exchanges below show that the initiator does not propose the use of additional key exchanges to establish an IKE SA, but they are required in order to establish a Child SA. In order to establish a fully quantum-resistant IPsec SA, both peers include CHILDLESS_IKEV2_SUPPORTED notification in their exchange so that the first Child SA is not created in IKE_AUTH, but instead the IKE SA is immediately rekeyed using CREATED_CHILD_SA. Any Child SA will have to be created via subsequent CREATED_CHILD_SA exchange.

```
Initiator                     Responder
-----------------------------------------------------------------------
HDR(IKE_SA_INIT), SAi1, --->
KEi(Curve25519), Ni, N(IKEV2_FRAG_SUPPORTED),
N(CHILDLESS_IKEV2_SUPPORTED)
                     <--- HDR(IKE_SA_INIT), SAr1,
                          KEr(Curve25519), Nr, N(IKEV2_FRAG_SUPPORTED),
                          N(CHILDLESS_IKEV2_SUPPORTED)
HDR(IKE_AUTH), SK{ IDi, AUTH  } --->
                       <--- HDR(IKE_AUTH), SK{ IDr, AUTH }
HDR(CREATE_CHILD_SA), SK{ SAi(.. AKE*...), Ni, KEi(Curve25519) } --->
  Proposal #1
    Transform ECR (ID = ENCR_AES_GCM_16,
                   256-bit key)
    Transform PRF (ID = PRF_HMAC_SHA2_512)
    Transform KE (ID = Curve25519)
    Transform AKE1 (ID = PQ_KEM_1)
    Transform AKE1 (ID = PQ_KEM_2)
    Transform AKE2 (ID = PQ_KEM_5)
    Transform AKE2 (ID = PQ_KEM_6)
    Transform AKE2 (ID = NONE)
                       <--- HDR(CREATE_CHILD_SA), SK{ SAr(.. AKE*...),
                            Nr, KEr(Curve25519),
                            N(ADDITIONAL_KEY_EXCHANGE)(link1) }
                              Proposal #1
                                Transform ECR (ID = ENCR_AES_GCM_16,
                                               256-bit key)
                                Transform PRF (ID = PRF_HMAC_SHA2_512)
                                Transform KE (ID = Curve25519)
                                Transform AKE1 (ID = PQ_KEM_2)
                                Transform AKE2 (ID = PQ_KEM_5)

HDR(IKE_FOLLOWUP_KE), SK{ KEi(1)(PQ_KEM_2), --->
N(ADDITIONAL_KEY_EXCHANGE)(link1) }
                      <--- HDR(IKE_FOLLOWUP_KE), SK{ KEr(1)(PQ_KEM_2),
                            N(ADDITIONAL_KEY_EXCHANGE)(link2) }
HDR(IKE_FOLLOWUP_KE), SK{ KEi(2)(PQ_KEM_5), --->
N(ADDITIONAL_KEY_EXCHANGE)(link2) }
                      <--- HDR(IKE_FOLLOWUP_KE), SK{ KEr(2)(PQ_KEM_5) }
```

A.4. Not Matching Proposal for Additional Key Exchanges

The initiator proposes the combination of PQ_KEM_1, PQ_KEM_2, PQ_KEM_3, and PQ_KEM_4 as the additional key exchanges. The initiator indicates that either PQ_KEM_1 or PQ_KEM_2 must be used to establish a security association, but Additional Key Exchange 2 is optional so the responder can either select PQ_KEM_3 or PQ_KEM_4 or omit this key exchange by selecting NONE. The responder, although supports the optional PQ_KEM_3 and PQ_KEM_4 methods, does not support either PQ_KEM_1 or PQ_KEM_2 mandatory method and therefore responds with NO_PROPOSAL_CHOSEN notification.

```
Initiator                     Responder
-----------------------------------------------------------------------
HDR(IKE_SA_INIT), SAi1(.. AKE*...), --->
KEi(Curve25519), Ni, N(IKEV2_FRAG_SUPPORTED),
N(INTERMEDIATE_EXCHANGE_SUPPORTED)
  Proposal #1
    Transform ECR (ID = ENCR_AES_GCM_16,
                   256-bit key)
    Transform PRF (ID = PRF_HMAC_SHA2_512)
    Transform KE (ID = Curve25519)
    Transform AKE1 (ID = PQ_KEM_1)
    Transform AKE1 (ID = PQ_KEM_2)
    Transform AKE2 (ID = PQ_KEM_3)
    Transform AKE2 (ID = PQ_KEM_4)
    Transform AKE2 (ID = NONE)
                         <--- HDR(IKE_SA_INIT), N(NO_PROPOSAL_CHOSEN)
```

Appendix B. 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.

- 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.

```
                 1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Payload  |C|F| RESERVED  |         Payload Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Diffie-Hellman Group Number  |     Fragment Identifier       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Fragment Index        |        Total Fragments        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Total KE Payload Data Length                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                       Fragmented KE Payload                   ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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, and using the generic IKEv2 fragmentation protocol [RFC7383] solve the issue.

*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
Quantum Secret

Email: graham.ietf@gmail.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