Stream: Internet Engineering Task Force (IETF)
RFC: 8784
Category: Standards Track
Published: June 2020
ISSN: 2070-1721
Authors: S. Fluhrer, *Cisco Systems*; P. Kampanakis, *Cisco Systems*; D. McGrew, *Cisco Systems*; V. Smyslov, *ELVIS-PLUS*

# RFC 8784
# Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security

## Abstract

The possibility of quantum computers poses a serious challenge to cryptographic algorithms deployed widely today. The Internet Key Exchange Protocol Version 2 (IKEv2) is one example of a cryptosystem that could be broken; someone storing VPN communications today could decrypt them at a later time when a quantum computer is available. It is anticipated that IKEv2 will be extended to support quantum-secure key exchange algorithms; however, that is not likely to happen in the near term. To address this problem before then, this document describes an extension of IKEv2 to allow it to be resistant to a quantum computer by using preshared keys.

## Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8784.

## Copyright Notice

## Table of Contents



# 1. Introduction

Recent achievements in developing quantum computers demonstrate that it is probably feasible to build one that is cryptographically significant. If such a computer is implemented, many of the cryptographic algorithms and protocols currently in use would be insecure. A quantum computer would be able to solve Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH) problems in polynomial time [C2PQ], and this would imply that the security of existing IKEv2

[RFC7296] systems would be compromised. IKEv1 [RFC2409], when used with strong preshared keys, is not vulnerable to quantum attacks because those keys are one of the inputs to the key derivation function. If the preshared key has sufficient entropy and the Pseudorandom Function (PRF), encryption, and authentication transforms are quantum secure, then the resulting system is believed to be quantum secure -- that is, secure against classical attackers of today or future attackers with a quantum computer.

This document describes a way to extend IKEv2 to have a similar property; assuming that the two end systems share a long secret key, then the resulting exchange is quantum secure. By bringing post-quantum security to IKEv2, this document removes the need to use an obsolete version of IKE in order to achieve that security goal.

The general idea is that we add an additional secret that is shared between the initiator and the responder; this secret is in addition to the authentication method that is already provided within IKEv2. We stir this secret into the SK_d value, which is used to generate the key material (KEYMAT) for the Child Security Associations (SAs) and the SKEYSEED for the IKE SAs created as a result of the initial IKE SA rekey. This secret provides quantum resistance to the IPsec SAs and any subsequent IKE SAs. We also stir the secret into the SK_pi and SK_pr values; this allows both sides to detect a secret mismatch cleanly.

It was considered important to minimize the changes to IKEv2. The existing mechanisms to perform authentication and key exchange remain in place (that is, we continue to perform (EC)DH and potentially PKI authentication if configured). This document does not replace the authentication checks that the protocol does; instead, they are strengthened by using an additional secret key.

### 1.1. Requirements Language

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

## 2. Assumptions

We assume that each IKE peer has a list of Post-quantum Preshared Keys (PPKs) along with their identifiers (PPK_ID), and any potential IKE initiator selects which PPK to use with any specific responder. In addition, implementations have a configurable flag that determines whether this PPK is mandatory. This PPK is independent of the preshared key (if any) that the IKEv2 protocol uses to perform authentication (because the preshared key in IKEv2 is not used for any key derivation and thus doesn't protect against quantum computers). The PPK-specific configuration that is assumed to be on each node consists of the following tuple:

```
Peer, PPK, PPK_ID, mandatory_or_not
```

We assume the reader is familiar with the payload notation defined in Section 1.2 of [RFC7296].

# 3. Exchanges

If the initiator is configured to use a PPK with the responder (whether or not the use of the PPK is mandatory), then it **MUST** include a notification USE_PPK in the IKE_SA_INIT request message as follows:

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

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

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

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

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

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

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

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

```
SKEYSEED = prf(Ni | Nr, g^ir)
{SK_d' | SK_ai | SK_ar | SK_ei | SK_er | SK_pi' | SK_pr'}
                 = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr)

SK_d  = prf+ (PPK, SK_d')
SK_pi = prf+ (PPK, SK_pi')
SK_pr = prf+ (PPK, SK_pr')
```

That is, we use the standard IKEv2 key derivation process, except that the three resulting subkeys SK_d, SK_pi, and SK_pr (marked with primes in the formula above) are then run through the prf+ again, this time using the PPK as the key. The result is the unprimed versions of these keys, which are then used as inputs to subsequent steps of the IKEv2 exchange.

Using a prf+ construction ensures that it is always possible to get the resulting keys of the same size as the initial ones, even if the underlying PRF has an output size different from its key size. Note that at the time of this writing, all PRFs defined for use in IKEv2 (see the "Transform Type 2 - Pseudorandom Function Transform IDs" subregistry [IANA-IKEV2]) have an output size equal to the (preferred) key size. For such PRFs, only the first iteration of prf+ is needed:

```
SK_d  = prf (PPK, SK_d'  | 0x01)
SK_pi = prf (PPK, SK_pi' | 0x01)
SK_pr = prf (PPK, SK_pr' | 0x01)
```

Note that the PPK is used in SK_d, SK_pi, and SK_pr calculations only during the initial IKE SA setup. It **MUST NOT** be used when these subkeys are calculated as result of IKE SA rekey, resumption, or other similar operations.

The initiator then sends the IKE_AUTH request message, including the PPK_ID value as follows:

```
Initiator                       Responder
-----------------------------------------------------------------
HDR, SK {IDi, [CERT,] [CERTREQ,]
    [IDr,] AUTH, SAi2,
    TSi, TSr, N(PPK_IDENTITY, PPK_ID), [N(NO_PPK_AUTH)]}  --->
```

PPK_IDENTITY is a status notification with the type 16436; it has a protocol ID of 0, no SPI, and notification data that consists of the identifier PPK_ID.

A situation may happen when the responder has some PPKs but doesn't have a PPK with the PPK_ID received from the initiator. In this case, the responder cannot continue with the PPK (in particular, it cannot authenticate the initiator), but the responder could be able to continue with the normal IKEv2 protocol if the initiator provided its authentication data computed as in the normal IKEv2 without using PPKs. For this purpose, if using PPKs for communication with this

responder is optional for the initiator (based on the mandatory_or_not flag), then the initiator **MUST** include a NO_PPK_AUTH notification in the above message. This notification informs the responder that PPKs are optional and allows for authenticating the initiator without using PPKs.

NO_PPK_AUTH is a status notification with the type 16437; it has a protocol ID of 0 and no SPI. The Notification Data field contains the initiator's authentication data computed using SK_pi', which has been computed without using PPKs. This is the same data that would normally be placed in the Authentication Data field of an AUTH payload. Since the Auth Method field is not present in the notification, the authentication method used for computing the authentication data **MUST** be the same as the method indicated in the AUTH payload. Note that if the initiator decides to include the NO_PPK_AUTH notification, the initiator needs to perform authentication data computation twice, which may consume computation power (e.g., if digital signatures are involved).

When the responder receives this encrypted exchange, it first computes the values:

```
SKEYSEED = prf(Ni | Nr, g^ir)
{SK_d' | SK_ai | SK_ar | SK_ei | SK_er | SK_pi' | SK_pr'}
                = prf+ (SKEYSEED, Ni | Nr | SPIi | SPIr)
```

The responder then uses the SK_ei/SK_ai values to decrypt/check the message and then scans through the payloads for the PPK_ID attached to the PPK_IDENTITY notification. If no PPK_IDENTITY notification is found and the peers successfully exchanged USE_PPK notifications in the IKE_SA_INIT exchange, then the responder **MUST** send back an AUTHENTICATION_FAILED notification and then fail the negotiation.

If the PPK_IDENTITY notification contains a PPK_ID that is not known to the responder or is not configured for use for the identity from the IDi payload, then the responder checks whether using PPKs for this initiator is mandatory and whether the initiator included a NO_PPK_AUTH notification in the message. If using PPKs is mandatory or no NO_PPK_AUTH notification is found, then the responder **MUST** send back an AUTHENTICATION_FAILED notification and then fail the negotiation. Otherwise (when a PPK is optional and the initiator included a NO_PPK_AUTH notification), the responder **MAY** continue the regular IKEv2 protocol, except that it uses the data from the NO_PPK_AUTH notification as the authentication data (which usually resides in the AUTH payload) for the purpose of the initiator authentication. Note that the authentication method is still indicated in the AUTH payload.

Table 1 summarizes the above logic for the responder:

| Received USE_PPK | Received NO_PPK_AUTH | Configured with PPK | PPK is Mandatory | Action |
|---|---|---|---|---|
| No | * | No | * | Standard IKEv2 protocol |

| Received USE_PPK | Received NO_PPK_AUTH | Configured with PPK | PPK is Mandatory | Action |
|---|---|---|---|---|
| No | * | Yes | No | Standard IKEv2 protocol |
| No | * | Yes | Yes | Abort negotiation |
| Yes | No | No | * | Abort negotiation |
| Yes | Yes | No | Yes | Abort negotiation |
| Yes | Yes | No | No | Standard IKEv2 protocol |
| Yes | * | Yes | * | Use PPK |

*Table 1*

If a PPK is in use, then the responder extracts the corresponding PPK and computes the following values:

```
SK_d  = prf+ (PPK, SK_d')
SK_pi = prf+ (PPK, SK_pi')
SK_pr = prf+ (PPK, SK_pr')
```

The responder then continues with the IKE_AUTH exchange (validating the AUTH payload that the initiator included) as usual and sends back a response, which includes the PPK_IDENTITY notification with no data to indicate that the PPK is used in the exchange:

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

When the initiator receives the response, it checks for the presence of the PPK_IDENTITY notification. If it receives one, it marks the SA as using the configured PPK to generate SK_d, SK_pi, and SK_pr (as shown above); the content of the received PPK_IDENTITY (if any) **MUST** be ignored. If the initiator does not receive the PPK_IDENTITY, it **MUST** either fail the IKE SA negotiation sending the AUTHENTICATION_FAILED notification in the INFORMATIONAL exchange (if the PPK was configured as mandatory) or continue without using the PPK (if the PPK was not configured as mandatory and the initiator included the NO_PPK_AUTH notification in the request).

If the Extensible Authentication Protocol (EAP) is used in the IKE_AUTH exchange, then the initiator doesn't include the AUTH payload in the first request message; however, the responder sends back the AUTH payload in the first reply. The peers then exchange AUTH payloads after EAP is successfully completed. As a result, the responder sends the AUTH payload twice -- in the first and last IKE_AUTH reply message -- while the initiator sends the AUTH payload only in the last IKE_AUTH request. See more details about EAP authentication in IKEv2 in Section 2.16 of [RFC7296].

The general rule for using a PPK in the IKE_AUTH exchange, which covers the EAP authentication case too, is that the initiator includes a PPK_IDENTITY (and optionally a NO_PPK_AUTH) notification in the request message containing the AUTH payload. Therefore, in case of EAP, the responder always computes the AUTH payload in the first IKE_AUTH reply message without using a PPK (by means of SK_pr'), since PPK_ID is not yet known to the responder. Once the IKE_AUTH request message containing the PPK_IDENTITY notification is received, the responder follows the rules described above for the non-EAP authentication case.

```
Initiator                          Responder
-------------------------------------------------------------
HDR, SK {IDi, [CERTREQ,]
    [IDr,] SAi2,
    TSi, TSr}  -->
                                <--  HDR, SK {IDr, [CERT,] AUTH,
                                         EAP}
HDR, SK {EAP}  -->
                                <--  HDR, SK {EAP (success)}
HDR, SK {AUTH,
    N(PPK_IDENTITY, PPK_ID)
    [, N(NO_PPK_AUTH)]}  -->
                                <--  HDR, SK {AUTH, SAr2, TSi, TSr
                                     [, N(PPK_IDENTITY)]}
```

Note that the diagram above shows both the cases when the responder uses a PPK and when it chooses not to use it (provided the initiator has included the NO_PPK_AUTH notification); thus, the responder's PPK_IDENTITY notification is marked as optional. Also, note that the IKE_SA_INIT exchange using a PPK is as described above (including exchange of the USE_PPK notifications), regardless of whether or not EAP is employed in the IKE_AUTH.

# 4. Upgrade Procedure

This algorithm was designed so that someone can introduce PPKs into an existing IKE network without causing network disruption.

In the initial phase of the network upgrade, the network administrator would visit each IKE node and configure:

- The set of PPKs (and corresponding PPK_IDs) that this node would need to know.
- The PPK that will be used for each peer that this node would initiate to.

- The value "false" for the mandatory_or_not flag for each peer that this node would initiate to (thus indicating that the use of PPKs is not mandatory).

With this configuration, the node will continue to operate with nodes that have not yet been upgraded. This is due to the USE_PPK notification and the NO_PPK_AUTH notification; if the initiator has not been upgraded, it will not send the USE_PPK notification (and so the responder will know that the peers will not use a PPK). If the responder has not been upgraded, it will not send the USE_PPK notification (and so the initiator will know to not use a PPK). If both peers have been upgraded but the responder isn't yet configured with the PPK for the initiator, then the responder could continue with the standard IKEv2 protocol if the initiator sent a NO_PPK_AUTH notification. If both the responder and initiator have been upgraded and properly configured, they will both realize it, and the Child SAs will be quantum secure.

As an optional second step, after all nodes have been upgraded, the administrator should then go back through the nodes and mark the use of a PPK as mandatory. This will not affect the strength against a passive attacker, but it would mean that an active attacker with a quantum computer (which is sufficiently fast to be able to break the (EC)DH in real time) would not be able to perform a downgrade attack.

# 5. PPK

## 5.1. PPK_ID Format

This standard requires that both the initiator and the responder have a secret PPK value, with the responder selecting the PPK based on the PPK_ID that the initiator sends. In this standard, both the initiator and the responder are configured with fixed PPK and PPK_ID values and perform the lookup based on the PPK_ID value. It is anticipated that later specifications will extend this technique to allow dynamically changing PPK values. To facilitate such an extension, we specify that the PPK_ID the initiator sends will have its first octet be the PPK_ID type value. This document defines two values for the PPK_ID type:

- PPK_ID_OPAQUE (1) - For this type, the format of the PPK_ID (and the PPK itself) is not specified by this document; it is assumed to be mutually intelligible by both the initiator and the responder. This PPK_ID type is intended for those implementations that choose not to disclose the type of PPK to active attackers.
- PPK_ID_FIXED (2) - In this case, the format of the PPK_ID and the PPK are fixed octet strings; the remaining bytes of the PPK_ID are a configured value. We assume that there is a fixed mapping between PPK_ID and PPK, which is configured locally to both the initiator and the responder. The responder can use the PPK_ID to look up the corresponding PPK value. Not all implementations are able to configure arbitrary octet strings; to improve the potential interoperability, it is recommended that, in the PPK_ID_FIXED case, both the PPK and the PPK_ID strings be limited to the base64 character set [RFC4648].

## 5.2. Operational Considerations

The need to maintain several independent sets of security credentials can significantly complicate a security administrator's job and can potentially slow down widespread adoption of this specification. It is anticipated that administrators will try to simplify their job by decreasing the number of credentials they need to maintain. This section describes some of the considerations for PPK management.

### 5.2.1. PPK Distribution

PPK_IDs of the type PPK_ID_FIXED (and the corresponding PPKs) are assumed to be configured within the IKE device in an out-of-band fashion. While the method of distribution is a local matter and is out of scope of this document or IKEv2, [RFC6030] describes a format for the transport and provisioning of symmetric keys. That format could be reused using the PIN profile (defined in Section 10.2 of [RFC6030]) with the "Id" attribute of the <Key> element being the PPK_ID (without the PPK_ID type octet for a PPK_ID_FIXED) and the <Secret> element containing the PPK.

### 5.2.2. Group PPK

This document doesn't explicitly require that the PPK be unique for each pair of peers. If this is the case, then this solution provides full peer authentication, but it also means that each host must have as many independent PPKs as peers it is going to communicate with. As the number of peers grows, the PPKs will not scale.

It is possible to use a single PPK for a group of users. Since each peer uses classical public key cryptography in addition to a PPK for key exchange and authentication, members of the group can neither impersonate each other nor read each other's traffic unless they use quantum computers to break public key operations. However, group members can record any traffic they have access to that comes from other group members and decrypt it later, when they get access to a quantum computer.

In addition, the fact that the PPK is known to a (potentially large) group of users makes it more susceptible to theft. When an attacker equipped with a quantum computer gets access to a group PPK, all communications inside the group are revealed.

For these reasons, using a group PPK is **NOT RECOMMENDED**.

### 5.2.3. PPK-Only Authentication

If quantum computers become a reality, classical public key cryptography will provide little security, so administrators may find it attractive not to use it at all for authentication. This will reduce the number of credentials they need to maintain because they only need to maintain PPK credentials. Combining group PPK and PPK-only authentication is **NOT RECOMMENDED** since, in this case, any member of the group can impersonate any other member, even without the help of quantum computers.

PPK-only authentication can be achieved in IKEv2 if the NULL Authentication method [RFC7619] is employed. Without PPK, the NULL Authentication method provides no authentication of the peers; however, since a PPK is stirred into the SK_pi and the SK_pr, the peers become authenticated if a PPK is in use. Using PPKs **MUST** be mandatory for the peers if they advertise support for PPKs in IKE_SA_INIT and use NULL Authentication. Additionally, since the peers are authenticated via PPKs, the ID Type in the IDi/IDr payloads **SHOULD NOT** be ID_NULL, despite using the NULL Authentication method.

## 6. Security Considerations

A critical consideration is how to ensure the randomness of this post-quantum preshared key. Quantum computers are able to perform Grover's algorithm [GROVER]; that effectively halves the size of a symmetric key. In addition, an adversary impersonating the server, even with a conventional computer, can perform a dictionary search over plausible post-quantum preshared key values. The strongest practice is to ensure that any post-quantum preshared key contains at least 256 bits of entropy; this will provide 128 bits of post-quantum security, while providing security against conventional dictionary attacks. That provides the security equivalent to Category 5 as defined in the NIST Post-Quantum Cryptography Call for Proposals [NISTPQCFP]. Deriving a post-quantum preshared key from a password, name, or other low-entropy source is not secure because of these known attacks.

With this protocol, the computed SK_d is a function of the PPK. Assuming that the PPK has sufficient entropy (for example, at least $2^{256}$ possible values), even if an attacker was able to recover the rest of the inputs to the PRF function, it would be infeasible to use Grover's algorithm with a quantum computer to recover the SK_d value. Similarly, all keys that are a function of SK_d, which include all Child SA keys and all keys for subsequent IKE SAs (created when the initial IKE SA is rekeyed), are also quantum secure (assuming that the PPK was of high enough entropy and that all the subkeys are sufficiently long).

An attacker with a quantum computer that can decrypt the initial IKE SA has access to all the information exchanged over it, such as identities of the peers, configuration parameters, and all negotiated IPsec SA information (including traffic selectors), with the exception of the cryptographic keys used by the IPsec SAs, which are protected by the PPK.

Deployments that treat this information as sensitive or that send other sensitive data (like cryptographic keys) over IKE SAs **MUST** rekey the IKE SA before the sensitive information is sent to ensure this information is protected by the PPK. It is possible to create a childless IKE SA as specified in [RFC6023]. This prevents Child SA configuration information from being transmitted in the original IKE SA that is not protected by a PPK. Some information related to IKE SA that is sent in the IKE_AUTH exchange, such as peer identities, feature notifications, vendor IDs, etc., cannot be hidden from the attack described above, even if the additional IKE SA rekey is performed.

In addition, the policy **SHOULD** be set to negotiate only quantum-secure symmetric algorithms; while this RFC doesn't claim to give advice as to what algorithms are secure (as that may change based on future cryptographical results), below is a list of defined IKEv2 and IPsec algorithms that should not be used, as they are known to provide less than 128 bits of post-quantum security:

- Any IKEv2 encryption algorithm, PRF, or integrity algorithm with a key size less than 256 bits.
- Any ESP transform with a key size less than 256 bits.
- PRF_AES128_XCBC and PRF_AES128_CBC: even though they can use as input a key of arbitrary size, such input keys are converted into a 128-bit key for internal use.

Section 3 requires the initiator to abort the initial exchange if using PPKs is mandatory for it but the responder does not include the USE_PPK notification in the response. In this situation, when the initiator aborts the negotiation, it leaves a half-open IKE SA on the responder (because IKE_SA_INIT completes successfully from the responder's point of view). This half-open SA will eventually expire and be deleted, but if the initiator continues its attempts to create IKE SA with a high enough rate, then the responder may consider it a denial-of-service (DoS) attack and take protective measures (see [RFC8019] for more details). In this situation, it is **RECOMMENDED** that the initiator cache the negative result of the negotiation and not attempt to create it again for some time. This period of time may vary, but it is believed that waiting for at least few minutes will not cause the responder to treat it as a DoS attack. Note that this situation would most likely be a result of misconfiguration, and some reconfiguration of the peers would probably be needed.

If using PPKs is optional for both peers and they authenticate themselves using digital signatures, then an attacker in between, equipped with a quantum computer capable of breaking public key operations in real time, is able to mount a downgrade attack by removing the USE_PPK notification from the IKE_SA_INIT and forging digital signatures in the subsequent exchange. If using PPKs is mandatory for at least one of the peers or if a preshared key mode is used for authentication, then the attack will be detected and the SA won't be created.

If using PPKs is mandatory for the initiator, then an attacker able to eavesdrop and inject packets into the network can prevent creation of an IKE SA by mounting the following attack. The attacker intercepts the initial request containing the USE_PPK notification and injects a forged response containing no USE_PPK. If the attacker manages to inject this packet before the responder sends a genuine response, then the initiator would abort the exchange. To thwart this kind of attack, it is **RECOMMENDED** that, if using PPKs is mandatory for the initiator and the received response doesn't contain the USE_PPK notification, the initiator not abort the exchange immediately. Instead, it waits for more response messages, retransmitting the request as if no responses were received at all, until either the received message contains the USE_PPK notification or the exchange times out (see Section 2.4 of [RFC7296] for more details about retransmission timers in IKEv2). If none of the received responses contains USE_PPK, then the exchange is aborted.

If using a PPK is optional for both peers, then in case of misconfiguration (e.g., mismatched PPK_ID), the IKE SA will be created without protection against quantum computers. It is advised that if a PPK was configured but was not used for a particular IKE SA, then implementations **SHOULD** audit this event.

## 7. IANA Considerations

This document defines three new Notify Message Types in the "IKEv2 Notify Message Types - Status Types" subregistry under the "Internet Key Exchange Version 2 (IKEv2) Parameters" registry [IANA-IKEV2]:

| Value | NOTIFY MESSAGES - STATUS TYPES | Reference |
|---|---|---|
| 16435 | USE_PPK | RFC 8784 |
| 16436 | PPK_IDENTITY | RFC 8784 |
| 16437 | NO_PPK_AUTH | RFC 8784 |

*Table 2*

Per this document, IANA has created a new subregistry titled "IKEv2 Post-quantum Preshared Key ID Types" under the "Internet Key Exchange Version 2 (IKEv2) Parameters" registry [IANA-IKEV2]. This new subregistry is for the PPK_ID types used in the PPK_IDENTITY notification defined in this specification. The initial contents of the new subregistry are as follows:

| Value | PPK_ID Type | Reference |
|---|---|---|
| 0 | Reserved | RFC 8784 |
| 1 | PPK_ID_OPAQUE | RFC 8784 |
| 2 | PPK_ID_FIXED | RFC 8784 |
| 3-127 | Unassigned | RFC 8784 |
| 128-255 | Reserved for Private Use | RFC 8784 |

*Table 3*

The PPK_ID type value 0 is reserved; values 3-127 are to be assigned by IANA; and values 128-255 are for private use among mutually consenting parties. To register new PPK_IDs in the Unassigned range, a type name, a value between 3 and 127, and a reference specification need to be defined. Changes and additions to the Unassigned range of this registry are made using the Expert Review policy [RFC8126]. Changes and additions to the Reserved for Private Use range of this registry are made using the Private Use policy [RFC8126].

# 8. References

## 8.1. Normative References

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

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

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

## 8.2. Informative References

**[C2PQ]** Hoffman, P., "The Transition from Classical to Post-Quantum Cryptography", Work in Progress, Internet-Draft, draft-hoffman-c2pq-07, 26 May 2020, <https://tools.ietf.org/html/draft-hoffman-c2pq-07>.

**[GROVER]** Grover, L., "A Fast Quantum Mechanical Algorithm for Database Search", STOC '96: Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, pp. 212-219", DOI 10.1145/237814.237866, July 1996, <https://doi.org/10.1145/237814.237866>.

**[IANA-IKEV2]** IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", <https://www.iana.org/assignments/ikev2-parameters/>.

**[NISTPQCFP]** NIST, "Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process", December 2016, <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>.

**[RFC2409]** Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <https://www.rfc-editor.org/info/rfc2409>.

**[RFC4648]** Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

**[RFC6023]** Nir, Y., Tschofenig, H., Deng, H., and R. Singh, "A Childless Initiation of the Internet Key Exchange Version 2 (IKEv2) Security Association (SA)", RFC 6023, DOI 10.17487/RFC6023, October 2010, <https://www.rfc-editor.org/info/rfc6023>.

**[RFC6030]** Hoyer, P., Pei, M., and S. Machani, "Portable Symmetric Key Container (PSKC)", RFC 6030, DOI 10.17487/RFC6030, October 2010, <https://www.rfc-editor.org/info/rfc6030>.

**[RFC7619]** Smyslov, V. and P. Wouters, "The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7619, DOI 10.17487/RFC7619, August 2015, <https://www.rfc-editor.org/info/rfc7619>.

**[RFC8019]** Nir, Y. and V. Smyslov, "Protecting Internet Key Exchange Protocol Version 2 (IKEv2) Implementations from Distributed Denial-of-Service Attacks", RFC 8019, DOI 10.17487/RFC8019, November 2016, <https://www.rfc-editor.org/info/rfc8019>.

**[RFC8126]** Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

# Appendix A. Discussion and Rationale

The primary goal of this document is to augment the IKEv2 protocol to provide protection against quantum computers without requiring novel cryptographic algorithms. The idea behind this document is that while a quantum computer can easily reconstruct the shared secret of an (EC)DH exchange, it cannot as easily recover a secret from a symmetric exchange. This document makes the SK_d (and thus also the IPsec KEYMAT and any subsequent IKE SA's SKEYSEED) depend on both the symmetric PPK and the Diffie-Hellman exchange. If we assume that the attacker knows everything except the PPK during the key exchange and there are $2^n$ plausible PPKs, then a quantum computer (using Grover's algorithm) would take $O(2^{n/2})$ time to recover the PPK. So, even if the (EC)DH can be trivially solved, the attacker still can't recover any key material (except for the SK_ei, SK_er, SK_ai, and SK_ar values for the initial IKE exchange) unless they can find the PPK, which is too difficult if the PPK has enough entropy (for example, 256 bits). Note that we do allow an attacker with a quantum computer to rederive the keying material for the initial IKE SA; this was a compromise to allow the responder to select the correct PPK quickly.

Another goal of this protocol is to minimize the number of changes within the IKEv2 protocol, in particular, within the cryptography of IKEv2. By limiting our changes to notifications and only adjusting the SK_d, SK_pi, and SK_pr, it is hoped that this would be implementable, even on systems that perform most of the IKEv2 processing in hardware.

A third goal is to be friendly to incremental deployment in operational networks for which we might not want to have a global shared key or for which quantum-secure IKEv2 is rolled out incrementally. This is why we specifically try to allow the PPK to be dependent on the peer and why we allow the PPK to be configured as optional.

A fourth goal is to avoid violating any of the security properties provided by IKEv2.

# Acknowledgements

We would like to thank Tero Kivinen, Paul Wouters, Graham Bartlett, Tommy Pauly, Quynh Dang, and the rest of the IPSECME Working Group for their feedback and suggestions for the scheme.

## Authors' Addresses

**Scott Fluhrer**
Cisco Systems
Email: sfluhrer@cisco.com

**Panos Kampanakis**
Cisco Systems
Email: pkampana@cisco.com

**David McGrew**
Cisco Systems
Email: mcgrew@cisco.com

**Valery Smyslov**
ELVIS-PLUS
Phone: +7 495 276 0211
Email: svan@elvis.ru