Internet Engineering Task Force
Internet-Draft
Intended Status: Informational
Expires: July 19, 2018

C. Tjhai
M. Tomlinson
Post-Quantum
G. Bartlett
S. Fluhrer
Cisco Systems
D. Van Geest
ISARA Corporation
Z. Zhang
Onboard Security
O. Garcia-Morchon
Philips
January 15, 2018

Framework to Integrate Post-quantum Key Exchanges into Internet Key Exchange Protocol Version 2 (IKEv2) draft-tjhai-ipsecme-hybrid-qske-ikev2-01

Abstract

This document describes how to extend Internet Key Exchange Protocol Version 2 (IKEv2) so that the shared secret exchanged between peers has resistance against quantum computer attacks. The basic idea is to exchange one or more post quantum key exchange payloads in conjunction with the existing (Elliptic Curve) Diffie-Hellman payload. This document also addresses the challenge of fragmentation as a result of sending large post quantum key exchange data in the first pair of message of the initial exchange phase.

Table of Contents

1. Introduction 3
1.1. Problem Description 3
1.2. Proposed Extension 3
1.3. Changes 4
1.4. Document organization 4
2. Design criteria 5
3. The Framework of Hybrid Post-quantum Key Exchange 6
3.1. Overall design 6
3.2. Overall Protocol 7
3.2.1. First Protocol Round 7
3.2.2. Second Protocol Round 10
3.2.3. Child SA Negotiation 11
3.3. Computation of a Post-Quantum Shared Secret 12
3.4. Post-Quantum Group Transform Type and Group Identifiers . 12
3.5. Hybrid Group Negotiation 13
3.5.1. Protocol for the Initiator 14
3.5.2. Protocol from the Responder 17
3.6. Fragmentation Support 19
3.6.1. Fragmentation Problem Description 19
3.6.2. Fragmentation Solution 19
3.6.2.1. Fragmentation Pointer Payload 19
3.6.2.2. Fragmentation Body Payload 20
3.6.2.3. Fragmentation Semantics 23
3.6.2.4. IKE AUTH Processing 24
3.6.2.5. Design Rationale 25
3.7. Protection against Downgrade Attacks 25
4. Alternative Design 28
5. Security considerations 31
6. References 33
Acknowledgements 34

Authors' Addresses 35

1. Introduction

1.1. Problem Description

Internet Key Exchange Protocol (IKEv2) as specified in RFC 7296 [RFC7296] uses the Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH) algorithm to establish a shared secret between an initiator and a responder. The security of the DH and ECDH algorithms relies on the difficulty to solve a discrete logarithm problem in multiplicative and elliptic curve groups respectively when the order of the group parameter is large enough. While solving such a problem remains difficult with current computing power, it is believed that general purpose quantum computers will be able to solve this problem, implying that the security of IKEv2 is compromised. There are, however, a number of cryptosystems that are conjectured to be resistant against quantum computer attack. This family of cryptosystems are known as post-quantum cryptography (PQC). It is sometime also referred to as quantum-safe cryptography (QSC) or quantum-resistant cryptography (QRC).

1.2. Proposed Extension

This document describes a framework to integrate QSC for IKEv2, whilst maintaining backwards compatibility, to exchange a shared secret such that it has resistance to quantum computer attacks. Our framework allows the negotiation of one or more QSC algorithm to exchange data, in addition to the existing DH or ECDH key exchange data. We believe that the feature of using more than one post quantum algorithm is important as many of these algorithms are relatively new and there may be a need to hedge the security risk with multiple key exchange data from several distinct QSC algorithms.

The secrets established from each key exchange are combined in a way such that should the post quantum secrets not be present, the derived shared secret is equivalent to that of the standard IKEv2; on the other hand, a post quantum shared secret is obtained if both classical and post quantum key exchange data are present. This framework also applies to key exchanges in IKE Security Associations (SAs) for Encapsulating Security Payload (ESP) [ESP] or Authentication Header (AH) [AH], i.e. Child SAs, in order to provide a stronger guarantee of forward security.

One of the key challenges in this framework is fragmentation handling

during the first message pair of the initial exchange phase, i.e. IKE_SA_INIT. Almost all of the known PQC algorithms to date have key exchange data size that exceeds 1K octets. When transmitted alongside other payloads, the total payload size could easily exceed the common maximum transmission unit (MTU) size of 1500 octets, and hence this may lead to IP level fragmentation. While IKEv2 has a mechanism to handle fragmentation [RFC7383], it is applicable to messages exchanged after IKE_SA_INIT. Of course, fragmentation will not be an issue if messages are sent over TCP [RFC8229]; however, we believe that a UDP-based solution will also be required. This document describes a simple mechanism to fragment IKE_SA_INIT messages, which also allows exchanges for payload larger than 65,535 octets.

Note that readers should consider the approach in this document as providing a long term solution in upgrading the IKEv2 protocol to support post quantum algorithms. A short term solution to make IKEv2 key exchange quantum secure is to use post quantum pre-shared keys as discussed in [FMKS].

1.3. Changes

Changes in this draft in each version iterations.

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

- We added a feature to allow more than one post quantum key exchange algorithms to be negotiated and used to exchange a post-quantum shared secret.

- Instead of relying on TCP encapsulation to deal with IP level fragmentation, we introduced a new key exchange payload that can be sent as multiple fragments within IKE_SA_INIT message.

1.4. Document organization

The remainder of this document is organized as follows. Section 2 summarizes design criteria. Section 3 describes how post-quantum key exchange is performed between two IKE peers, how keying materials are derived and how downgrade attack is prevented. This section also specifies we handle fragmentation in IKE_SA_INIT exchanges. A number of alternative designs to Section 3, which we have considered but not adopted, are discussed in Section 4. Lastly, Section 5 discusses security considerations.

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [RFC2119].

2. Design criteria

The design of the proposed post-quantum IKEv2 is driven by the following criteria:

1) Need for post-quantum cryptography in IPsec. Quantum computers might become feasible in the next 5-10 years. If current Internet communications are monitored and recorded today (D), the communications could be decrypted as soon as a quantum-computer is available (e.g., year Q) if key negotiation only relies on non post-quantum primitives. This is a high threat for any information that must remain confidential for a long period of time T > Q-D. The need is obvious if we assume that Q is 2040, D is 2020, and T is 30 years. Such a value of T is typical in classified or healthcare data.

2) Hybrid. Currently, there does not exist a post-quantum key exchange that is trusted at the level that ECDH is trusted against conventional (non-quantum) adversaries. A hybrid approach allows introducing promising post-quantum candidates next to well-established primitives, since the overall security is at least as strong as each individual primitive.

3) Focus on quantum-resistant confidentiality. A passive attacker can eavesdrop 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 future, this framework can be used unchanged to facilitate migration to those algorithms.

6) Identification of hybrid algorithms. The usage of a hybrid approach in which each hybrid combination of several classical and post-quantum algorithms leads to a different group identifier can lead to an exponential number of combinations. Thus, the design should seek an approach in which a hybrid group -- comprising multiple post-quantum algorithms -- can be efficiently negotiated.

7) 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.

8) 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.

9) Deterministic operation. This requirement means that the hybrid post-quantum exchange, and thus, the computed key, will be based on algorithms that both client and server wish to support.

10) 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.

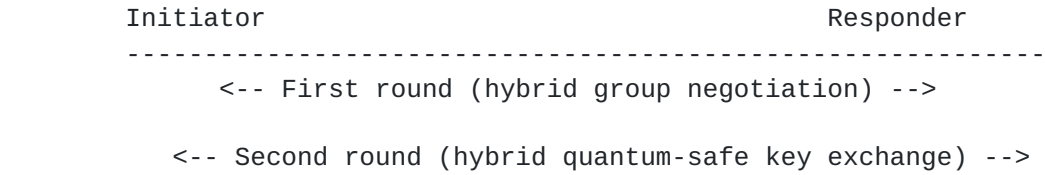
11) Backwards compatibility and interoperability. This is a fundamental requirement to ensure that hybrid post-quantum IKEv2 and a non-post-quantum IKEv2 implementations are interoperable.

12) FIPS compliance. IPsec is widely used in Federal Information Systems and FIPS certification is an important requirement. However, algorithms that are believed to be post-quantum are not FIPS compliant yet. Still, the goal is that the overall hybrid post-quantum IKEv2 design can be FIPS compliant.

3. The Framework of Hybrid Post-quantum Key Exchange

3.1. Overall design

The proposed hybrid post-quantum IKEv2 protocol extends RFC7296 [RFC7296] by duplicating the initial exchange in [RFC7296]. In order to minimize communication overhead, only the key shares that are agreed to be used are actually exchanged. In order to achieve this, the IKE_SA_INIT exchange now consists of two message exchange pairs. The first pair of IKE_SA_INIT messages negotiates which classical cryptographic algorithms are to be used, along with the supported PQC algorithms by initiator and responder, and policies of the initiator that specify its requirements on a hybrid group. The second IKE_SA_INIT message pair, on the other hand, consists of each peer sending the Diffie-Hellman public value along with the post-quantum key-shares. Note that no Diffie-Hellman exchange or exchange of post-quantum key-shares is performed in the first round of IKE_SA_INIT exchange. Messages are described as message 1 for the initiator's first message, message 2 for the responder's first message, message 3 for the initiator's second message and message 4 for the responder's second message.



This hybrid post-quantum IKEv2 key exchange can occur in IKE_SA_INIT or CREATE_CHILD_SA message pair which contains various payloads for negotiating cryptographic algorithms, exchanging nonces, and performing a Diffie-Hellman shared secret exchange for an IKE SA or a Child SA. These payloads are chained together forming a linked-list and this flexible structure allows additional key exchange payloads to be introduced. The additional key exchange uses 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.

3.2. Overall Protocol

In the following we overview the two protocol rounds involved in the hybrid post-quantum protocol.

3.2.1. First Protocol Round

In the first round, the IKE_SA_INIT request and response messages are used to negotiate the hybrid group. The method to negotiate and exchange post-quantum policies is achieved using the key exchange payload (with a Diffie-Hellman Group Num of #TBA). The KE payload with group number #TBA does not contain Diffie-Hellman or post-

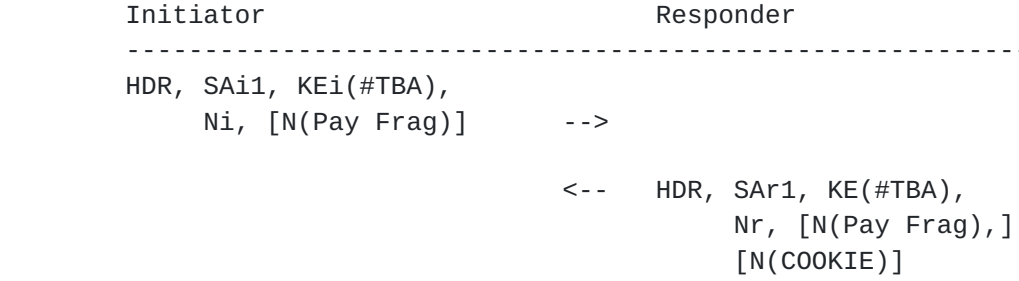
quantum public values, but proposed post-quantum algorithms and the policy for the post-quantum ciphers.

The initiator negotiates cryptographic suites as per RFC7296, the only exception is, the Diffie-Hellman KE payload is not populated with a keyshare, but rather the KE payload contains the proposed post-quantum algorithms and policies. The Diffie-Hellman groups are negotiated in the security association payload as per RFC7296 and the public values sent in the next round of exchange.

When a responder that supports the hybrid exchange, receives an IKE_SA_INIT message with a KE payload with group number #TBA, it performs a lookup of the policies and algorithms contained within the KE payload. Assuming that it supports one or more proposed post-quantum algorithms, it then indicates these in the KE payload response with group number #TBA. The responder also selects the cryptographic suites, including the chosen Diffie-Hellman Group Num in the security association payload as per RFC7296. In this exchange the Diffie-Hellman public value is not sent in the KE payload.

The initiator can signal support of IKE_SA_INIT message fragmentation by including a payload fragmentation Notify payload. The responder can also include this Notify payload to indicate support of IKE_SA_INIT message fragmentation.

The responder may choose to allocate state to the session, as the initial message is used in authenticating the IKE SA messages. Optionally, the responder may prefer not to allocate any state and reply with a cookie instead. The cookie can provide two functions. One being the standard RFC7296 behaviour. The other benefit of using the cookie is to provide fast detection of a downgrade attack without running into the risk of state exhaustion attacks. Whether or not any states are allocated, the responder detects the post-quantum cryptographic algorithms and policies that do not match and can then abort the session prior to calculating the shared secrets. See Section 3.7 for more information on cookie and downgrade attack prevention.



By using the KE payload, peers that do not support the hybrid

exchange will reject the initial negotiation and assuming that a Diffie-Hellman Group Num contained in the Diffie-Hellman Group Transform IDs was acceptable, the peer will send an INVALID_KE_PAYLOAD message to indicate its preferred Diffie-Hellman group. Note that using the KE payload enables backward compatibility with existing RFC7296 implementations. If this scenario occurs, the initiator SHOULD retry the hybrid exchange. Dependent on policies, the initiator may retry the exchange as per RFC7296, and if this occurs then the N(PQ_ALGO_POLICIES) notify payload MUST be included to prevent downgrade attacks. The N(PQ_ALGO_POLICIES) notify payload contains the same post-quantum algorithms and policies that were sent in the KE(#TBA) payload in the first round of IKE_SA_INIT request.

On receipt of the N(PQ_ALGO_POLICIES) payload, the responder MUST validate these post-quantum algorithms and policies against its own policies. This validation is required to ensure that the post-quantum algorithms were not amended in the initial exchange, resulting in a downgrade attack.

Should the proposed post-quantum algorithms not be acceptable to the responder, the responder SHOULD indicate this by sending the INVALID_KE_PAYLOAD Notify message to indicate its preferred Diffie-Hellman group or the NO_PROPOSAL_CHOSEN Notify message if no Diffie-Hellman group were acceptable. If the classical cryptographic suite is acceptable, but the post-quantum algorithms are not, the responder SHOULD indicate this by specifying the preferred Diffie-Hellman group in the INVALID_KE_PAYLOAD notification. The initiator should then revert to the classical IKEv2 exchange and include the N(PQ_ALGO_POLICIES) payload to prevent downgrade attacks. Below is an example that shows the proposed hybrid group is not supported by the responder and that the responder prefers a Diffie-Hellman Group 19 (P-256), assuming that this group is in the list proposed (although it is not preferred), in the previous message.

```
   Initiator                       Responder
   -----------------------------------------------------------
   HDR, SAi1, KEi(#TBA),
        Ni, [N(Pay Frag)]    -->

                               <--  HDR, N(INVALID_KE_PAYLOAD, 19)
   HDR, SAi1, KEi(19),
        N(PQ_ALGO_POLICIES), -->
        Ni
```

For implementations that mandate only the use of hybrid exchange, these MUST not revert to using the classical IKEv2 exchange. This should be a configurable parameter in implementations.

As per RFC7296, should the responder not accept any of the cryptographic suites that were sent in the security association payload, a NO_PROPOSAL_CHOSEN message is responded, as depicted below.

```
   Initiator                          Responder
   -----------------------------------------------------------
   HDR, SAi1, KEi(#TBA),
        Ni, [N(Pay Frag)]      -->

                                  <--   HDR, N(NO_PROPOSAL_CHOSEN)
```

3.2.2. Second Protocol Round

In the second protocol round, the initiator sends again the IKE_SA_INIT request. The main difference is that this message includes the key-shares associated to each of the post-quantum algorithms agreed in the previous round. Each key-share is transported in a KE payload, and therefore there may exist multiple KE payloads in the second round of the IKE_SA_INIT message. Furthermore, these KE payloads may be fragmented if the key-shares are large and both peers indicate the support of fragmentation.

In a general hybrid arrangement, the RFC7296 Diffie-Hellman public value is sent in the first KE payload (denoted KEi1), with one or more post-quantum key-share being sent in additional KE payloads (denoted KEi2, KEi3, etc). However, this ordering is not mandatory.

If the responder sent a cookie in the first round of exchange, the initiator MUST return this cookie. In addition to that, the initiator MUST send the same post-quantum algorithms and policies that were included in the KE payload of type #TBA sent in the first round of the exchange, in a notify payload N(PQ_ALGO_POLICIES). The responder MUST examine the post-quantum algorithms and policies, and confirm that the presented KE payloads match those represented by the cookie, see Section 3.7 for more information. Should an anomaly or a mismatch be detected, the responder MUST abort the session. On the other hand, if the validation passes, then the responder can proceed to compute a shared secret as detailed in Section 3.3.

The responder also sends the IKE_SA-INIT response message including its key-shares. As before, if agreed and if required, fragmentation is handled as described in Section 3.6. Once the initiator has received all key-shares from the responder, the initiator can compute the same shared secret following the description in Section 3.3.

Below is an example message exchanged in the second round of IKE_SA_INIT message.

```
   Initiator                          Responder
   -----------------------------------------------------------
   HDR, [N(COOKIE),] SAi1,
       KEi1[, KEi2, ..., KEiX,]
       Ni[, N(PQ_ALGO_POLICIES)] -->

                                <--  HDR, SAr1, KEr1[, KEr2,
                                          ..., KErX,] Nr
```

For implementations that are to be used by peers that are pre-configured with matching hybrid policies, resulting in the initial exchange used to negotiate the post-quantum policies and algorithms contained in the first round of exchanges being redundant, the initiator can skip the first round of exchanges by sending the IKE_SA_INIT containing the key-shares. However the initiator MUST be sure that the responder will accept the presented key-shares. In this instance the responder is open to abuse by fragmentation related attacks and MUST revert to using the initial exchange, should it find itself under any form of attack.

3.2.3. Child SA Negotiation

After the initial IKE SA is negotiated, either side may later negotiate child SAs or rekey the IKE SA which may involve a fresh key exchange. If a hybrid group is desired, then the initiator proposes a Transform Type 4 (Diffie-Hellman) of (TBA); he includes the KE payloads for the key exchange types that were negotiated for the child SAs during the initial negotiation (see Section 3.5.1). The responder replies with the corresponding KE payloads, and both use the shared secrets to generate the child SA keying material (see section 3.3). If hybrid groups were not initially negotiated as a part of the initial key exchange, then child SAs MUST NOT propose a hybrid group.

Specifically, the key exchange for creating a child SA using a hybrid group is:

```
   Initiator                          Responder
   -----------------------------------------------------------
   HDR, SK{SA, Ni, KEi1, KEi2,
        ..., KEiN, TSi, TSr}   -->
                               <--  HDR, SK{SA, Nr, KEr1, KEr2,
                                         ..., KErN, TSi, TSr}
```

where both SA payloads include a transform type 4 of (TBA), and the KEi1, ..., KEiN, KEr1, ..., KErN are the KE types there were initially negotiated.

3.3. Computation of a Post-Quantum Shared Secret

After the second protocol round detailed in Section 2.2., both initiator and responder can compute the common shared secrets to generate an SKEYSEED, from which all keying materials for protection of the IKE SA are derived. The quantity SKEYSEED is computed as follows:

SKEYSEED = prf(Ni | Nr, SS1 | SS2 | ...| SSN)

where prf, Ni and Nr are defined as in IKEv2 [RFC7296], SSi represents the i-th shared secret computed from the i-th key exchange algorithm contained in the hybrid group that was negotiated in the protocol. Note that if at least one of these SSi is a classical shared secret that is FIPS approved, then FIPS compliance design criteria as outlined in Section 2 is achieved. The seven secrets derived from SKEYSEED, namely SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, and SK_pr, are generated as defined in IKEv2 [RFC7296].

For child SAs that are negotiated using a hybrid group, the keying material is defined as:

KEYMAT = prf+(SK_d, SS1 | SS2 | ... | SSN | Ni | Nr)

where SSi represents the i-th shared secret computed from the i-th key exchange algorithm that was performed during the negotiation of the child SA.

When rekeying an IKE SA using a hybrid group, the new SKEYSEED is computed as:

SKEYSEED = prf(SK_d (old), SS1 | SS2 | ... | SSN | Ni | Nr)

where SSi represents the i-th shared secret computed from the i-th key exchange algorithm that was performed during the negotiation of the new IKE SA.

3.4. Post-Quantum Group Transform Type and Group Identifiers

In generating keying material within IKEv2, both initiator and responder negotiate up to four cryptographic algorithms in the SA payload of an IKE_SA_INIT or a CREATE_CHILD_SA exchange. One of the negotiated algorithms is a Diffie-Hellman algorithm, which is used for key exchange. This negotiation is done using the Transform Type 4 (Diffie-Hellman Group) where each Diffie-Hellman group is assigned a unique value.

In order to enable a post-quantum key exchange in IKEv2, the various

post-quantum algorithms MUST be negotiated between two IKEv2 peers. To this end, this draft assigns new meanings to various transforms of transform type 4 ("Diffie-Hellman group"). It assigns identifiers to each of the various post-quantum algorithms (even though they are not actually Diffie-Hellman groups, they are methods of performing key exchange). In addition, it assigns two artificial values that are not actually key exchange mechanisms, but are used as a part of the negotiation.

We expect that in the future, IANA will assign permanent values to these transforms. Until it does, we will use the following mappings in the 'reserved for private use space':

0x9000 HYBRID - this signifies that we are negotiating a hybrid group, the details are listed in the KE payload.

The following values are reserved for the below key exchanges: 0x9100 - 0xXXXX. The following abstract identifiers are used to illustrate their usage in our framework. Actual identifiers will be maintained by IANA and updated during the NIST standardization process.

Name	Number	Key exchange
NIST_CANDIDATE_1	0x9100	The 1st candidate of NIST PQC submission
NIST_CANDIDATE_2	0x9101	The 2nd candidate of NIST PQC submission

Because we are using transforms in the private use space, both the initiator and responder must include a vendor id with this payload:

d4 48 11 94 c0 c3 4c 9d d1 22 76 aa 9a 4e 80 d5

This payload is the MD5 hash of "IKEv2 Quantum Safe Key Exchange v1"). If the other side does not include this vendor id, an implementation MUST NOT process these private use transforms as listed in this draft.

3.5. Hybrid Group Negotiation

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 classical key exchanges with post-quantum key agreements. However, there are many post-quantum proposals that when combined can lead to many potential hybrid groups. Furthermore, different organizations might have different requirements when using a hybrid group. For instance, the type of underlying problem that is trusted, the minimum number of algorithms that should be used in a hybrid group, or the security strength of each of the algorithms. For these reasons, hybrid groups might lead to many potential combinations difficult to define, maintain and standardize. This motivates our hybrid group negation protocol.

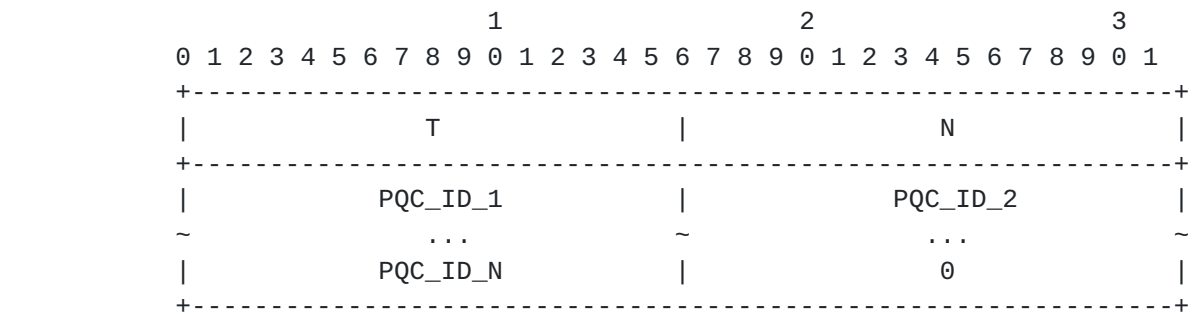
Our hybrid group negotiation protocol allows the initiator and responder to agree on a common hybrid group based on their respective policies. The protocol categorizes each type of key exchange into a type T, which is based on the type of hard problem it relies upon. We use the aforementioned abstract identifiers to illustrate the idea.

Given this categorization of the key agreement protocols, initiator and responder have security policies that define the minimum number of post-quantum algorithms that they require in a hybrid group and also the types of key agreement protocols that they support (and therefore, trust). The initiator and responder can then engage in a simple protocol that allows them to compute a common hybrid group fulfilling their policies. The protocol for the initiator and responder is described below.

Note that it would be possible for the responder to search for a mutually acceptable combination without specifying the key exchange types, however the algorithm to search for such a combination would be considerably more complex. This design assumes that the security policies of the initiator and the responder will rely on key exchanges based upon distinct types of hard problems, and hence the additional complexity of the more general algorithm is not warranted.

3.5.1. Protocol for the Initiator

To define the protocol, we first define a "DH_Group_List", this is a list of groups of a specific type; it has the format:

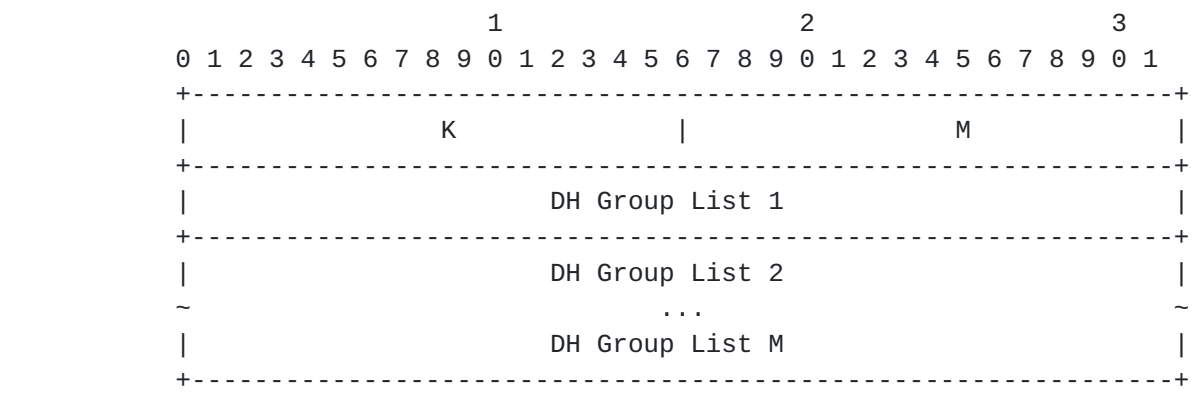


where

- T is the type of the groups that are in this list, with this proposed encoding:
  - 0x0001 is classical
  - 0x0002 is lattice
  - 0x0003 is code-based
  - 0x0004 is isogeny-based
  - 0x0005 is symmetric (preshared key)
- N is the number of groups that make up the list. The semantics of this proposal is that the initiator is proposing M different types of groups; any selection of one group from K different types will be acceptable.
- PQC_ID_1, PQC_ID_2, PQC_ID_N, such as NIST_CANDIDATE_1, is the identifier for the PQC algorithm to be used for negotiation, in order of preference.
- 0 is padding, present if N is odd.

The semantics of this list is that these are the groups of PQC algorithms of type T that are acceptable to the initiator.

We now define a "DH_Group_Policy"; this is a list of group types that are negotiated together. It has the format:



where:

- K is the minimum number of group lists that must be satisfied;
- M is the number of group lists;
- DH_Group_LIST_1, ..., DH_Group_List_M are the lists of different types of DH groups, in order of preference.

The semantics of this proposal is that the initiator is proposing M different types of groups; any selection of one group from K different types will be acceptable.

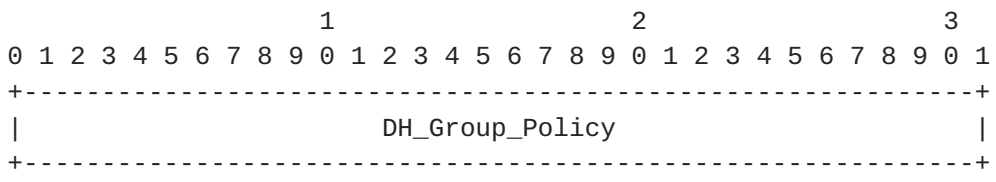
For example, suppose our policy is "we must agree on at least 2 groups from the list (P-256, P-384), (NIST_CANDIDATE_1, NIST_CANDIDATE_2; both of type lattice) and (NIST_CANDIDATE_1 of type isogeny), where NIST_CANDIDATE_1 and NIST_CANDIDATE_2 of type lattice are assigned group numbers 40 and 41 respectively, and NIST_CANDIDATE_1 of type isogeny is assigned group number 60"; we have the following list (in hexadecimal)

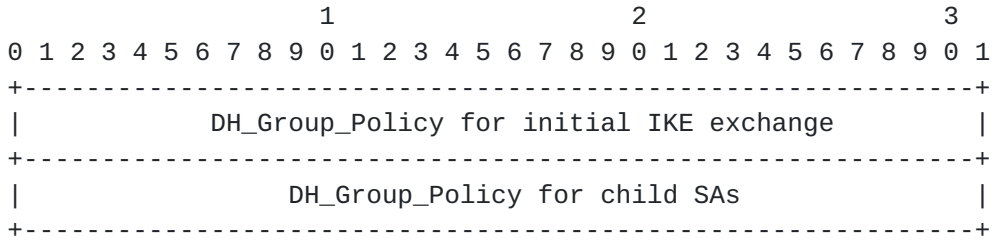
```
0002 0003 0001 0002 0013 0014 0002
0002 0028 0029 0004 0001 003c 0000
```

which is parsed as

```
0002       K = 2
0003       We have 3 group lists
0001 0002  First list is of type classical, and consists of two
           groups
0013 0014  Group 19 (P-256) and group 20 (P-384)
0002 0002  Second list is of type lattice, and consists of two
           groups
0028 0029  Group 40 (NIST_CANDIDATE_1 of type lattice) and group
           41 (NIST_CANDIDATE_2 of type lattice)
0004 0001  Third list is of type isogeny, and consists of one
           group
003c       Group 60 (NIST_CANDIDATE_1 of type isogeny)
0000       Zero-pad
```

We can now give the format that the initiator sends to the responder in the KEi payload. The initiator sends its group policy in one of the following two formats:

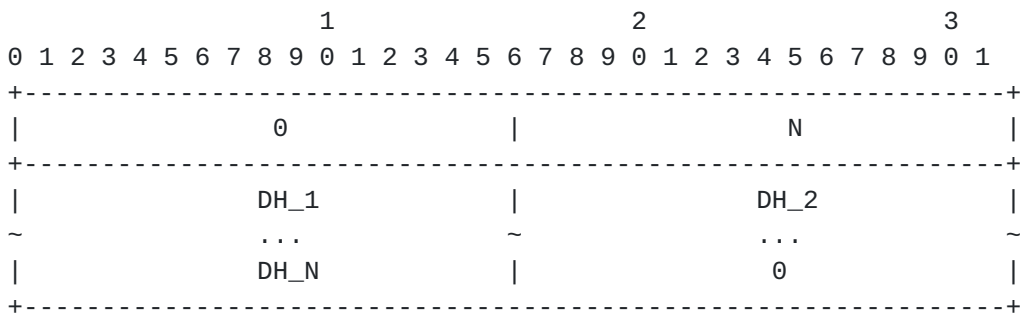


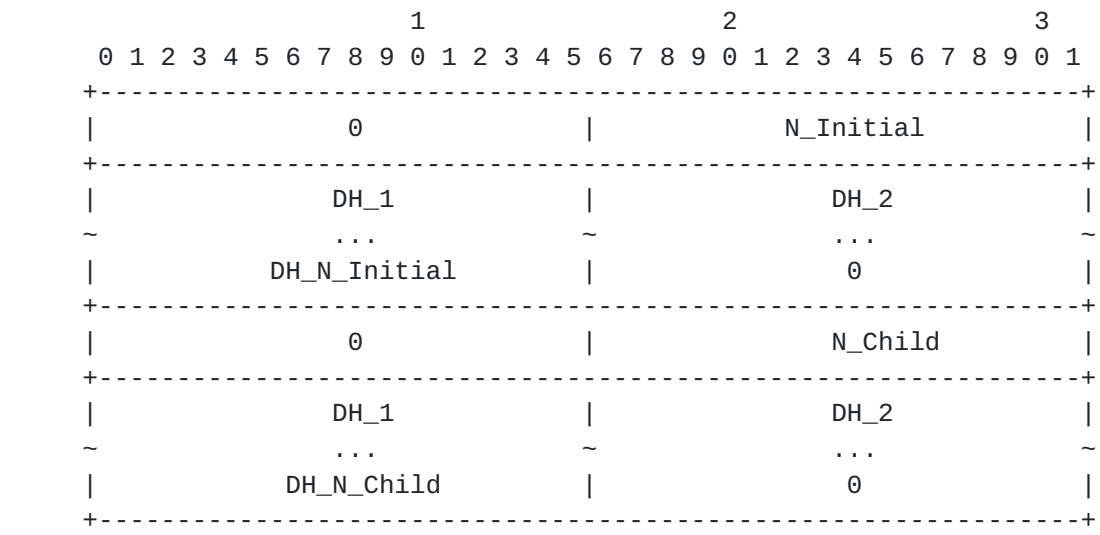


If the initiator uses the first format, then the same DH policy will be negotiated for both the initial IKE exchange, as well as any child SA exchanges. If the initiator uses the second format, then the first policy listed will be used for the initial IKE exchange; the second policy listed will be used for any child SA negotiations.

3.5.2. Protocol from the Responder

If the responder finds a combination of groups that are mutually acceptable, then it responds with the KEr payload in one of the following two formats:





where

- 0 is a fixed 0000 pattern;
- N, N_Initial, N_Child is the number of groups that are selected;
- DH_1, DH_2, ..., DH_N are the selected groups.

If the second format is selected, then the groups used for the initial IKE SA and the groups used for child SAs are listed separately.

We assume that the responder has a similar local policy governing what it is willing to negotiate. To search the initiator's vector to find such a mutually acceptable combination, the responder can run the following algorithm.

1. Set list of accepted DH groups to empty
2. Set K to the maximum of (minimum number of group lists specified by the initiator, minimum number of group lists acceptable according to the responder policy).
3. For every DH_Group_list in the initiator proposal
   a. Set T to the DH_Group_list type
   b. Find for the responder DH_Group_list of type T
   c. If the responder has such a group list
      * Scan for a DH element that the two lists have in common
        - If there is such a group
          - Append that to the "list of accepted DH groups"
          - (Optional) if the list is at least K elements long, stop searching (and accept)
4. If the list of accepted DH groups is at least K elements long, accept. Otherwise, fail.

3.6. Fragmentation Support

3.6.1. Fragmentation Problem Description

When the IKE message size exceeds the path MTU, the IKE messages are fragmented at the IP level. IP fragments can be blocked or dropped by network devices such as NAT/PAT gateways, firewalls, proxies and load balancers. If IKE messages are dropped, the IKE and subsequent IPsec Security Association (SA) will fail to be established. In many instances the quantum safe key exchange data could be too large to send in a single IKE message as the path MTU between hosts is set below the total size of the IKE message. As this draft defines multiple key exchanges in a single IKE message, there is a high chance that IP fragmentation will occur in IKE_SA_INIT messages.

The maximum length of an IKE payload is 65,535 octets. It is anticipated that some post quantum algorithms will require a key exchange payload size that is greater than 65,535 octets. Furthermore, CERT payloads in IKE_AUTH messages are expected to exceed 65,565 octets when sending certificates containing post quantum public keys and signatures.

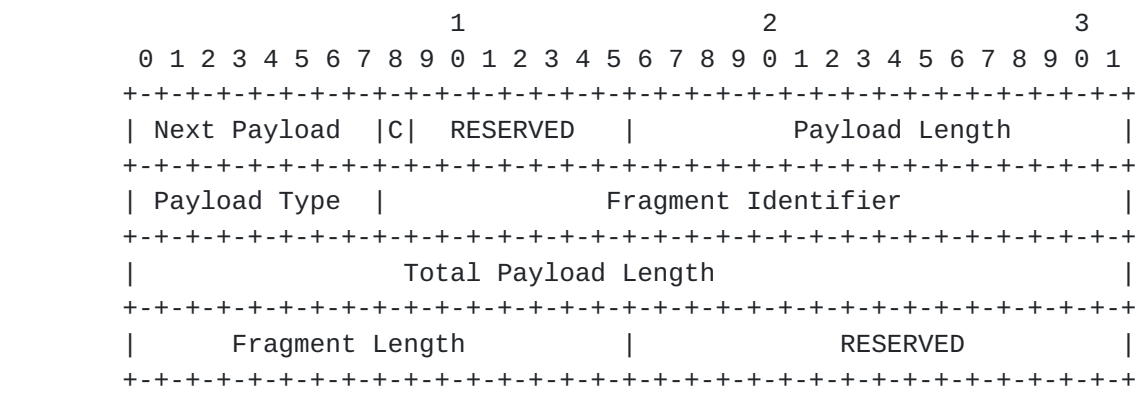
To overcome these limitations we present a method to split any payload into multiple fragments and optionally send these fragments in separate IKE_SA_INIT messages.

3.6.2. Fragmentation Solution

To enable fragmentation of IKE payloads, we introduce new FRAG_POINTER and FRAG_BODY payloads. Further, we introduce a method of sending payload fragments in multiple IKE_SA_INIT messages as well as a method of sending payload fragments in encrypted IKE messages which then may or may not be fragmented using RFC 7383's IKEv2 message fragmentation.

3.6.2.1. Fragmentation Pointer Payload

In place of any payload within an IKE packet, the sender may replace it with a FRAG_POINTER payload; this FRAG_POINTER type (rather than the original payload type) will appear in the next payload field of the previous payload (or IKE header). This payload has the following format

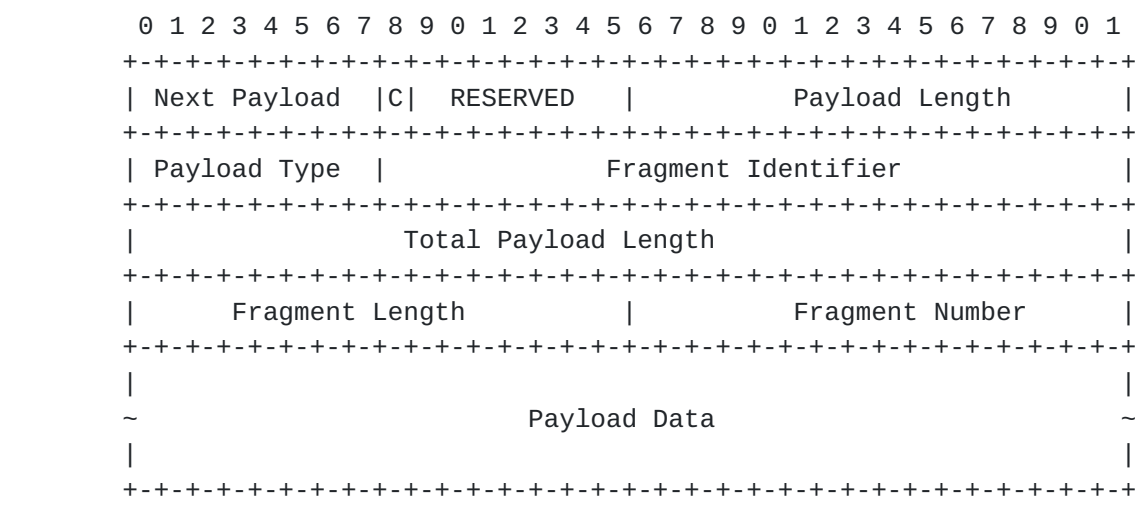


where:

- C is the Critical flag for the original payload.
- Payload Type is the payload type of the original payload; e.g. if this payload is a KE payload, this will be the value 34.
- Fragment Identifier is a 24 bit value that the sender does not reuse often, that is, within the timeout period of this IKE packet. It is intended to be used to allow the receiver to correlate the fragments (contained in other packets) to the payload within the original IKE packet.
- Total Payload Length is the length of the original payload. Note that this draft allows the transmission of payloads greater than 64k, if necessary.
- Fragment Length is the amount of data contained within each fragment (except for the last fragment, which may be smaller).
- RESERVED will be an all-0's field.

3.6.2.2. Fragmentation Body Payload

The sender can split the contents of any payload across one or more FRAG_BODY payloads. This payload has the format:



where:

- Next Payload is the identifier for the payload type of the next payload in the message. There may be additional restrictions on the value of Next Payload during the fragmentation of an IKE_SA_INIT message, see Section 3.6.2.3 below.
- Payload Type, Fragment Identifier, Total Payload Length, Fragment Length are the same as the corresponding fields in the FRAG_POINTER payload. Take careful note, like the other fields described here the Fragment Length field will be identical across all fragments. Thus, if this is the last fragment, Fragment Length could be longer than the size of the Payload Data field.
- Fragment Number is the current fragment message number, starting from 1. This field MUST NOT be 0.
- Payload Data is the contents of the payload for this fragment. For any fragment other than the last, this will be 'Fragment Length' bytes long; for the last one, it will be (Total Payload Length-1) % Fragment Length + 1 bytes long. Note that the Generic Payload Header from the original payload MUST NOT be included in the Payload Data of the fragment, but any additional payload header fields after the Generic Payload Header MUST be included. The Generic Payload Header cannot be included because it includes the 16-bit Payload Length field, however the length of a fragmented payload may require more than 16 bits to be stored.

The logical contents of the reassembled payload will be

Payload Data[1] | PayloadData[2] | ... | PayloadData[N]

where N = Total Payload Length / Fragment Length (rounded up).

As an example, the following KE payload could be fragmented into a FRAG_POINTER and two FRAG_BODY payloads with Fragment Length of 1000 as follows:

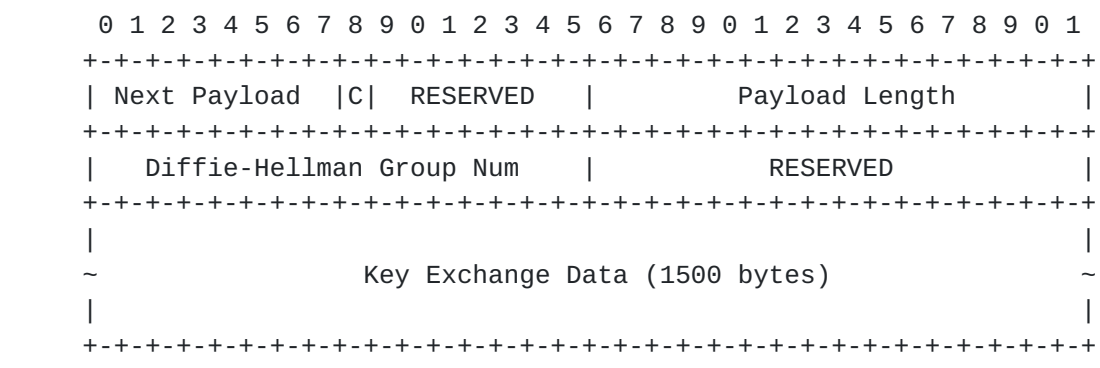


Figure 1: Key Exchange Payload to be Fragmented

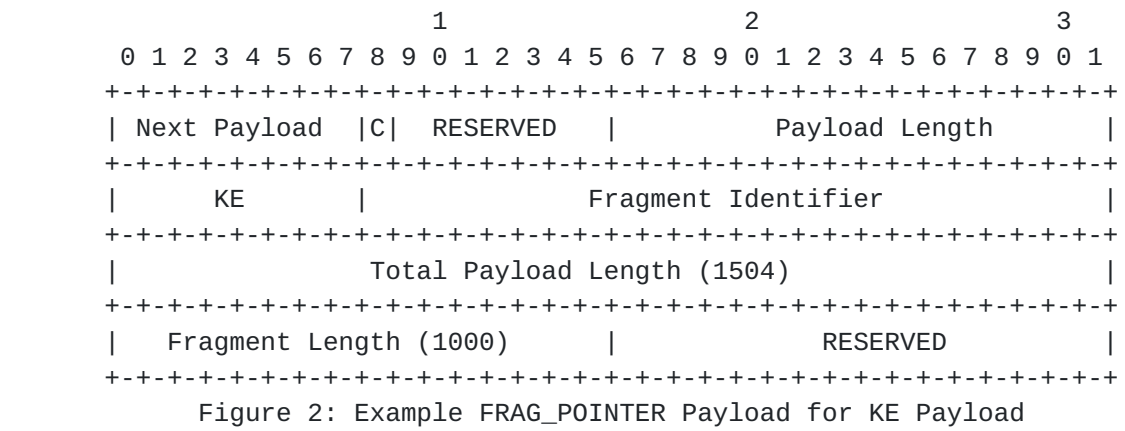


Figure 2: Example FRAG_POINTER Payload for KE Payload

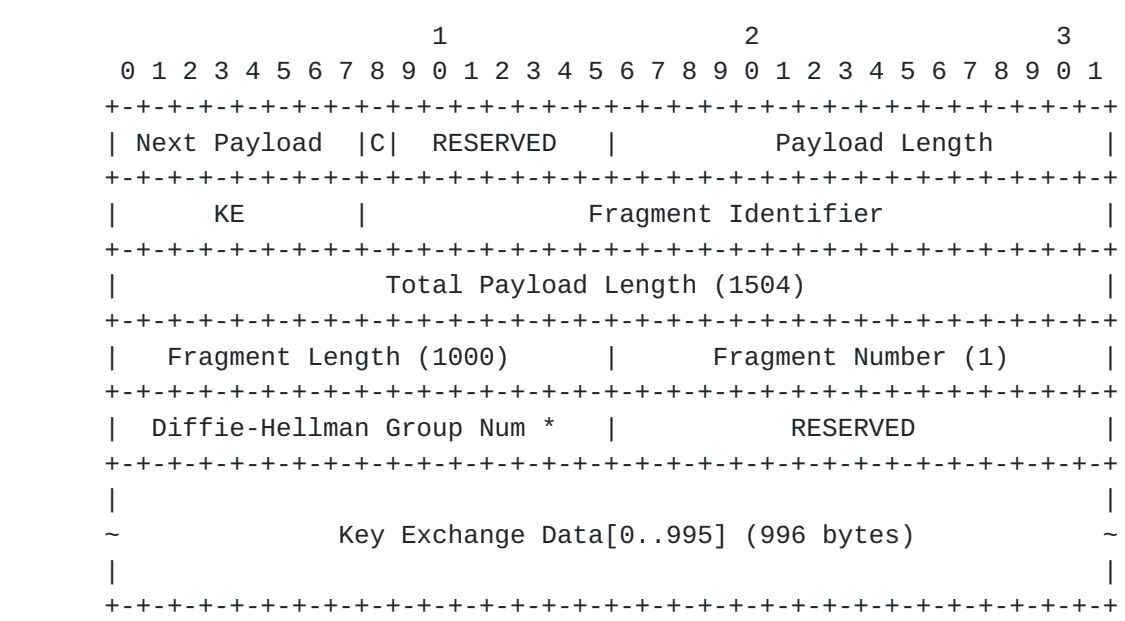


Figure 3: Example FRAG_BODY Payload 1 for KE Payload

(*) Corresponds to the payload-specific header fields beginning immediately after the Generic Payload Header of the Key Exchange payload being fragmented. This is the beginning of the Payload Data field in the FRAG_BODY payload.

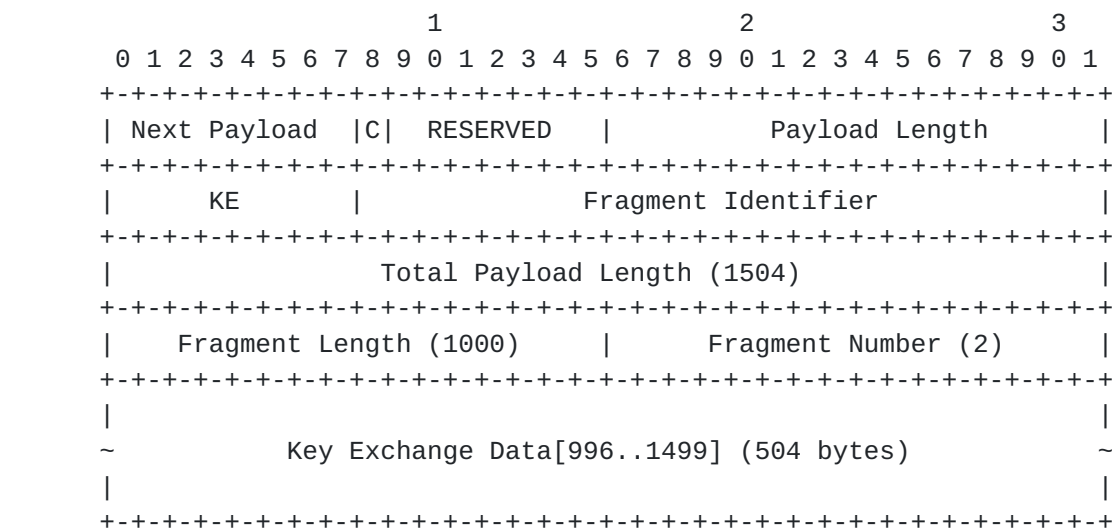


Figure 4: Example FRAG_BODY Payload 2 for KE Payload

3.6.2.3. Fragmentation Semantics

If the receiver supports this fragmentation extension, the sender may fragment any payload by replacing the payload with a FRAG_POINTER payload and one of more FRAG_BODY payloads. If IP fragmentation is not a concern (e.g. when IKEv2 fragmentation is achieved using encrypted fragment payloads, or it's known that IP fragmentation of IKE_SA_INIT won't be an issue) then the corresponding FRAG_BODY payloads MUST appear anywhere after the FRAG_POINTER in an IKE message.

An IKE_SA_INIT message may be fragmented across multiple IKE messages using this payload fragmentation. In this case the sender first sends an IKE_SA_INIT message containing the FRAG_POINTER payloads and any unfragmented payloads. Then it sends one IKE_SA_INIT message per FRAG_BODY payload generated from the original IKE_SA_INIT message. Each IKE_SA_INIT message must be sent with a Message ID of 0. Each IKE_SA_INIT message subsequent to the first one MUST contain one FRAG_BODY payload, MAY contain a COOKIE notification and SHOULD NOT contain any other payloads. Since FRAG_POINTER support is negotiated in an initial IKE_SA_INIT round-trip which didn't generate any shared keys, the responder had the opportunity to send a COOKIE notify payload back to the initiator. This COOKIE can be used by the responder as a denial-of-service prevention measure. If the sender received a COOKIE notification payload in the initial exchange, it MUST include the COOKIE notify payload in each fragmented IKE_SA_INIT message that it sends. This allows the receiver to reject any IKE_SA_INIT messages without a COOKIE or with an unrecognized COOKIE, thus mitigating a DoS attack where an attacker sends malformed IKE_SA_INIT messages containing a FRAG_BODY payload which the receiver would enqueue, filling up its receiving buffers. Note, this

does not prevent an attack where the attacker listens in on messages to determine a valid COOKIE and emits malformed IKE_SA_INIT messages with that cookie, or where it sends a valid initial round IKE_SA_INIT message to received a valid cookie and then emit malformed messages using that cookie.

When the receiver receives an IKE payload with one or more FRAG_POINTER payloads, it waits until it processes all the corresponding FRAG_BODY payloads to transform the payloads into the original unfragmented payload which it processes as normal. If the IKE message was not a fragmented IKE_SA_INIT message, all corresponding FRAG_BODY payloads will be contained in the IKE message, if they are not the receiver MUST reject the IKE message.

When the receiver receives an IKE_SA_INIT message, is may have to process several IKE_SA_INIT messages to reconstruct the original unfragmented message. If it receives the initial message containing the FRAG_POINTER payloads, it enqueues that message and awaits the corresponding IKE_SA_INIT messages containing the FRAG_POINTER payloads needed to reconstruct the original message. In addition, if it receives a FRAG_BODY message without receiving a corresponding FRAG_POINTER payload first, it may enqueue that payload.

The receiver may vet the declared payload length, and reject it if it decides that the length is too long.

Also note that we allow the FRAG_BODY payload to consist of the entire payload; this can happen if (for example) the MTU size is 1500, and we want to transmit a 1300 byte KE payload, in addition to 400 bytes of other IKE data.

Once all the FRAG_BODY payloads have been received and reassembled, the IKE receiver may commence parsing the IKE packet. This proceeds as normal, except that when it sees a payload of type FRAG_POINTER, it looks into the FRAG_POINTER payload to see the actual payload type and length, and refers to the reassembly buffer for the actual payload data.

Note about the criticality field; a FRAG_POINTER field may be marked as noncritical, which means that the IKE parser may ignore it if it does not understand the payload type within the FRAG_POINTER payload. However, even if it does that, it MUST still reassemble all the FRAG_BODY payloads (because of the IKE AUTH processing depends on them).

3.6.2.4. IKE AUTH Processing

When generating the IKE AUTH payload, the reassembled texts contained

within the FRAG_BODY payloads is logically appended to the IKE message (and before the nonce). Specifically, we modify how InitiatorSignedOctets and ResponderSignedOctets are computed as follows:

```
InitiatorSignedOctets = RealMessage1 | PayloadData1 |
              PayloadData2 | ... | PayloadDataN |
              NonceRData | MACedIDForI

ResponderSignedOctets = RealMessage2 | PayloadData1 |
              PayloadData2 | ... | PayloadDataN |
              NonceIData | MACedIDForR
```

where PayloadData1, ..., PayloadDataN are the fields from the FRAG_BODY payloads associated with the IKE message being authenticated, in the same order that the corresponding FRAG_POINTERS appear in, and for payloads from the same FRAG_POINTER, in increasing FRAGMENT_NUMBER order.

3.6.2.5. Design Rationale

The contents of the FRAG_POINTER/FRAG_BODY payloads were designed to make it easy to reassemble. The initial segments of the payloads were intentionally kept identical (to simply the processing if the FRAG_BODY arrived first); the receiver always knows how long the payload will be (allowing the allocation of buffers, if required); the receiver always knows the location in the payload data of each fragment (and so is able to save the data immediately into the buffer), and the receiver can compute the number of fragments up front (and hence can easily tell when all fragments have been received).

The method of performing IKE AUTH processing was also designed to make it easy to implement; that PayloadData1 | PayloadData2 | ... | PayloadDataN is just the reassembled payloads concatenated together.

3.7. Protection against Downgrade Attacks

In RFC7296, man-in-the-middle (MitM) downgrade attack is prevented by always resending the full set of group proposal in subsequent IKE_SA_INIT messages if the responder chooses a different Diffie-Hellman group from the one in the initial IKE_SA_INIT message. The two-round nature of the protocol in this document presents some challenges in terms of downgrade attack protection. However, the general idea is the same as the one in RFC7296, in that the responder must have sufficient information to verify that the downgrade is a genuine one, rather than one instigated by a malicious attacker. Consider the following example: an initiator proposes to use a hybrid

key exchange, and for backward compatibility also purposes a Diffie-Hellman group 19 (P-256 elliptic curve) through SAi payload, in the first round of the exchange. The initiator may receive an INVALID_KE_PAYLOAD notification response. This could be a genuine response from a responder that does not understand or support the selected hybrid key exchange, or it could also be a malicious downgrade response from an MitM attacker. The initiator, on the second round of the exchange, MUST send the same cipher proposals and policies as in the first exchange round to indicate that the initiator would have preferred to use the hybrid key exchange. The responder MUST check that the chosen proposal is indeed not caused by a downgrade attack. If the check fails, it indicates a potential downgrade attack and the connection SHALL be dropped immediately.

In order to check the proposals and policies, the responder may choose to maintain states between IKE_SA_INIT rounds. However, this increases the risk of state exhaustion attack. Of course, the responder may decide not to allocate any states and rely on the authentication in IKE_AUTH for any tampering of the exchange. Unfortunately, this option does not offer the benefit of an early downgrade attack detection; the responder will have to spend resources computing entities such as shared secrets and authentication code before knowing whether or not there is a downgrade attack. Such a benefit may be obtained by encoding some information into a cookie (Section 2.6. RFC7296).

Whilst this document does not mandate how information should be encoded to form the cookie, it could be efficiently done as follows

Cookie = <VersionIDofSecret> | Hash(KEi(#TBA) | <secret>)

where KEi(#TBA) is the KE payload in the first round of IKE_SA_INIT exchange, <secret> is a randomly entity generated by the responder which SHOULD be changed periodically as suggested in RFC7296, and the entity <VersionIDofSecret> should be updated whenever <secret> is changed. In this scenario, the responder calculates a cookie value from the first round of the IKE_SA_INIT request message and sends it to the initiator as part of the first round IKE_SA_INIT response message. The initiator echoes back the cookie and a N(PQ_ALGO_POLICIES) notify payload along with other IKE_SA_INIT attributes. When the responder receives the second round of the IKE_SA_INIT message, it recalculates the cookie value and checks whether or not this value is the same as the one in the previous round of the exchange, which is available from N(PQ_ALGO_POLICIES). If they mismatch, it indicates an attempt to force a downgrade attack and therefore the connection SHALL be terminated. As before, any attempts of the attacker to modify the packets so that cookie validation passes will be detectable in IKE_AUTH stage.

In the event of the value <secret> goes out-of-sync, as suggested in RFC7296, the responder MAY reject the request by responding with a new cookie, or it MAY keep using the old value of <secret> for a short time and accept cookies computed from either one.

The complete two-round IKE_SA_INIT message exchange flow with cookie is shown below. In this particular example, the responder understands and accepts the hybrid key exchange proposed in the first IKE_SA_INIT round.

```
   Initiator                          Responder
   ------------------------------------------------------------------
   HDR, SAi1, KEi(#TBA),
        Ni, [N(Pay Frag)]       -->
                                <--    HDR, SAr1, KE(#TBA),
                                            Nr, N(COOKIE),
                                            [N(Pay Frag),]

   HDR, N(COOKIE), SAi1,
       KEi1[, KEi2, ..., KEiX,]
       Ni, N(PQ_ALGO_POLICIES) -->
                                <--    HDR, SAr1, KEr1[, KEr2,
                                            ..., KErX,] Nr
```

The following shows the flow whereby the responder does not support the proposed hybrid key exchange and proposes to switch to classical Diffie-Hellman key exchange of type P-256. Because the responder does not keep any states, it relies on the cookie and N(PQ_ALGO_POLICIES) to detect that it is a genuine downgrade.

```
   Initiator                          Responder
   ------------------------------------------------------------------
   HDR, SAi1, KEi(#TBA),
        Ni, [N(Pay Frag)]       -->
                                <--    HDR, N(INVALID_KE_PAYLOAD, 19),
                                            N(COOKIE)

   HDR, N(COOKIE), SAi1,
       KEi(19), Ni,
       N(PQ_ALGO_POLICIES)      -->
                                <--    HDR, SAr1, KEr(19), Nr
```

The cookie does not protect against an adversary that amends the KE(#TBA) payload in the first IKE_SA_INIT request round and also then amends the N(PQ_ALGO_POLICIES) payload in the second IKE_SA_INIT request round to create a match. In this instance, IKE_AUTH authentication SHALL fail due to the InitiatorSignedOctets being inconsistent between both peers.

The decision to use a cookie or allocate state SHOULD be a decision taken by the responder. This should be a configurable value, and/or activated when a certain threshold of half open connections is reached. The cookie is sent in addition to the other attributes contained in first round of IKE_SA_INIT response.

The cookie does not mitigate an attack whereby an adversary cause the responder to perform many lookups for the post-quantum algorithms and policies, resulting in a denial-of-service (DoS) condition. In order to mitigate this type of attack, the RFC7296 cookie mechanism or a puzzle-solving mechanism as described in RFC8019 SHOULD be used. A responder MAY decide to combine DoS and downgrade prevention cookies, in which case, the combined cookie may be encoded as follows

```
Cookie = <VersionIDofSecret> | Hash(Ni | IPi | SPIi |
                                    KEi(#TBA) | <secret>)
```

where Ni, IPi and SPIi are as described in RFC7296.

4. Alternative Design

This section gives an overview on a number of alternative approaches that we have considered, but later discarded. These approaches are:

- 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.

- o New payload types to negotiate hybrid proposal and to carry post-quantum public values

  Semantically, it makes sense to use a new payload type, which mimics the SA payload, to carry a hybrid proposal. Likewise, another new payload type that mimics the KE payload, could be used to transport hybrid public value. Although, in theory a new payload type could be made backwards compatible by not setting its critical flag as per Section 2.5 of RFC7296, we believe that it may not be that simple in practice. Since the original release of IKEv2 in RFC4306, no new payload type has ever been proposed and therefore, this creates a potential risk of having a backward compatibility issue from non-conforming RFC IKEv2 implementations. Since we could not see any other compelling advantages apart from a semantic one, we use the existing KE and notify payloads instead. In fact, as described above, we use the KE payload in the first IKE_SA_INIT request round and the notify payload to carry the post-quantum proposals and policies. We use one or more of the existing KE payloads to carry the hybrid public values.

- o Hybrid public value payload

  One way to transport the negotiated hybrid public payload, which contains one classical Diffie-Hellman public value and one or more post-quantum public values, is to bundle these into a single KE payload. Alternatively, these could also be transported in a single new hybrid public value payload, but following the same reasoning as above, this may not be a good idea from a backward compatibility perspective. Using a single KE payload would require an encoding or formatting to be defined so that both peers are able to compose and extract the individual public values. However, we believe that it is cleaner to send the hybrid public values in multiple KE payloads--one for each group or algorithm. Furthermore, at this point in the protocol exchange, both peers should have indicated support of handling multiple KE payloads.

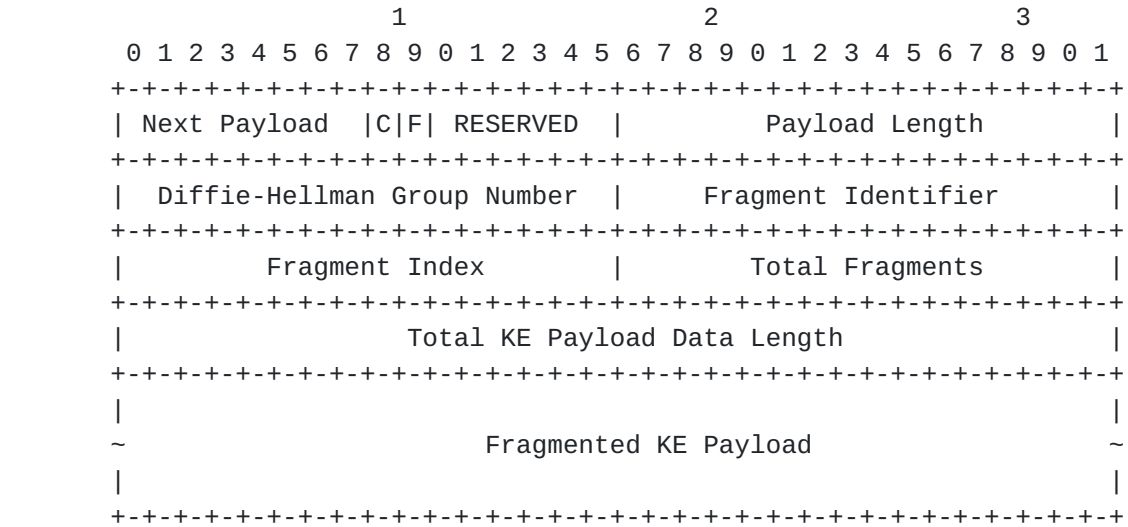
- o Fragmentation

  Handling of large IKE_SA_INIT messages has been one of the most challenging tasks. A number of approaches have been considered and the two prominent ones that we have discarded are outlined as

follows.

The first approach was to treat the entire IKE_SA_INIT message as a stream of bytes, which we then split it into a number of fragments, each of which is wrapped onto a payload that would fit into the size of the network MTU. The payload that wraps each fragment is a new payload type and it was envisaged that this new payload type will not cause a backward compatibility issue because at this stage of the protocol, both peers should have indicated support of fragmentation in the first pass of the IKE_SA_INIT exchange. The negotiation of fragmentation is performed using a notify payload, which also defines supporting parameters such as the size of fragment in octets and the fragment identifier. The new payload that wraps each fragment of the messages in this exchange is assigned the same fragment identifier. Furthermore, it also has other parameters such as a fragment index and total number of fragments. We decided to discard this approach due to its blanket approach to fragmentation. In cases where only a few payloads need to be fragmented, we felt that this approach is overly complicated.

Another idea that was discarded was fragmenting an individual payload without introducing a new payload type. The idea was to use the 9-th bit (the bit after the critical flag in the RESERVED field) in the generic payload header as a flag to mark that this payload is fragmented. As an example, if a KE payload is to be fragmented, it may look as follows.



When the flag F is set, this means the current KE payload is a fragment of a larger KE payload. The Payload Length field denotes the size of this payload fragment in octets--including the size of the generic payload header. The two-octet RESERVED field

following Diffie-Hellman Group Number was to be used as a fragment identifier to help assembly and disassembly of fragments. The Fragment Index and Total Fragments fields are self-explanatory. The Total KE Payload Data Length indicates the size of the assembled KE payload data in octets. Finally, the actual fragment is carried in Fragment KE Payload field.

We discarded this approach because we believe that the working group may not be happy using the RESERVED field to change the format of a packet and that implementers may not like the complexity added from checking the fragmentation flag in each received payload. Furthermore, we dismissed this idea in favour of the idea present in Section 3.6 due to the handling of the total IKEv2 payload size. There was not a clean method for the receiver to determine the total size of all the IKEv2 fragmented payloads. The method defined in Section 3.6 allows for a clean method for implementations to determine the IKE payload size and make a policy decision to allocate memory or discard the packet.

- Group sub-identifier

  As discussed in Section 3.4, 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.

- State Keeping in Downgrade Attack Protection

  Another way of checking whether or not a downgrade attack is in effect is to have a responder to commit the state of the first-pass of the IKE_SA_INIT message onto memory or a temporary database. When the responder receives the second-pass of the exchange, it can verify it against the saved state to determine whether or not a downgrade attack is in effect. While this simple verification does offer protection against downgrade attack, it is susceptible to state exhaustion attack. While we do not discard this idea, it is RECOMMENDED to use the other two downgrade protection mechanisms described in Section 3.7.

5. Security considerations

The key length of the Encryption Algorithm (Transform Type 1), the

Pseudorandom Function (Transform Type 2) and the Integrity Algorithm (Transform Type 3), all have to be of sufficient length to prevent attacks using Grover's algorithm [GROVER]. In order to use the extension proposed in this document, the key lengths of these transforms SHALL be at least 256 bits long in order to provide sufficient resistance to quantum attacks. Accordingly the post-quantum security level achieved is at least 128 bits.

SKEYSEED is calculated from shared, KEx, using an algorithm defined in Transform Type 2. While a quantum attacker may learn the value of KEx', if this value is obtained by means of a classical key exchange, other KEx values generated by means of a quantum-resistant algorithm ensure that 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 since 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 the hash based method, XMSS has the status of an Internet Draft, see [XMSS]. Currently, quantum-safe authentication methods are not specified in this document, but are planned to be incorporated in due course.

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

The key exchange mechanism in this document provides a method for malicious parties to send multiple KE payloads, where each of which

could be large, to a responder. As the standard behavior is for the responder to consume computational resources to compute and send multiple KE payloads back to the initiator, this allows for a simple method for malicious parties to cause a VPN gateway to perform excessive processing. In order to mitigate against this threat, implementations MAY make use of the DoS prevention COOKIE notification as defined in [RFC7296], to mitigate spoofed traffic and a puzzle-solving notification [RFC8019] to minimize the impact from hosts who use their own IP address.

Cookie notification is used to prevent downgrade attacks. The cookie SHALL NOT be of arbitrary length, otherwise it will be susceptible to SLOTH attack as described in [BL]. It is RECOMMENDED that the length of the cookie be no longer than 64 octets.

6. References

[ADPS] Alkim, E., Ducas, L., Poppelmann, T., and Schwabe, P., "Post-quantum Key Exchange - a New Hope", 25th USENIX Security Symposium, pp. 327-343, 2016.

[AH] Kent, S., "IP Authentication Header", RFC 4302, December 2005, <http://www.rfc-editor.org/info/rfc4302>.

[BL] Bhargavan, K. and Leurent, G., "Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH", Network and Distributed System Security Symposium, 2016.

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

[FMKS] Fluhrer, S., McGrew, D., Kampanakis, P., and Smyslov, V., "Postquantum Preshared Keys for IKEv2", Internet draft, https://datatracker.ietf.org/doc/draft-ietf-ipsecme-qr-ikev2, 2017.

[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

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

[LOGJAM] Adrian, D., Bhargavan, K., Durumeric, Z., Gaudry, P.,

Green, M., Halderman, J., Heninger, N., Springall, D., Thome, E., Valenta, L., VanderSloot, B., Wustrow, E., Beguelin, S., and Zimmermann, P., "Imperfect forward secrecy: How Diffie-Hellman fails in practice", Proc. 22rd ACM SIGSAC Conference on Computer and Communications Security, pp. 5-17, 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and Kivinen, T., "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7296, October 2014.

[RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, November 2014.

[RFC8019] Nir, Y., Smyslov, V., "Protecting Internet Key Exchange Protocol Version 2 (IKEv2) Implementations from Distributed Denial-of-Service Attacks", RFC 8019, November 2016.

[RFC8229] Pauly, T., Touati, S., and Mantha, R., "TCP Encapsulation of IKE and IPsec Packets", RFC8229, August 2017.

[XMSS] Huelsing, A., Butin, D., Gazdag, S., and Mohaisen, A., "XMSS: Extended Hash-Based Signatures", Crypto Forum Research Group Internet Draft, 2017

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.

Authors' Addresses

C. Tjhai
Post-Quantum
email: cjt [at] post-quantum.com

M. Tomlinson
Post-Quantum
email: mt [at] post-quantum.com

G. Bartlett
Cisco Systems
email: grbartle [at] cisco.com

S. Fluhrer
Cisco Systems
email: sfluhrer [at] cisco.com

D. Van Geest
ISARA Corporation
email: daniel.vangeest [at] isara.com

Z. Zhang
Onboard Security
email: zzhang [at] onboardsecurity.com

O. Garcia-Morchon
Philips
email: oscar.garcia-morchon [at] philips.com