Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: May 1, 2017

S. Fluhrer
D. McGrew
P. Kampanakis
Cisco Systems

October 28, 2016

Postquantum Preshared Keys for IKEv2
draft-fluhrer-qr-ikev2-03

Abstract

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

Copyright Notice

Table of Contents

1. Introduction

It is an open question whether or not it is feasible to build a quantum computer, 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 preshared keys does not share this vulnerability, because those keys are one of the inputs to the key derivation function. If the preshared key have sufficient entropy and the PRF and 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 when generating the key material (KEYMAT) keys for the child SAs (along with the parameters that IKEv2 normally uses); this secret provides quantum resistance to the IPsec SAs.

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-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 (both the initiator and the responder) has an optional Postquantum Preshared Key (PPK) (potentially on a per-peer basis, selected by peer identity), and also has a configurable flag that determines whether this postquantum preshared key is mandatory. This preshared key is independent of the preshared key (if any that the IKEv2 protocol uses to perform authentication.

3. Exchanges

If the initiator has a configured postquantum preshared key (whether or not it is optional), then it will include a notify payload in its initial encrypted exchange as follows:

```
Initiator                       Responder
------------------------------------------------------------------
HDR, SK {IDi, [CERT,] [CERTREQ,]
    [IDr,] AUTH, SAi2,
    TS, TSr, N(PPK_NOTIFY)}  --->
```

N(PPK_NOTIFY) 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.

When the responder receives the initial encrypted exchange, it checks to see if it received a notify within that exchange, is configured to support PPK with the initiator's identity, and whether that use is mandatory. If the notify was received, and the responder does have a PPK for that identity, then it responds with the standard IKE response with the PPK_NOTIFY notify message included, namely:

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

If the responder is not configured to support PPK with that identity, it continues with the standard IKE protocol, not including the notification.

If the responder is configured to support PPK with that identity, and it does not receive the notification, then if the PPK usage is configured as mandatory, it MUST abort the exchange. If the PPK usage is configured as optional, it continues with the standard IKE protocol, not including the notification.

This table summarizes the above logic by the responder

Received Nonce	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_NOTIFY Nonce

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; 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).

The protocol continues as standard until it comes time to compute the child SA keying material.

4. Creating Child SA Keying Material

When it comes time to generate the keying material for a child SA, the implementation (both the initiator and the responder) checks to see if they agreed to use a PPK. If they did, then they look up (based on the peer's identity) the configured PPK, and then both sides use one of these alternative formula (based on whether an optional Diffie-Hellman was included):

```
 Ni&apos; = prf(PPK, Ni)
 Nr&apos; = prf(PPK, Nr)
 KEYMAT = prf+(SK_d, Ni&apos; | Nr&apos;)
```

or

```
 Ni&apos; = prf(PPK, Ni)
 Nr&apos; = prf(PPK, Nr)
 KEYMAT = prf+(SK_d, g^ir (new) | Ni&apos; | Nr&apos;)
```

where PPK is the configured postquantum preshared key, Ni, Nr are the nonces from the IKE_SA_INIT exchange if this require is the first Child SA created or the fresh Ni and Nr from the CREATE_CHILD_SA exchange if this is a subsequent creation, and prf is the pseudorandom function that was negotiated for this SA.

This is the standard IKE KEYMAT generation, except that the nonces are transformed (via the negotiated PRF function) using the preshared PPK value

We use this negotiated PRF, rather than negotiating a separate one, because this PRF is agreed by both sides to have sufficient security properties (otherwise, they would have negotiated something else), and so that we don't need to specify a separate negotiation procedure.

When you rekey an IKE SA (generating a fresh SKEYSEED), the initiator and the responder will transform the nonces using the same PPK as they used during the original IKE SA negotiation. That is, they will use the alternate derivation:

```
 Ni&apos; = prf(PPK, Ni)
 Nr&apos; = prf(PPK, Nr)
 SKEYSEED = prf( SK_d (old), g^ir (new) | Ni&apos; | Nr&apos; )
 (SK_d | SK_ai | SK_ar | SK_ei | SK_er | SK_pi | SK_pr) =
       prf+(SKEYSEED, Ni&apos; | Nr&apos; | SPIi | SPIr)
```

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

5. 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 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

6. References

6.1. 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>.

6.2. 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>.

[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 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 values for the initial IKE exchange) unless they can find the PPK, and that's too difficult if the PPK has enough entropy (say, 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.

The third and fourth goals are in partial conflict. In order to achieve postquantum security, we need to stir in the PPK when the keys are computed, however the keys are computed before we know who we're talking to (and so which PPK we should use). And, we can't just tell the other side which PPK to use, as we might use different PPK's for different peers, and so that would violate the anonymity goal. If we just (for example) included a hash of the PPK, someone listening in could easily tell when we're using the same PPK for different exchanges, and thus deduce that the systems are related. The compromise we selected was to stir in the PPK in all the derived keys except the initial IKE SA keys, While this allows an attacker with a Quantum Computer to recover the identities, a poll on the IPsecME mailing list indicated that the majority of the people on the list did not think anonymity was an important property within IKE. We stir in the shared secret within the Child SA keying material; this allows an implementation that wants to protect the other IKE-based traffic to create an initial IKE SA to exchange identities, and then immediately create a Child SA, and use that Child SA to exchange the rest of the negotiation.

In addition, when we stir in the PPK, we always use it to modify a nonce (using the negotiated PRF). We modify the nonce (rather than, say, including the PPK in with the prf or prf+ computation directly) so that this would be easier to implement on an hardware-based IKE implementation; the prf computations might be built-in, but the nonces would be external inputs, and so modifying those would minimize the changes.

Appendix B. Acknowledgement

The idea of stirring in the PPK into the IPsec key generation process was originally suggested on the list by Tero Kivinen.

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