                   Postquantum Preshared Keys for IKEv2
                        draft-fluhrer-qr-ikev2-04

Abstract

   The possibility of quantum computers pose a serious challenge to
   cryptography algorithms 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 http://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 October 21, 2017.

carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

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 might one 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, and this would imply that the security of
   existing IKEv2 systems would be compromised. IKEv1 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, believed to be
   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 in 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 in 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 does not replace the
authentication checks that the protocol does; instead, it is done as
a parallel check.

1.1. Changes

Changes in this draft from the previous versions

draft-03

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

- Modified how the use of PPKs is negotiated

draft-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 negotation.

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

draft-01

- Added explicit guidance as to what IKE and IPsec algorithms are
Quantum Resistant

draft-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 with any specific
responder. In addition, the implementation has 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.

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 optional),
then it will include a notify payload in the initial exchange as
follows:

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

N(PPK_SUPPORT) is a status notification payload with the type [TBA];
it has a protocol ID of 0, and 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 PPK_SUPPORT notification if the original
message did.

When the responder receives this initial exchange with the notify,
then it MUST check if has a PPK configured. If it does, it MUST

reply with the IKE initial exchange including a notification in
response.

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

If the responder does not have a PPK configured, then it continues
with the IKE protocol as normal, not including the notify.

When the initiator receives this reply, it checks whether the
responder included the PPK_SUPPORT notify. If the responder did not,
then the initiator MUST either proceed with the standard IKE
negotiation (without using a PPK), or abort the exchange (for
example, because the initiator has the PPK marked as mandatory). If
the responder did include the PPK_SUPPORT notify, then it selects a
PPK, along with its identifier PPK_id. Then, it computes this
modification of the standard IKE 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 IKE 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.

The initiator then sends the initial encrypted 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(PPK_IDENITY) is a status notification payload with the type [TBA];
it has a protocol ID of 0, and no SPI and has a notification data
that consists of the identifier PPK_id.

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 )
```

It then uses the SK_ei value to decrypt the message; and then finds
the PPK_id value attached to the notify. It then scans through the
payload for the PPK_id attached to the N(PPK_IDENTITY); if it has no
such PPK, it fails the negotiation. If it does have a PPK with that
identity, it further computes:

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

And computes the enchange (validating the AUTH payload that the
initiator included) as standard.

This table summarizes the above logic by the responder

Received PPK_SUPPORT	Have PPK	PPK Mandatory	Action
No	No	*	Standard IKE protocol
No	Yes	No	Standard IKE protocol
No	Yes	Yes	Abort negotiation
Yes	No	*	Standard IKE protocol
Yes	Yes	*	Include PPK_SUPPORT

When the initiator receives the response, then (if it is configured
to use a PPK with the responder), then it checks for the presense of
the notification. If it receives one, it marks the SA as using the
configured PPK to generate SK_d, SK_pi, SK_pr (as shown above); if it
does not receive one, it MUST either abort the exchange (if the PPK
was configured as mandatory), or it MUST continue without using the
PPK (if the PPK was configured as optional).

If the initial exchange had PPK_SUPPORT sent by both the initiator
and the responder, and the initiator does not include a PPK_NOTIFY
notification, then the responder SHOULD fail the exchange.

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 were 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). However, this quantum

resistance does not extend to the initial SK_ei, SK_er keys; an
implementation MAY rekey the initial IKE SA immediately after
negotiating it; this would reduce the amount of data available to an
attacker with a Quantum Computer.

4. 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 initial standard, both the
   initiator and the responder are configured with fixed PPK and PPK_ID
   values, and do the look up based on that. 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 that the initiator sends will have its first octet be
   the PPK ID Type value, which is encoded as follows:

        PPK ID Type   Value

        PPK_ID_OPAQUE   0
        PPK_ID_FIXED   1
        RESERVED TO IANA   2-127
        Reserved for private use 128-255

   For PPK_ID_OPAQUE, 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.

   For PPK_ID_FIXED, 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 values 2-127 are reserved for IANA; values 128-255
   are for private use among mutually consenting parties.

5. PPK Distribution

   PPK_id's 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, one
   suggestion would be to reuse the format within [RFC6030], with the
   Key Id field being the PPK_ID (without the 0x01 prefix for a
   PPK_ID_FIXED), and with the PPK being the secret, and the algorithm
   as PIN ("Algorithm=urn:ietf:params:xml:ns:keyprov:pskc:pin").

6. 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_id's) that this node would
   need to know

   - For each peer that this node would initiate to, which PPK that we
   would use

   - That the use of PPK is currently optional

   With this configuration, the node will continue to operate with nodes
   that have not yet been upgraded. This is due to the PPK_SUPPORT
   notify; if the initiator has not been upgraded, it will not send the
   PPK_SUPPORT notify (and so the responder will know that we will not
   use a PPK); if the responder has not been upgraded, it will not send
   the PPK_SUPPORT notify (and so the initiator will know not to use a
   PPK). And, if both peers have been upgraded, 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).

7. 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.

Although this protocol preserves all the security properties of IKE
against adversaries with conventional computers, this protocol 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, one suggestion would be to form an initial IKE SA (which
is used to exchange identities), perhaps by using the protocol
documented in RFC6023. Then, you would immediately create a child
IKE SA (which is used to exchange everything else). Because the
child IKE SA keys are a function of SK_d, which is a function of the
PPK (among other things), traffic protected by that SA is secure
against Quantum capable adversaries.

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), here is a list of defined IKEv2 and
IPsec algorithms that should NOT be used, as they are known not to be
Quantum Resistant

Any IKE Encryption algorithm, PRF or Integrity algorithm with key
size <256 bits

Any ESP Transform with key size <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

8. References

8.1. Normative References

   [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-
              Hashing for Message Authentication", RFC 2104,
              DOI 10.17487/RFC2104, February 1997,
              <http://www.rfc-editor.org/info/rfc2104>.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <http://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, <http://www.rfc-editor.org/info/rfc7296>.

8.2. Informational References

   [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,
               <http://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, <http://www.rfc-editor.org/info/rfc6030>.

   [SPDP]   McGrew, D., "A Secure Peer Discovery Protocol (SPDP)",
               2001, <http://www.mindspring.com/~dmcgrew/spdp.txt>.

Appendix A. Discussion and Rationale

   The idea behind this 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 PPK's, 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, and that's 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 translating
   the nonces, 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, and also if we're rolling this 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. Acknowledgement

   We would like to thank Tero Kivine, Valery Smyslov, Paul Wouters 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