Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 31, 2018

S. Fluhrer
D. McGrew
P. Kampanakis
Cisco Systems
V. Smyslov
ELVIS-PLUS

February 27, 2018

Postquantum Preshared Keys for IKEv2
draft-ietf-ipsecme-qr-ikev2-02

Abstract

The possibility of Quantum Computers pose a serious challenge to cryptography algorithms deployed widely today. IKEv2 is one example of a cryptosystem that could be broken; someone storing VPN communications today could decrypt them at a later time when a Quantum Computer is available. It is anticipated that IKEv2 will be extended to support quantum secure key exchange algorithms; however that is not likely to happen in the near term. To address this problem before then, this document describes an extension of IKEv2 to allow it to be resistant to a Quantum Computer, by using preshared keys.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 31, 2018.

Copyright Notice

Table of Contents

1. Introduction

It is an open question whether or not it is feasible to build a Quantum Computer (and if so, when one might be implemented), but if it is, many of the cryptographic algorithms and protocols currently in use would be insecure. A Quantum Computer would be able to solve DH and ECDH problems in polynomial time [I-D.hoffman-c2pq], and this would imply that the security of existing IKEv2 [RFC7296] systems would be compromised. IKEv1 [RFC2409], when used with strong preshared keys, is not vulnerable to quantum attacks, because those keys are one of the inputs to the key derivation function. If the preshared key has sufficient entropy and the PRF, encryption and authentication transforms are postquantum secure, then the resulting system is believed to be quantum resistant, that is, invulnerable to an attacker 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 resistant. By bringing postquantum security to IKEv2, this note removes the need to use an obsolete version of the Internet Key Exchange in order to achieve that security goal.

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

It was considered important to minimize the changes to IKEv2. The existing mechanisms to do authentication and key exchange remain in place (that is, we continue to do (EC)DH, and potentially a PKI authentication if configured). This document does not replace the authentication checks that the protocol does; instead, it is done as a parallel check.

1.1. Changes

RFC EDITOR PLEASE DELETE THIS SECTION.

Changes in this draft in each version iterations.

draft-ietf-ipsecme-qr-ikev2-02

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

draft-ietf-ipsecme-qr-ikev2-01

- Nits and minor fixes.
- prf is replaced with prf+ for the SK_d and SK_pi/r calculations.

- Clarified using PPK in case of EAP authentication.

- PPK_SUPPORT notification is changed to USE_PPK to better reflect its purpose.

draft-ietf-ipsecme-qr-ikev2-00

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

draft-fluhrer-qr-ikev2-05

- Nits and editorial fixes.

- Made PPK_ID format and PPK Distributions subsection of the PPK section. Also added an Operational Considerations section.

- Added comment about Child SA rekey in the Security Considerations section.

- Added NO_PPK_AUTH to solve the cases where a PPK_ID is not configured for a responder.

- Various text changes and clarifications.

- Expanded Security Considerations section to describe some security concerns and how they should be addressed.

draft-fluhrer-qr-ikev2-03

- Modified how we stir the PPK into the IKEv2 secret state.

- Modified how the use of PPKs is negotiated.

draft-fluhrer-qr-ikev2-02

- Simplified the protocol by stirring in the preshared key into the child SAs; this avoids the problem of having the responder decide which preshared key to use (as it knows the initiator identity at that point); it does mean that someone with a Quantum Computer can recover the initial IKE negotiation.

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

draft-fluhrer-qr-ikev2-01

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

draft-fluhrer-qr-ikev2-00

- We switched from using vendor ID's to transmit the additional data to notifications.
- We added a mandatory cookie exchange to allow the server to communicate to the client before the initial exchange.
- We added algorithm agility by having the server tell the client what algorithm to use in the cookie exchange.
- We have the server specify the PPK Indicator Input, which allows the server to make a trade-off between the efficiency for the search of the clients PPK, and the anonymity of the client.
- We now use the negotiated PRF (rather than a fixed HMAC-SHA256) to transform the nonces during the KDF.

1.2. Requirements Language

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

2. Assumptions

We assume that each IKE peer has a list of Postquantum Preshared Keys (PPK) along with their identifiers (PPK_ID), and any potential IKE initiator has a selection of which PPK to use with any specific responder. In addition, implementations have a configurable flag that determines whether this postquantum preshared key is mandatory. This PPK is independent of the preshared key (if any) that the IKEv2 protocol uses to perform authentication. The PPK specific configuration that is assumed on each peer consists of the following tuple:

Peer, PPK, PPK_ID, mandatory_or_not

3. Exchanges

If the initiator is configured to use a postquantum preshared key with the responder (whether or not the use of the PPK is mandatory), then he will include a notification USE_PPK in the IKE_SA_INIT request message as follows:

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

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

If the initiator needs to resend this initial message with a cookie (because the responder response included a COOKIE notification), then the resend would include the USE_PPK notification if the original message did.

If the responder does not support this specification or does not have any PPK configured, then she ignores the received notification and continues with the IKEv2 protocol as normal. Otherwise the responder checks if she has a PPK configured, and if she does, then 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, he checks whether the responder included the USE_PPK notification. If the responder did not and the flag mandatory_or_not indicates that using PPKs is mandatory for communication with this responder, then the initiator MUST abort the exchange. This situation may happen in case of misconfiguration, when the initiator believes he has a mandatory to use PPK for the responder, while the responder either doesn't support PPKs at all or doesn't have any PPK configured for the initiator. See Section 6 for discussion of the possible impacts of this situation.

If the responder did not include the USE_PPK notification and using PPKs for this 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, she computes this modification of the standard IKEv2 key derivation:

```
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 subkeys SK_d, SK_pi, SK_pr are run through the prf+ again, this time using the PPK as the key. Using prf+ construction ensures that it is always possible to get the resulting keys of the same size as the initial ones, even if the underlying prf has output size different from its key size. Note, that at the time this document was written, all prfs defined for use in IKEv2 [IKEV2-IANA-PRFS] had output size equal to the (preferred) key size. For such prfs only the first iteration of prf+ is needed:

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

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

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

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

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

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

optional for the initiator, then the initiator MAY include a notification NO_PPK_AUTH in the above message.

NO_PPK_AUTH is a status notification with the type 16437; it has a protocol ID of 0 and no SPI. A notification data consists of the initiator's authentication data computed using SK_pi' (i.e. the data that computed without using PPKs and would normally be placed in the AUTH payload). Authentication Method for computing the authentication data MUST be the same as indicated in the AUTH payload and is not included in the notification. Note that if the initiator decides to include NO_PPK_AUTH notification, then it means that the initiator needs to perform authentication data computation twice that may consume substantial computation power (e.g. if digital signatures are involved).

When the responder receives this encrypted exchange, she 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 )
```

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

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

This table summarizes the above logic by the responder:

Received USE_PPK	Received NO_PPK_AUTH	Have PPK	PPK Mandatory	Action
No	*	No	*	Standard IKEv2 protocol
No	*	Yes	No	Standard IKEv2 protocol
No	*	Yes	Yes	Abort negotiation
Yes	No	No	*	Abort negotiation
Yes	Yes	No	Yes	Abort negotiation
Yes	Yes	No	No	Standard IKEv2 protocol
Yes	*	Yes	*	Use PPK

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

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

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

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

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

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

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

4. Upgrade procedure

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

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

- The set of PPKs (and corresponding PPK_IDs) that this node would need to know.
- For each peer that this node would initiate to, which PPK will be used.
- That the use of PPK is currently not mandatory.

With this configuration, the node will continue to operate with nodes that have not yet been upgraded. This is due to the USE_PPK notify and the NO_PPK_AUTH notify; if the initiator has not been upgraded,

he will not send the USE_PPK notify (and so the responder will know that we will not use a PPK). If the responder has not been upgraded, she will not send the USE_PPK notify (and so the initiator will know to not use a PPK). If both peers have been upgraded, but the responder isn't yet configured with the PPK for the initiator, then the responder could do standard IKEv2 protocol if the initiator sent NO_PPK_AUTH notification. If both the responder and initiator have been upgraded and properly configured, they will both realize it, and in that case, the link will be quantum secure.

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

5. PPK

5.1. PPK_ID format

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

- PPK_ID_OPAQUE (1) - for this type the format of the PPK_ID (and the PPK itself) is not specified by this document; it is assumed to be mutually intelligible by both by initiator and the responder. This PPK_ID type is intended for those implementations that choose not to disclose the type of PPK to active attackers.

- PPK_ID_FIXED (2) - in this case the format of the PPK_ID and the PPK are fixed octet strings; the remaining bytes of the PPK_ID are a configured value. We assume that there is a fixed mapping between PPK_ID and PPK, which is configured locally to both the initiator and the responder. The responder can use to do a look up the passed PPK_ID value to determine 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, namely the 64 characters 0-9, A-Z, a-z, + and /.

The PPK_ID type value 0 is reserved; values 3-127 are reserved for IANA; values 128-255 are for private use among mutually consenting parties.

5.2. Operational Considerations

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

5.2.1. PPK Distribution

PPK_IDs of the type PPK_ID_FIXED (and the corresponding PPKs) are assumed to be configured within the IKE device in an out-of-band fashion. While the method of distribution is a local matter and out of scope of this document or IKEv2, [RFC6030] describes a format for symmetric key exchange. That format could be reused with the Key Id field being the PPK_ID (without the PPK_ID Type octet for a PPK_ID_FIXED), the PPK being the secret, and the algorithm ("Algorithm=urn:ietf:params:xml:ns:keyprov:pskc:pin") as PIN.

5.2.2. Group PPK

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

Even though it is NOT RECOMMENDED, it is possible to use a single PPK for a group of users. Since each peer uses classical public key cryptography in addition to PPK for key exchange and authentication, members of the group can neither impersonate each other nor read other's traffic, unless they use Quantum Computers to break public key operations.

Although it's probably safe to use group PPK in short term, the fact, that the PPK is known to a (potentially large) group of users makes it more susceptible to theft. If an attacker equipped with a Quantum Computer got access to a group PPK, then all the communications inside the group are revealed.

5.2.3. PPK-only Authentication

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

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

6. Security Considerations

Quantum computers are able to perform Grover's algorithm; that effectively halves the size of a symmetric key. Because of this, the user SHOULD ensure that the postquantum preshared key used has at least 256 bits of entropy, in order to provide a 128-bit security level.

With this protocol, the computed SK_d is a function of the PPK, and assuming that the PPK has sufficient entropy (for example, at least 2^256 possible values), then even if an attacker was able to recover the rest of the inputs to the prf function, it would be infeasible to use Grover's algorithm with a Quantum Computer to recover the SK_d value. Similarly, every child SA key is a function of SK_d, hence all the keys for all the child SAs are also quantum resistant (assuming that the PPK was high entropy and secret, and that all the subkeys are sufficiently long).

Although this protocol preserves all the security properties of IKEv2 against adversaries with conventional computers, it allows an adversary with a Quantum Computer to decrypt all traffic encrypted with the initial IKE SA. In particular, it allows the adversary to recover the identities of both sides. If there is IKE traffic other than the identities that need to be protected against such an adversary, implementations MAY rekey the initial IKE SA immediately after negotiating it to generate a new SKEYSEED from the postquantum SK_d. This would reduce the amount of data available to an attacker with a Quantum Computer.

Alternatively, an initial IKE SA (which is used to exchange identities) can take place, perhaps by using the protocol documented in [RFC6023]. After the childless IKE SA is created, implementations would immediately create a new IKE SA (which is used to exchange everything else) by using a rekey mechanism for IKE SAs. Because the rekeyed IKE SA keys are a function of SK_d, which is a function of the PPK (among other things), traffic protected by that IKE SA is secure against Quantum capable adversaries.

If some sensitive information (like keys) is to be transferred over IKE SA, then implementations MUST rekey the initial IKE SA before sending this information to get protection against Quantum Computers.

In addition, the policy SHOULD be set to negotiate only quantum-resistant symmetric algorithms; while this RFC doesn't claim to give advise 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 not to be quantum resistant

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

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

- PRF_AES128_XCBC and PRF_AES128_CBC; even though they are defined to be able to use an arbitrary key size, they convert it into a 128-bit key internally.

Section 3 requires the initiator to abort the initial exchange if using PPKs is mandatory for it, but the responder didn't include the USE_PPK notification in the response. In this situation when the initiator aborts negotiation he leaves half-open IKE SA on the responder (because IKE_SA_INIT completes successfully from responder's point of view). This half-open SA will eventually expire and be deleted, but if the initiator continues its attempts to create IKE SA with a high enough rate, then the responder may consider it as a Denial-of-Service attack and take some measures (see [RFC8019] for more detail). It is RECOMMENDED that implementations in this situation cache the negative result of negotiation for some time and don't make attempts to create it again for some time, because this is a result of misconfiguration and probably some re-configuration of the peers is needed.

If using PPKs is optional for both peers and they authenticate themselves using digital signatures, then an attacker in between, equipped with a Quantum Computer capable of breaking public key operations in real time, is able to mount downgrade attack by

removing USE_PPK notification from the IKE_SA_INIT and forging digital signatures in the subsequent exchange. If using PPKs is mandatory for at least one of the peers or PSK is used for authentication, then the attack will be detected and the SA won't be created.

If using PPKs is mandatory for the initiator, then an attacker capable to eavesdrop and to inject packets into the network can prevent creating IKE SA by mounting the following attack. The attacker intercepts the the initial request containing the USE_PPK notification and injects the forget response containing no USE_PPK. If the attacker manages to inject this packet before the responder sends a genuine response, then the initiator would abort the exchange. To thwart this kind of attack it is RECOMMENDED, that if using PPKs is mandatory for the initiator and the received response doesn't contain the USE_PPK notification, then the initiator doesn't abort exchange immediately, but instead waits some time for more responses (possibly retransmitting the request). If all the received responses contain no USE_PPK, then the exchange is aborted.

7. IANA Considerations

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

16435	USE_PPK
16436	PPK_IDENTITY
16437	NO_PPK_AUTH

This document also creates a new IANA registry for the PPK_ID types. The initial values of this registry are:

PPK_ID Type	Value
Reserved	0
PPK_ID_OPAQUE	1
PPK_ID_FIXED	2
Unassigned	3-127
Reserved for private use	128-255

Changes and additions to this registry are by Expert Review [RFC5226].

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>.

8.2. Informational References

[I-D.hoffman-c2pq] Hoffman, P., "The Transition from Classical to Post-Quantum Cryptography", draft-hoffman-c2pq-02 (work in progress), August 2017.

[IKEV2-IANA-PRFS] "Internet Key Exchange Version 2 (IKEv2) Parameters, Transform Type 2 - Pseudorandom Function Transform IDs", <https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-6>.

[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>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[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>.

Appendix A. Discussion and Rationale

The idea behind this document is that while a Quantum Computer can easily reconstruct the shared secret of an (EC)DH exchange, they cannot as easily recover a secret from a symmetric exchange. This makes the SK_d, and hence the IPsec KEYMAT and any child SA's SKEYSEED, depend on both the symmetric PPK, and also the Diffie-Hellman exchange. If we assume that the attacker knows everything except the PPK during the key exchange, and there are 2^n plausible PPKs, then a Quantum Computer (using Grover's algorithm) would take O(2^(n/2)) time to recover the PPK. So, even if the (EC)DH can be trivially solved, the attacker still can't recover any key material (except for the SK_ei, SK_er, SK_ai, SK_ar values for the initial IKE exchange) unless they can find the PPK, which is too difficult if the PPK has enough entropy (for example, 256 bits). Note that we do allow an attacker with a Quantum Computer to rederive the keying material for the initial IKE SA; this was a compromise to allow the responder to select the correct PPK quickly.

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

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

A fourth goal was to avoid violating any of the security goals of IKEv2.

Appendix B. Acknowledgements

We would like to thank Tero Kivinen, Paul Wouters, Graham Bartlett 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

David McGrew
Cisco Systems

Email: mcgrew@cisco.com

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

Valery Smyslov
ELVIS-PLUS

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