Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: July 4, 2016

S. Fluhrer
D. McGrew
P. Kampanakis
Cisco Systems
January 2016

Postquantum Preshared Keys for IKEv2
draft-fluhrer-qr-ikev2-01

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

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 July 4, 2016.

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 IKE keys (along with the parameters that IKEv2 normally uses); this secret adds quantum resistance to the exchange.

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-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), 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.

In addition, we assume that the initiator knows which PPK to use with the peer it is initiating to (for instance, if it knows the peer, then it can determine which PPK will be used).

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 exchange as follows:

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

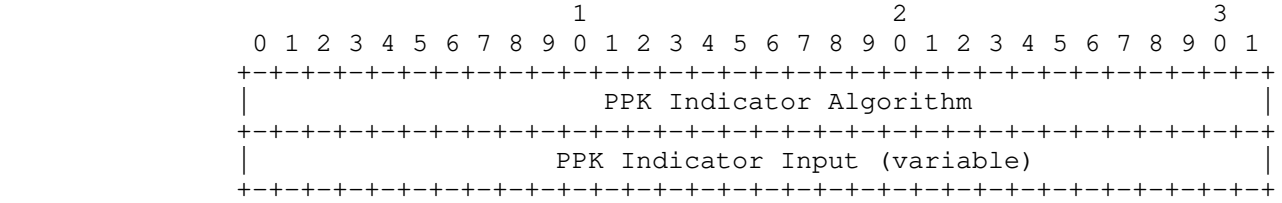
N(PPK_REQUEST) is a status notification payload with the type [TBA]; it has a protocol ID of 0, and no SPI and no notification data associated with it.

When the responder recieves the initial exchange with the notify payload, then (if it is configured to support PPK), it responds with:

```
Initiator                          Responder
-------------------------------------------------------------------
                            <--- HDR, N(COOKIE), N(PPK_ENCODE)
```

If it is not configured to support PPK, the responder continues with the standard IKEv2 protocol.

In other words, it asks for the responder to generate and send a cookie in its responses (as listed in section 2.6 of RFC7296), and in addition, include a notify that gives details of how the initiator should indicate what the PPK is. This notification payload has the type [TBA}; it has a protocol ID of 0, and no SPI; the notification data is of the format:



The PPK Indicator Algorithm is a 4 byte word that states which PPK indicator to use. That is, it gives the encoding format for the PPK that should be used is given to the responder. At present, the only assigned encoding is 0x00000001, which indicates that AES256_SHA256 will be used (as explained below).

PPK Indicator Input is a data input to the PPK indicator Algorithm; its length will depend on the PPK indicator; for the indicator AES256_SHA256, this PPK Indicator Input is 16 bytes.

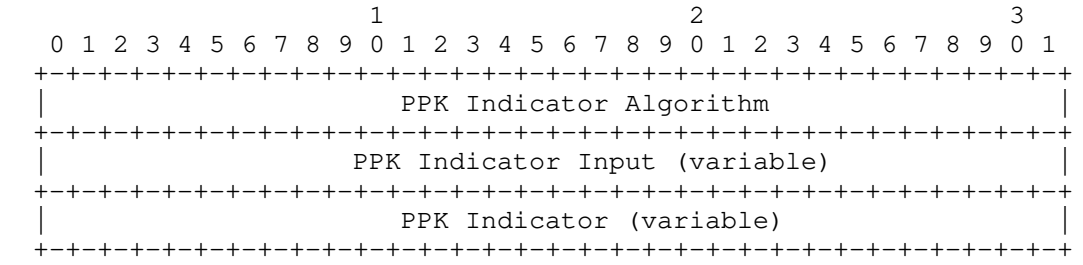
The contents of this PPK Indicator Input is selected by responder policy; below we give trade-offs of the various possibilities

When the initiator receives this notification, it responds as follows:

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

This is the standard IKEv2 cookie response, with a PPK_REQUEST notification added

N(PPK_REQUEST) is a status notification payload with the type [TBA]; it has a protocol ID of 0, and no SPI; however this time, the notification data as as follows:



The PPK Indicator Algorithm and PPK Indicator Input are precisely the same as was given in the PPK_ENCODE format (as is repeated in case the responder ran this cookie protocol in a stateless manner). The PPK Indicator is the encoded version of the PPK that the initiator has. The idea behind this is to allow the responder to select which PPK it should use when it derives the IKEv2 keys.

For the AES256_SHA256 PPK indicator, the PPK Indicator is 16 bytes. To compute it, we use HMAC_SHA256(PPK, "A") as the 256 bit AES key to encrypt the 16 bytes on PPK Indicator Input (in ECB mode), where "A" is a string consisting of a single 0x41 octet.

When the responder receives this notification payload, it verifies that the PPK Indicator Algorithm is as it has specified, and it MAY verify that the PPK Indicator Input is as it has specified. If everything is on the level, it scans through its list of configured postquantum preshared keys, and determines which one it is (possibly (assuming AES256_SHA256_PPK) by computing AES256(HMAC_SHA256(PPK, "A"), PPK_Indicator_Input) and comparing that value to the 16 bytes within the payload. Alternatively, it may have preselected a PPK Indicator Input, and has precomputed (again assuming

AES256_SHA256_PPK) AES256(HMAC_SHA256(PPK, "A"), PPK_Indicator_Input) for each PPK it knows about (in which case, this is a simple search).

If the responder finds a value that matches the payload for a particular PPK, that indicates that the intiator and responder share a PPK and can make use of this extension. Upon finding such a preshared key, the responder includes a notification payload with the response:

```
Initiator                        Responder
-------------------------------------------------------------------
                    <--- HDR, SAr1, Ker, Nr, [CERTREQ], N(PPK_ACK)
```

N(PPK_ACK) 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. This notification serves as a postquantum preshared key confirmation.

If the responder does not find such a PPK, then it MAY continue with the protocol without including a notification ID (if it is configured to not have mandatory preshared keys), or it MAY abort the exchange (if it configured to make preshared keys mandatory).

When the initiator receives the response, it MUST check for the presence of the notification. If it receives one, it marks the SA as using the configured preshared key; if it does not receive one, it MAY either abort the exchange (if the preshared key was configured as mandatory), or it MAY continue without using the preshared key (if the preshared key was configured as optional).

3.1. Computing SKEYSEED

When it comes time to generate the keying material during the initial Exchange, the implementation (both the initiator and the responder) checks to see if there was an agreed-upon preshared key. If there was, then both sides use this alternative formula:

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

where PPK is the postquantum preshared key, Ni, Nr are the nonces exchanged in the IKEv2 exchange, and prf is the pseudorandom function that was negotiated for this SA.

We reuse the negotiated PRF to transform the received nonces. We use this 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.

3.2. Verifying preshared key

Once both the initiator and the responder have exchanged identities, they both double-check with their policy database to verify that they were configured to use those preshared keys when negotiating with the peer. If they are not, they MUST abort the exchange.

3.3. Child SAs

When you create a child SA, 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 one of the alternative derivations (depending on whether an optional Diffie-Hellman was included):

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

or

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

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:

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

4. 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.

In addition, the policy SHOULD be set to negotiate only quantum-resistant symmetric algorithms (AES-256, SHA-256 or better).

The PPK Indicator Input within the PPK_ENCODE notification are there to prevent anyone from deducing whether two different exchanges use the same PPK values. To prevent such a leakage, servers are encouraged to vary them as much as possible (however, they may want to repeat values to speed up the search for the PPK). Repeating these values places the anonymity at risk; however it has no other security implication.

5. References

5.1. Normative References

[AES] National Institute of Technology, "Specification for the Advanced Encryption Standard (AES)", 2001, <FIPS 197>.

[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>.

5.2. Informational References

[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 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 unless they can find the PPK, and that's too difficult if the PPK has enough entropy (say, 256 bits).

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. One such goal is anonymity; that someone listening into the exchanges cannot easily determine who is negotiating with whom.

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 allow the responder to make the trade-off between anonymity and efficiency (by including the PPK Indicator Input, which varies how the PPK is encoded, and allowing the responder to specify it).

A responder who values anonymitity may select a random PPK Indicator Input each time; in this case, the responder needs to do a linear scan over all PPK's it has been configured with

A responder who can't afford a linear scan could precompute a small (possibly rolling) set of the PPK Indicator Inputs; in this case, it would precompute how each PPK would be indicated. If it reissues the same PPK Indicator Input to two different exchanges, someone would be able to verify whether the same PPK was used; this is some loss of anonymity; but is considerably more efficient.

An alternative approach to solve this problem would be to do a normal (non-QR) IKEv2 exchange, and when the two sides obtain identities, see if they need to be QR, and if so, create an immediate IKEv2 child SA (using the PPK). One issue with this is that someone with a quantum computer could deduce the identities used.

A slightly different approach to try to make this even more friendly to IKEv2-based cryptographic hardware might be to use invertible cryptography when we present the nonces to the kdf. The idea here is in case we have IKEv2 hardware that insists on selecting its own nonces (and so we won't be able to give a difference nonce to the KDF); instead, we encrypt the nonce that we send (and decrypt the nonce that we get). Of course, this means that the responder will need to figure out which PPK we're using up front (based on the notifications); we're not sure if this idea would be a net improvement (especially since the transform we're proposing now is cryptographically secure and simple).

The reasoning behind the cryptography used: the values we use in the AES256_SHA256 PPK Indicator Algorithm are cryptographically independent of the values used during the SKEYSEED generation (because, even if we use HMAC_256 as our PRF, HMAC_SHA256(PPK, A) is independent of HMAC_SHA256(PPK, B) if A and B are different strings (and as any real nonce must be longer than a single byte, there is never a collision between that and "A". This independent stems from the assumption that HMAC_SHA256 is a secure MAC.

The method of encoding the PPK within the notification (using AES-256) was chosen as it met two goals:

- Anonymity; given A, AES256_K1(A), B, AES256_K2(B), it's fairly obvious that gives someone (even if they have a quantum computer) no clue about whether K1==K2 (unless either A==B or AES256_K1(A)== AES256_K2(B); both highly unlikely events if A and B are chosen randomly).

- Performance during the linear search; a responder could preexpand the AES keys, and so comparing a potential PPK against a notification from the initiator would amount to performing a single AES block encryption and then doing a 16 byte comparison.

The first goal is considered important; one of the goals of IKEv2 is to provide anonymity. The second is considered important because the linear scan directly affects scalability. While this draft allows the server to gain performance at the cost of anonymity, it was considered useful if we make the fully-anonymous method as attractive as possible. This use of AES makes this linear scan as cheap as possible (while preserving security).

We allow the responder to specify the PPK Indicator Algorithm; this was in response to requests for algorithm agility. At present, it appears unlikely that there would be a need for an additional encoding (as the current one is extremely conservative cryptographically); however the option is there.

The current draft forces a cookie exchange, and hence adds a round trip over the normal IKEv2 operation. This was done to allow the server to specify the PPK Indicator algorithm. While as additional round trip may seem costly, it does not invalidate this proposal, The reason for this proposal is to give an alternative to IKEv1 with preshared keys. While this additional round trip may seem costly, it is important to note that, even with the additional round trip, this proposal is still cheaper than IKEv1. Thus the mechanisms specified in this note meet the goal of providing a better alternative than relying on an obsolete version of the protocol for post quantum security.

One issue that is currently open: what should happen if the initiator guesses at the PPK Indicator Algorithm, selects a random PPK Indicator Input, and includes that in the initial message? After all, if the server follows the recommendation that the cookie exchange is stateless, and if the server chooses the PPK Indicator Input In randomly, it has no way to know that the client isn't running this protocol as specified. If the responder supports that PPK Indicator Algorithm, it could very well respond without forcing a cookie exchange (which would eliminate a message exchange round). It's not clear is whether we should endorse this mode of operation, and explicitly state that if the server recieves such an initial request, and it doesn't recognize the PPK Indicator Input, it should act like it recieved an iniital PPK_REQUEST.

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

IPSecME Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2017

Y. Nir
Check Point
V. Smyslov
ELVIS-PLUS

July 1, 2016

Protecting Internet Key Exchange Protocol version 2 (IKEv2) Implementations from Distributed Denial of Service Attacks
draft-ietf-ipsecme-ddos-protection-07

Abstract

This document recommends implementation and configuration best practices for Internet Key Exchange Protocol version 2 (IKEv2) Responders, to allow them to resist Denial of Service and Distributed Denial of Service attacks. Additionally, the document introduces a new mechanism called "Client Puzzles" that help accomplish this task.

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 January 2, 2017.

Copyright Notice

Table of Contents

1. Introduction

Denial of Service (DoS) attacks have always been considered a serious threat. These attacks are usually difficult to defend against since the amount of resources the victim has is always bounded (regardless

of how high it is) and because some resources are required for distinguishing a legitimate session from an attack.

The Internet Key Exchange protocol version 2 (IKEv2) described in [RFC7296] includes defense against DoS attacks. In particular, there is a cookie mechanism that allows the IKE Responder to defend itself against DoS attacks from spoofed IP-addresses. However, bot-nets have become widespread, allowing attackers to perform Distributed Denial of Service (DDoS) attacks, which are more difficult to defend against. This document presents recommendations to help the Responder counter (D)DoS attacks. It also introduces a new mechanism -- "puzzles" -- that can help accomplish this task.

2. Conventions Used in This Document

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 [RFC2119].

3. The Vulnerability

The IKE_SA_INIT Exchange described in Section 1.2 of [RFC7296] involves the Initiator sending a single message. The Responder replies with a single message and also allocates memory for a structure called a half-open IKE Security Association (SA). This half-open SA is later authenticated in the IKE_AUTH Exchange. If that IKE_AUTH request never comes, the half-open SA is kept for an unspecified amount of time. Depending on the algorithms used and implementation, such a half-open SA will use from around 100 bytes to several thousands bytes of memory.

This creates an easy attack vector against an IKE Responder. Generating the IKE_SA_INIT request is cheap. Sending large amounts of IKE_SA_INIT requests can cause a Responder to use up all its resources. If the Responder tries to defend against this by throttling new requests, this will also prevent legitimate Initiators from setting up IKE SAs.

An obvious defense, which is described in Section 4.2, is limiting the number of half-open SAs opened by a single peer. However, since all that is required is a single packet, an attacker can use multiple spoofed source IP addresses.

If we break down what a Responder has to do during an initial exchange, there are three stages:

1. When the IKE_SA_INIT request arrives, the Responder:

   * Generates or re-uses a Diffie-Hellman (D-H) private part.

   * Generates a Responder Security Parameter Index (SPI).

   * Stores the private part and peer public part in a half-open SA database.

2. When the IKE_AUTH request arrives, the Responder:

   * Derives the keys from the half-open SA.

   * Decrypts the request.

3. If the IKE_AUTH request decrypts properly:

   * Validates the certificate chain (if present) in the IKE_AUTH request.

The fourth stage where the Responder creates the Child SA is not reached by attackers who cannot pass the authentication step.

Stage #1 is pretty light on CPU power, but requires some storage, and it's very light for the Initiator as well. Stage #2 includes private-key operations, so it is much heavier CPU-wise. Stage #3 may include public key operations if certificates are involved. These operations are often more computationly expensive than those performed at stage #2.

To attack such a Responder, an attacker can attempt either to exhaust memory or to exhaust CPU. Without any protection, the most efficient attack is to send multiple IKE_SA_INIT requests and exhaust memory. This is easy because IKE_SA_INIT requests are cheap.

There are obvious ways for the Responder to protect itself without changes to the protocol. It can reduce the time that an entry remains in the half-open SA database, and it can limit the amount of concurrent half-open SAs from a particular address or prefix. The attacker can overcome this by using spoofed source addresses.

The stateless cookie mechanism from Section 2.6 of [RFC7296] prevents an attack with spoofed source addresses. This doesn't completely solve the issue, but it makes the limiting of half-open SAs by address or prefix work. Puzzles, introduced in Section 4.4, accomplish the same thing only more of it. They make it harder for an attacker to reach the goal of getting a half-open SA. Puzzles do not have to be so hard that an attacker cannot afford to solve a single puzzle; it is enough that puzzles increase the cost of

creating a half-open SAs, so the attacker is limited in the amount they can create.

Reducing the lifetime of an abandoned half-open SA also reduces the impact of such attacks. For example, if a half-open SA is kept for 1 minute and the capacity is 60 thousand half-open SAs, an attacker would need to create one thousand half-open SAs per second. If the retention time is reduced to 3 seconds, the attacker would need to create 20 thousand half-open SAs per second to get the same result. By introducing a puzzle, each half-open SA becomes more expensive for an attacker, making it more likely to prevent an exhaustion attack against Responder memory.

At this point, filling up the half-open SA database is no longer the most efficient DoS attack. The attacker has two alternative attacks to do better:

1. Go back to spoofed addresses and try to overwhelm the CPU that deals with generating cookies, or

2. Take the attack to the next level by also sending an IKE_AUTH request.

If an attacker is so powerfull that it is able to overwhelm the Responder's CPU that deals with generating cookies, then the attack cannot be dealt with at the IKE level and must be handled by means of the Intrusion Prevention System (IPS) technology.

On the other hand, the second alternative of sending an IKE_AUTH request is very cheap. It requires generating a proper IKE header with the correct IKE SPIs and a single Encrypted payload. The content of the payload is irrelevant and might be junk. The Responder has to perform the relatively expensive key derivation, only to find that the MAC on the Encrypted payload on the IKE_AUTH request fails the integrity check. If a Responder does not hold on to the calculated SKEYSEED and SK_* keys (which it should in case a valid IKE_AUTH comes in later) this attack might be repeated on the same half-open SA. Puzzles make attacks of such sort more costly for an attacker. See Section 7.2 for details.

Here too, the number of half-open SAs that the attacker can achieve is crucial, because each one allows the attacker to waste some CPU time. So making it hard to make many half-open SAs is important.

A strategy against DDoS has to rely on at least 4 components:

1. Hardening the half-open SA database by reducing retention time.

2. Hardening the half-open SA database by rate-limiting single IPs/ prefixes.

3. Guidance on what to do when an IKE_AUTH request fails to decrypt.

4. Increasing the cost of half-open SAs up to what is tolerable for legitimate clients.

Puzzles are used as a solution for strategy #4.

4. Defense Measures while the IKE SA is being created

4.1. Retention Periods for Half-Open SAs

As a UDP-based protocol, IKEv2 has to deal with packet loss through retransmissions. Section 2.4 of [RFC7296] recommends "that messages be retransmitted at least a dozen times over a period of at least several minutes before giving up". Many retransmission policies in practice wait one or two seconds before retransmitting for the first time.

Because of this, setting the timeout on a half-open SA too low will cause it to expire whenever even one IKE_AUTH request packet is lost. When not under attack, the half-open SA timeout SHOULD be set high enough that the Initiator will have enough time to send multiple retransmissions, minimizing the chance of transient network congestion causing an IKE failure.

When the system is under attack, as measured by the amount of half-open SAs, it makes sense to reduce this lifetime. The Responder should still allow enough time for the round-trip, enough time for the Initiator to derive the D-H shared value, and enough time to derive the IKE SA keys and the create the IKE_AUTH request. Two seconds is probably as low a value as can realistically be used.

It could make sense to assign a shorter value to half-open SAs originating from IP addresses or prefixes that are considered suspect because of multiple concurrent half-open SAs.

4.2. Rate Limiting

Even with DDoS, the attacker has only a limited amount of nodes participating in the attack. By limiting the amount of half-open SAs that are allowed to exist concurrently with each such node, the total amount of half-open SAs is capped, as is the total amount of key derivations that the Responder is forced to complete.

In IPv4 it makes sense to limit the number of half-open SAs based on IP address. Most IPv4 nodes are either directly attached to the Internet using a routable address or are hidden behind a NAT device with a single IPv4 external address. For IPv6, ISPs assign between a /48 and a /64, so it does not make sense for rate-limiting to work on single IPv6 IPs. Instead, ratelimits should be done based on either the /48 or /64 of the misbehaving IPv6 address observed.

The number of half-open SAs is easy to measure, but it is also worthwhile to measure the number of failed IKE_AUTH exchanges. If possible, both factors should be taken into account when deciding which IP address or prefix is considered suspicious.

There are two ways to rate-limit a peer address or prefix:

1. Hard Limit - where the number of half-open SAs is capped, and any further IKE_SA_INIT requests are rejected.

2. Soft Limit - where if a set number of half-open SAs exist for a particular address or prefix, any IKE_SA_INIT request will be required to solve a puzzle.

The advantage of the hard limit method is that it provides a hard cap on the amount of half-open SAs that the attacker is able to create. The disadvantage is that it allows the attacker to block IKE initiation from small parts of the Internet. For example, if an network service provider or some establishment offers Internet connectivity to its customers or employees through an IPv4 NAT device, a single malicious customer can create enough half-open SAs to fill the quota for the NAT device external IP address. Legitimate Initiators on the same network will not be able to initiate IKE.

The advantage of a soft limit is that legitimate clients can always connect. The disadvantage is that an adversary with sufficient CPU resources can still effectively DoS the Responder.

Regardless of the type of rate-limiting used, legitimate initiators that are not on the same network segments as the attackers will not be affected. This is very important as it reduces the adverse impact caused by the measures used to counteract the attack, and allows most initiators to keep working even if they do not support puzzles.

4.3. The Stateless Cookie

Section 2.6 of [RFC7296] offers a mechanism to mitigate DoS attacks: the stateless cookie. When the server is under load, the Responder responds to the IKE_SA_INIT request with a calculated "stateless cookie" - a value that can be re-calculated based on values in the

IKE_SA_INIT request without storing Responder-side state. The Initiator is expected to repeat the IKE_SA_INIT request, this time including the stateless cookie. This mechanism prevents DoS attacks from spoofed IP addresses, since an attacker needs to have a routable IP address to return the cookie.

Attackers that have multiple source IP addresses with return routability, such as in the case of bot-nets, can fill up a half-open SA table anyway. The cookie mechanism limits the amount of allocated state to the number of attackers, multiplied by the number of half-open SAs allowed per peer address, multiplied by the amount of state allocated for each half-open SA. With typical values this can easily reach hundreds of megabytes.

4.4. Puzzles

The puzzle introduced here extends the cookie mechanism of [RFC7296]. It is loosely based on the proof-of-work technique used in Bitcoins [bitcoins]. Puzzles set an upper bound, determined by the attacker's CPU, to the number of negotiations the attacker can initiate in a unit of time.

A puzzle is sent to the Initiator in two cases:

- The Responder is so overloaded that no half-open SAs may be created without solving a puzzle, or
- The Responder is not too loaded, but the rate-limiting method described in Section 4.2 prevents half-open SAs from being created with this particular peer address or prefix without first solving a puzzle.

When the Responder decides to send the challenge to solve a puzzle in response to a IKE_SA_INIT request, the message includes at least three components:

1. Cookie - this is calculated the same as in [RFC7296], i.e. the process of generating the cookie is not specified.
2. Algorithm, this is the identifier of a Pseudo-Random Function (PRF) algorithm, one of those proposed by the Initiator in the SA payload.
3. Zero Bit Count (ZBC). This is a number between 8 and 255 (or a special value - 0, see Section 7.1.1.1) that represents the length of the zero-bit run at the end of the output of the PRF function calculated over the cookie that the Initiator is to send. The values 1-8 are explicitly excluded, because they

create a puzzle that is too easy to solve. Since the mechanism is supposed to be stateless for the Responder, either the same ZBC is used for all Initiators, or the ZBC is somehow encoded in the cookie. If it is global then it means that this value is the same for all the Initiators who are receiving puzzles at any given point of time. The Responder, however, may change this value over time depending on its load.

Upon receiving this challenge, the Initiator attempts to calculate the PRF output using different keys. When enough keys are found such that the resulting PRF output calculated using each of them has a sufficient number of trailing zero bits, that result is sent to the Responder.

The reason for using several keys in the results, rather than just one key, is to reduce the variance in the time it takes the initiator to solve the puzzle. We have chosen the number of keys to be four (4) as a compromise between the conflicting goals of reducing variance and reducing the work the Responder needs to perform to verify the puzzle solution.

When receiving a request with a solved puzzle, the Responder verifies two things:

- That the cookie is indeed valid.
- That the results of PRF of the transmitted cookie calculated with the transmitted keys has a sufficient number of trailing zero bits.

Example 1: Suppose the calculated cookie is 739ae7492d8a810cf5e8dc0f9626c9dda773c5a3 (20 octets), the algorithm is PRF-HMAC-SHA256, and the required number of zero bits is 18. After successively trying a bunch of keys, the Initiator finds the following four 3-octet keys that work:

Key	Last 32 Hex PRF Digits	# 0-bits
061840	e4f957b859d7fb1343b7b94a816c0000	18
073324	0d4233d6278c96e3369227a075800000	23
0c8a2a	952a35d39d5ba06709da43af40700000	20
0d94c8	5a0452b21571e401a3d00803679c0000	18

Table 1: Three solutions for 18-bit puzzle

Example 2: Same cookie, but modify the required number of zero bits to 22. The first 4-octet keys that work to satisfy that requirement are 005d9e57, 010d8959, 0110778d, and 01187e37. Finding these requires 18,382,392 invocations of the PRF.

# 0-bits	Time to Find 4 keys (seconds)
8	0.0025
10	0.0078
12	0.0530
14	0.2521
16	0.8504
17	1.5938
18	3.3842
19	3.8592
20	10.8876

Table 2: The time needed to solve a puzzle of various difficulty for the cookie = 739ae7492d8a810cf5e8dc0f9626c9dda773c5a3

The figures above were obtained on a 2.4 GHz single core i5. Run times can be halved or quartered with multi-core code, but would be longer on mobile phone processors, even if those are multi-core as well. With these figures 18 bits is believed to be a reasonable choice for puzzle level difficulty for all Initiators, and 20 bits is acceptable for specific hosts/prefixes.

Using puzzles mechanism in the IKE_SA_INIT exchange is described in Section 7.1.

4.5. Session Resumption

When the Responder is under attack, it SHOULD prefer previously authenticated peers who present a Session Resumption ticket [RFC5723]. However, the Responder SHOULD NOT serve resumed Initiators exclusively because dropping all IKE_SA_INIT requests would lock out legitimate Initiators that have no resumption ticket. When under attack the Responder SHOULD require Initiators presenting Session Resumption Tickets to pass a return routability check by including the COOKIE notification in the IKE_SESSION_RESUME response message, as described in Section 4.3.2. of [RFC5723]. Note that the Responder SHOULD cache tickets for a short time to reject reused tickets (Section 4.3.1), and therefore there should be no issue of half-open SAs resulting from replayed IKE_SESSION_RESUME messages.

Several kinds of DoS attacks are possible on servers supported IKE Session Resumption. See Section 9.3 of [RFC5723] for details.

4.6. Keeping computed Shared Keys

Once the IKE_SA_INIT exchange is finished, the Responder is waiting for the first message of the IKE_AUTH exchange from the Initiator. At this point the Initiator is not yet authenticated, and this fact allows an attacker to perform an attack, described in Section 3. The attacker can just send garbage in the IKE_AUTH message forcing the Responder to perform costly CPU operations to compute SK_* keys.

If the received IKE_AUTH message failed to decrypt correctly (or failed to pass ICV check), then the Responder SHOULD still keep the computed SK_* keys, so that if it happened to be an attack, then an attacker cannot get advantage of repeating the attack multiple times on a single IKE SA. The responder can also use puzzles in the IKE_AUTH exchange as decribed in Section 7.2.

4.7. Preventing "Hash and URL" Certificate Encoding Attacks

In IKEv2 each side may use the "Hash and URL" Certificate Encoding to instruct the peer to retrieve certificates from the specified location (see Section 3.6 of [RFC7296] for details). Malicious initiators can use this feature to mount a DoS attack on the responder by providing an URL pointing to a large file possibly containing garbage. While downloading the file the responder consumes CPU, memory and network bandwidth.

To prevent this kind of attack, the responder should not blindly download the whole file. Instead, it SHOULD first read the initial few bytes, decode the length of the ASN.1 structure from these bytes, and then download no more than the decoded number of bytes. Note, that it is always possible to determine the length of ASN.1 structures used in IKEv2, if they are DER-encoded, by analyzing the first few bytes. However, since the content of the file being downloaded can be under the attacker’s control, implementations should not blindly trust the decoded length and SHOULD check whether it makes sense before continuing to download the file. Implementations SHOULD also apply a configurable hard limit to the number of pulled bytes and SHOULD provide an ability for an administrator to either completely disable this feature or to limit its use to a configurable list of trusted URLs.

4.8. IKE Fragmentation

IKE Fragmentation described in [RFC7383] allows IKE peers to avoid IP fragmentation of large IKE messages. Attackers can mount several kinds of DoS attacks using IKE Fragmentation. See Section 5 of [RFC7383] for details on how to mitigate these attacks.

5. Defense Measures after an IKE SA is created

Once an IKE SA is created there usually are only a limited amount of IKE messages exchanged. This IKE traffic consists of exchanges aimed to create additional Child SAs, IKE rekeys, IKE deletions and IKE liveness tests. Some of these exchanges require relatively little resources (like liveness check), while others may be resource consuming (like creating or rekeying Child SA with D-H exchange).

Since any endpoint can initiate a new exchange, there is a possibility that a peer would initiate too many exchanges that could exhaust host resources. For example, the peer can perform endless continuous Child SA rekeying or create an overwhelming number of Child SAs with the same Traffic Selectors etc. Such behavior can be caused by broken implementations, misconfiguration, or as an intentional attack. The latter becomes more of a real threat if the peer uses NULL Authentication, as described in [RFC7619]. In this case the peer remains anonymous, allowing it to escape any responsibility for its behaviour. See Section 3 of [RFC7619] for details on how to mitigate attacks when using NULL Authentication.

The following recommendations apply especially for NULL Authenticated IKE sessions, but also apply to authenticated IKE sessions, with the difference that in the latter case, the identified peer can be locked out.

- If the IKEv2 window size is greater than one, peers are able to initiate multiple simultaneous exchanges that increase host resource consumption. Since there is no way in IKEv2 to decrease window size once it has been increased (see Section 2.3 of [RFC7296]), the window size cannot be dynamically adjusted depending on the load. It is NOT RECOMMENDED to allow an IKEv2 window size greater than one when NULL Authentication has been used.

- If a peer initiates an abusive amount of CREATE_CHILD_SA exchanges to rekey IKE SAs or Child SAs, the Responder SHOULD reply with TEMPORARY_FAILURE notifications indicating the peer must slow down their requests.

- If a peer creates many Child SA with the same or overlapping Traffic Selectors, implementations MAY respond with the NO_ADDITIONAL_SAS notification.

- If a peer initiates many exchanges of any kind, the Responder MAY introduce an artificial delay before responding to each request message. This delay would decrease the rate the Responder needs to process requests from any particular peer, and frees up resources on the Responder that can be used for answering legitimate clients. If the Responder receives retransmissions of the request message during the delay period, the retransmitted messages MUST be silently discarded. The delay must be short enough to avoid legitimate peers deleting the IKE SA due to a timeout. It is believed that a few seconds is enough. Note however, that even a few seconds may be too long when settings rely on an immediate response to the request message, e.g. for the purposes of quick detection of a dead peer.

- If these counter-measures are inefficient, implementations MAY delete the IKE SA with an offending peer by sending Delete Payload.

In IKE, a client can request various configuration attributes from server. Most often these attributes include internal IP addresses. Malicious clients can try to exhaust a server's IP address pool by continuously requesting a large number of internal addresses. Server implementations SHOULD limit the number of IP addresses allocated to any particular client. Note, this is not possible with clients using NULL Authentication, since their identity cannot be verified.

6. Plan for Defending a Responder

This section outlines a plan for defending a Responder from a DDoS attack based on the techniques described earlier. The numbers given here are not normative, and their purpose is to illustrate the configurable parameters needed for surviving DDoS attacks.

Implementations are deployed in different environments, so it is RECOMMENDED that the parameters be settable. For example, most commercial products are required to undergo benchmarking where the IKE SA establishment rate is measured. Benchmarking is indistinguishable from a DoS attack and the defenses described in this document may defeat the benchmark by causing exchanges to fail or take a long time to complete. Parameters SHOULD be tunable to allow for benchmarking (if only by turning DDoS protection off).

Since all countermeasures may cause delays and additional work for the Initiators, they SHOULD NOT be deployed unless an attack is

likely to be in progress. To minimize the burden imposed on Initiators, the Responder should monitor incoming IKE requests, for two scenarios:

1. A general DDoS attack. Such an attack is indicated by a high number of concurrent half-open SAs, a high rate of failed IKE_AUTH exchanges, or a combination of both. For example, consider a Responder that has 10,000 distinct peers of which at peak 7,500 concurrently have VPN tunnels. At the start of peak time, 600 peers might establish tunnels within any given minute, and tunnel establishment (both IKE_SA_INIT and IKE_AUTH) takes anywhere from 0.5 to 2 seconds. For this Responder, we expect there to be less than 20 concurrent half-open SAs, so having 100 concurrent half-open SAs can be interpreted as an indication of an attack. Similarly, IKE_AUTH request decryption failures should never happen. Supposing that the tunnels are established using EAP (see Section 2.16 of [RFC7296]), users may be expected to enter a wrong password about 20% of the time. So we'd expect 125 wrong password failures a minute. If we get IKE_AUTH decryption failures from multiple sources more than once per second, or EAP failures more than 300 times per minute, this can also be an indication of a DDoS attack.

2. An attack from a particular IP address or prefix. Such an attack is indicated by an inordinate amount of half-open SAs from a specific IP address or prefix, or an inordinate amount of IKE_AUTH failures. A DDoS attack may be viewed as multiple such attacks. If these are mitigated successfully, there will not be a need to enact countermeasures on all Initiators. For example, measures might be 5 concurrent half-open SAs, 1 decrypt failure, or 10 EAP failures within a minute.

Note that using counter-measures against an attack from a particular IP address may be enough to avoid the overload on the half-open SA database. In this case the number of failed IKE_AUTH exchanges will never exceed the threshold of attack detection.

When there is no general DDoS attack, it is suggested that no cookie or puzzles be used. At this point the only defensive measure is to monitor the number of half-open SAs, and setting a soft limit per peer IP or prefix. The soft limit can be set to 3-5. If the puzzles are used, the puzzle difficulty should be set to such a level (number of zero-bits) that all legitimate clients can handle it without degraded user experience.

As soon as any kind of attack is detected, either a lot of initiations from multiple sources or a lot of initiations from a few sources, it is best to begin by requiring stateless cookies from all

Initiators. This will This will mitigate attacks based on IP address spoofing, and help avoid the need to impose a greater burden in the form of puzzles on the general population of Initiators. This makes the per-node or per-prefix soft limit more effective.

When cookies are activated for all requests and the attacker is still managing to consume too many resources, the Responder MAY start to use puzzles for these requests or increase the difficulty of puzzles imposed on IKE_SA_INIT requests coming from suspicious nodes/ prefixes. This should still be doable by all legitimate peers, but the use of puzzles at a higher difficulty may degrade the user experience, for example by taking up to 10 seconds to solve the puzzle.

If the load on the Responder is still too great, and there are many nodes causing multiple half-open SAs or IKE_AUTH failures, the Responder MAY impose hard limits on those nodes.

If it turns out that the attack is very widespread and the hard caps are not solving the issue, a puzzle MAY be imposed on all Initiators. Note that this is the last step, and the Responder should avoid this if possible.

7. Using Puzzles in the Protocol

This section describes how the puzzle mechanism is used in IKEv2. It is organized as follows. The Section 7.1 describes using puzzles in the IKE_SA_INIT exchange and the Section 7.2 describes using puzzles in the IKE_AUTH exchange. Both sections are divided into subsections describing how puzzles should be presented, solved and processed by the Initiator and the Responder.

7.1. Puzzles in IKE_SA_INIT Exchange

IKE Initiator indicates the desire to create a new IKE SA by sending an IKE_SA_INIT request message. The message may optionally contain a COOKIE notification if this is a repeated request performed after the Responder’s demand to return a cookie.

HDR, [N(COOKIE),] SA, KE, Ni, [V+][N+] -->

According to the plan, described in Section 6, the IKE Responder should monitor incoming requests to detect whether it is under attack. If the Responder learns that a (D)DoS attack is likely to be in progress, then its actions depend on the volume of the attack. If the volume is moderate, then the Responder requests the Initiator to return a cookie. If the volume is so high, that puzzles need to be

used for defense, then the Responder requests the Initiator to solve a puzzle.

The Responder MAY choose to process some fraction of IKE_SA_INIT requests without presenting a puzzle while being under attack to allow legacy clients, that don't support puzzles, to have a chance to be served. The decision whether to process any particular request must be probabilistic, with the probability depending on the Responder's load (i.e. on the volume of attack). The requests that don't contain the COOKIE notification MUST NOT participate in this lottery. In other words, the Responder must first perform a return routability check before allowing any legacy client to be served if it is under attack. See Section 7.1.4 for details.

7.1.1. Presenting a Puzzle

If the Responder makes a decision to use puzzles, then it includes two notifications in its response message - the COOKIE notification and the PUZZLE notification. Note that the PUZZLE notification MUST always be accompanied with the COOKIE notification, since the content of the COOKIE notification is used as an input data when solving puzzle. The format of the PUZZLE notification is described in Section 8.1.

```
                          <--   HDR, N(COOKIE), N(PUZZLE), [V+][N+]
```

The presence of these notifications in an IKE_SA_INIT response message indicates to the Initiator that it should solve the puzzle to have a better chance to be served.

7.1.1.1. Selecting the Puzzle Difficulty Level

The PUZZLE notification contains the difficulty level of the puzzle - the minimum number of trailing zero bits that the result of PRF must contain. In diverse environments it is next to impossible for the Responder to set any specific difficulty level that will result in roughly the same amount of work for all Initiators, because computation power of different Initiators may vary by an order of magnitude, or even more. The Responder may set the difficulty level to 0, meaning that the Initiator is requested to spend as much power to solve a puzzle as it can afford. In this case no specific value of ZBC is required from the Initiator, however the larger the ZBC that Initiator is able to get, the better the chance is that it will be served by the Responder. In diverse environments it is RECOMMENDED that the Initiator set the difficulty level to 0, unless the attack volume is very high.

If the Responder sets a non-zero difficulty level, then the level should be determined by analyzing the volume of the attack. The Responder MAY set different difficulty levels to different requests depending on the IP address the request has come from.

7.1.1.2. Selecting the Puzzle Algorithm

The PUZZLE notification also contains identifier of the algorithm, that must be used by Initiator to compute puzzle.

Cryptographic algorithm agility is considered an important feature for modern protocols ([RFC7696]). Algorithm agility ensures that a protocol doesn't rely on a single built-in set of cryptographic algorithms, but has a means to replace one set with another, and negotiate new algorithms with the peer. IKEv2 fully supports cryptographic algorithm agility for its core operations.

To support crypto agility in case of puzzles, the algorithm that is used to compute a puzzle needs to be negotiated during the IKE_SA_INIT exchange. The negotiation is performed as follows. The initial request message sent by the Initiator contains an SA payload with the list of transforms the Initiator supports and is willing to use in the IKE SA being established. The Responder parses the received SA payload and finds a mutually supported PRFs. The Responder selects the preferred PRF from the list of mutually supported ones and includes it into the PUZZLE notification. There is no requirement that the PRF selected for puzzles be the same as the PRF that is negotiated later for use in core IKE SA crypto operations. If there are no mutually supported PRFs, then IKE SA negotiation will fail anyway and there is no reason to return a puzzle. In this case the Responder returns a NO_PROPOSAL_CHOSEN notification. Note that PRF is a mandatory transform type for IKE SA (see Sections 3.3.2 and 3.3.3 of [RFC7296]) and at least one transform of this type must always be present in the SA payload in an IKE_SA_INIT request message.

7.1.1.3. Generating a Cookie

If the Responder supports puzzles then a cookie should be computed in such a manner that the Responder is able to learn some important information from the sole cookie, when it is later returned back by Initiator. In particular - the Responder should be able to learn the following information:

- Whether the puzzle was given to the Initiator or only the cookie was requested.
- The difficulty level of the puzzle given to the Initiator.

- The number of consecutive puzzles given to the Initiator.
- The amount of time the Initiator spent to solve the puzzles. This can be calculated if the cookie is timestamped.

This information helps the Responder to make a decision whether to serve this request or demand more work from the Initiator.

One possible approach to get this information is to encode it in the cookie. The format of such encoding is an implementation detail of Responder, as the cookie would remain an opaque blob to the Initiator. If this information is encoded in the cookie, then the Responder MUST make it integrity protected, so that any intended or accidental alteration of this information in the returned cookie is detectable. So, the cookie would be generated as:

```
Cookie = <VersionIDofSecret> | <AdditionalInfo> |
                  Hash(Ni | IPi | SPIi | <AdditionalInfo> | <secret>)
```

Note, that according to the Section 2.6 of [RFC7296], the size of the cookie cannot exceed 64 bytes.

Alternatively, the Responder may continue to generate a cookie as suggested in Section 2.6 of [RFC7296], but associate the additional information, using local storage identified with the particular version of the secret. In this case the Responder should have different secrets for every combination of difficulty level and number of consecutive puzzles, and should change the secrets periodically, keeping a few previous versions, to be able to calculate how long ago a cookie was generated.

The Responder may also combine these approaches. This document doesn't mandate how the Responder learns this information from a cookie.

7.1.2. Solving a Puzzle and Returning the Solution

If the Initiator receives a puzzle but it doesn't support puzzles, then it will ignore the PUZZLE notification as an unrecognized status notification (in accordance to Section 3.10.1 of [RFC7296]). The Initiator MAY ignore the PUZZLE notification if it is not willing to spend resources to solve the puzzle of the requested difficulty, even if it supports puzzles. In both cases the Initiator acts as described in Section 2.6 of [RFC7296] - it restarts the request and includes the received COOKIE notification into it. The Responder should be able to distinguish the situation when it just requested a cookie from the situation where the puzzle was given to the Initiator, but the Initiator for some reason ignored it.

If the received message contains a PUZZLE notification and doesn't contain a COOKIE notification, then this message is malformed because it requests to solve the puzzle, but doesn't provide enough information to allow the puzzle to be solved. In this case the Initiator MUST ignore the received message and continue to wait until either a valid PUZZLE notification is received or the retransmission timer fires. If it fails to receive a valid message after several retransmissions of IKE_SA_INIT requests, then it means that something is wrong and the IKE SA cannot be established.

If the Initiator supports puzzles and is ready to solve them, then it tries to solve the given puzzle. After the puzzle is solved the Initiator restarts the request and returns back to the Responder the puzzle solution in a new payload called a Puzzle Solution payload (denoted as PS, see Section 8.2) along with the received COOKIE notification.

HDR, N(COOKIE), [PS,] SA, KE, Ni, [V+][N+] -->

7.1.3. Computing a Puzzle

General principals of constructing puzzles in IKEv2 are described in Section 4.4. They can be summarized as follows: given unpredictable string S and pseudo-random function PRF find N different keys Ki (where i=[1..N]) for that PRF so that the result of PRF(Ki,S) has at least the specified number of trailing zero bits. This specification requires that the solution to the puzzle contain 4 different keys (i.e. N=4).

In the IKE_SA_INIT exchange it is the cookie that plays the role of unpredictable string S. In other words, in the IKE_SA_INIT the task for the IKE Initiator is to find the four different, equal-sized keys Ki for the agreed upon PRF such that each result of PRF(Ki,cookie) where i = [1..4] has a sufficient number of trailing zero bits. Only the content of the COOKIE notification is used in puzzle calculation, i.e. the header of the Notification payload is not included.

Note, that puzzles in the IKE_AUTH exchange are computed differently than in the IKE_SA_INIT_EXCHANGE. See Section 7.2.3 for details.

7.1.4. Analyzing Repeated Request

The received request must at least contain a COOKIE notification. Otherwise it is an initial request and it must be processed according to Section 7.1. First, the cookie MUST be checked for validity. If the cookie is invalid, then the request is treated as initial and is processed according to Section 7.1. It is RECOMMENDED that a new cookie is requested in this case.

If the cookie is valid, then some important information is learned from it, or from local state based on identifier of the cookie's secret (see Section 7.1.1.3 for details). This information helps the Responder to sort out incoming requests, giving more priority to those which were created by spending more of the Initiator's resources.

First, the Responder determines if it requested only a cookie, or presented a puzzle to the Initiator. If no puzzle was given, this means that at the time the Responder requested a cookie it didn't detect the (D)DoS attack or the attack volume was low. In this case the received request message must not contain the PS payload, and this payload MUST be ignored if the message contains a PS payload for any reason. Since no puzzle was given, the Responder marks the request with the lowest priority since the Initiator spent little resources creating it.

If the Responder learns from the cookie that the puzzle was given to the Initiator, then it looks for the PS payload to determine whether its request to solve the puzzle was honored or not. If the incoming message doesn't contain a PS payload, this means that the Initiator either doesn't support puzzles or doesn't want to deal with them. In either case the request is marked with the lowest priority since the Initiator spent little resources creating it.

If a PS payload is found in the message, then the Responder MUST verify the puzzle solution that it contains. The solution is interpreted as four different keys. The result of using each of them in the PRF (as described in Section 7.1.3) must contain at least the requested number of trailing zero bits. The Responder MUST check all of the four returned keys.

If any checked result contains fewer bits than were requested, this means that the Initiator spent less resources than expected by the Responder. This request is marked with the lowest priority.

If the Initiator provided the solution to the puzzle satisfying the requested difficulty level, or if the Responder didn't indicate any particular difficulty level (by setting ZBC to zero) and the Initiator was free to select any difficulty level it can afford, then the priority of the request is calculated based on the following considerations:

- The Responder must take the smallest number of trailing zero bits among the checked results and count it as the number of zero bits the Initiator solved for.

- The higher number of zero bits the Initiator provides, the higher priority its request should receive.

- The more consecutive puzzles the Initiator solved, the higher priority it should receive.

- The more time the Initiator spent solving the puzzles, the higher priority it should receive.

After the priority of the request is determined the final decision whether to serve it or not is made.

7.1.5. Deciding if to Serve the Request

The Responder decides what to do with the request based on the request's priority and the Responder's current load. There are three possible actions:

- Accept request.

- Reject request.

- Demand more work from the Initiator by giving it a new puzzle.

The Responder SHOULD accept an incoming request if its priority is high - this means that the Initiator spent quite a lot of resources. The Responder MAY also accept some low-priority requests where the Initiators don't support puzzles. The percentage of accepted legacy requests depends on the Responder's current load.

If the Initiator solved the puzzle, but didn't spend much resources for it (the selected puzzle difficulty level appeared to be low and the Initiator solved it quickly), then the Responder SHOULD give it another puzzle. The more puzzles the Initiator solves the higher its chances are to be served.

The details of how the Responder makes a decision for any particular request are implementation dependent. The Responder can collect all of the incoming requests for some short period of time, sort them out based on their priority, calculate the number of available memory slots for half-open IKE SAs and then serve that number of requests from the head of the sorted list. The remainder of requests can be either discarded or responded to with new puzzle requests.

Alternatively, the Responder may decide whether to accept every incoming request with some kind of lottery, taking into account its priority and the available resources.

7.2. Puzzles in an IKE_AUTH Exchange

Once the IKE_SA_INIT exchange is completed, the Responder has created a state and is waiting for the first message of the IKE_AUTH exchange from the Initiator. At this point the Initiator has already passed the return routability check and has proved that it has performed some work to complete IKE_SA_INIT exchange. However, the Initiator is not yet authenticated and this allows a malicious Initiator to perform an attack, described in Section 3. Unlike a DoS attack in the IKE_SA_INIT exchange, which is targeted on the Responder's memory resources, the goal of this attack is to exhaust a Responder's CPU power. The attack is performed by sending the first IKE_AUTH message containing garbage. This costs nothing to the Initiator, but the Responder has to perform relatively costly operations when computing the D-H shared secret and deriving SK_* keys to be able to verify authenticity of the message. If the Responder doesn't keep the computed keys after an unsuccessful verification of the IKE_AUTH message, then the attack can be repeated several times on the same IKE SA.

The Responder can use puzzles to make this attack more costly for the Initiator. The idea is that the Responder includes a puzzle in the IKE_SA_INIT response message and the Initiator includes a puzzle solution in the first IKE_AUTH request message outside the Encrypted payload, so that the Responder is able to verify puzzle solution before computing the D-H shared secret. The difficulty level of the puzzle should be selected so that the Initiator would spend substantially more time to solve the puzzle than the Responder to compute the shared secret.

The Responder should constantly monitor the amount of the half-open IKE SA states that receive IKE_AUTH messages that cannot be decrypted due to integrity check failures. If the percentage of such states is high and it takes an essential fraction of Responder's computing power to calculate keys for them, then the Responder may assume that it is under attack and SHOULD use puzzles to make it harder for attackers.

7.2.1. Presenting Puzzle

The Responder requests the Initiator to solve a puzzle by including the PUZZLE notification in the IKE_SA_INIT response message. The Responder MUST NOT use puzzles in the IKE_AUTH exchange unless a puzzle has been previously presented and solved in the preceding IKE_SA_INIT exchange.

```
                          <--   HDR, SA, KE, Nr, N(PUZZLE), [V+][N+]
```

7.2.1.1. Selecting Puzzle Difficulty Level

The difficulty level of the puzzle in the IKE_AUTH exchange should be chosen so that the Initiator would spend more time to solve the puzzle than the Responder to compute the D-H shared secret and the keys needed to decrypt and verify the IKE_AUTH request message. On the other hand, the difficulty level should not be too high, otherwise legitimate clients will experience an additional delay while establishing the IKE SA.

Note, that since puzzles in the IKE_AUTH exchange are only allowed to be used if they were used in the preceding IKE_SA_INIT exchange, the Responder would be able to estimate the computational power of the Initiator and select the difficulty level accordingly. Unlike puzzles in the IKE_SA_INIT, the requested difficulty level for IKE_AUTH puzzles MUST NOT be zero. In other words, the Responder must always set a specific difficulty level and must not let the Initiator to choose it on its own.

7.2.1.2. Selecting the Puzzle Algorithm

The algorithm for the puzzle is selected as described in Section 7.1.1.2. There is no requirement that the algorithm for the puzzle in the IKE_SA INIT exchange be the same as the algorithm for the puzzle in IKE_AUTH exchange; however, it is expected that in most cases they will be the same.

7.2.2. Solving Puzzle and Returning the Solution

If the IKE_SA_INIT response message contains the PUZZLE notification and the Initiator supports puzzles, it MUST solve the puzzle. Note, that puzzle construction in the IKE_AUTH exchange differs from the puzzle construction in the IKE_SA_INIT exchange and is described in Section 7.2.3. Once the puzzle is solved the Initiator sends the IKE_AUTH request message, containing the Puzzle Solution payload.

```
HDR, PS, SK {IDi, [CERT,] [CERTREQ,]
             [IDr,] AUTH, SA, TSi, TSr}   -->
```

The Puzzle Solution (PS) payload MUST be placed outside the Encrypted payload, so that the Responder is able to verify the puzzle before calculating the D-H shared secret and the SK_* keys.

If IKE Fragmentation [RFC7383] is used in IKE_AUTH exchange, then the PS payload MUST be present only in the first IKE Fragment message, in accordance with the Section 2.5.3 of [RFC7383]. Note, that calculation of the puzzle in the IKE_AUTH exchange doesn't depend on the content of the IKE_AUTH message (see Section 7.2.3). Thus the

Initiator has to solve the puzzle only once and the solution is valid for both unfragmented and fragmented IKE messages.

7.2.3. Computing the Puzzle

A puzzle in the IKE_AUTH exchange is computed differently than in the IKE_SA_INIT exchange (see Section 7.1.3). The general principle is the same; the difference is in the construction of the string S. Unlike the IKE_SA_INIT exchange, where S is the cookie, in the IKE_AUTH exchange S is a concatenation of Nr and SPIr. In other words, the task for IKE Initiator is to find the four different keys Ki for the agreed upon PRF such that each result of PRF(Ki,Nr | SPIr) where i=[1..4] has a sufficient number of trailing zero bits. Nr is a nonce used by the Responder in the IKE_SA_INIT exchange, stripped of any headers. SPIr is the IKE Responder's SPI from the IKE header of the SA being established.

7.2.4. Receiving the Puzzle Solution

If the Responder requested the Initiator to solve a puzzle in the IKE_AUTH exchange, then it MUST silently discard all the IKE_AUTH request messages without the Puzzle Solution payload.

Once the message containing a solution to the puzzle is received, the Responder MUST verify the solution before performing computationlly intensive operations i.e. computing the D-H shared secret and the SK_* keys. The Responder MUST verify all four of the returned keys.

The Responder MUST silently discard the received message if any checked verification result is not correct (contains insufficient number of trailing zero bits). If the Responder successfully verifies the puzzle and calculates the SK_* key, but the message authenticity check fails, then it SHOULD save the calculated keys in the IKE SA state while waiting for the retransmissions from the Initiator. In this case the Responder may skip verification of the puzzle solution and ignore the Puzzle Solution payload in the retransmitted messages.

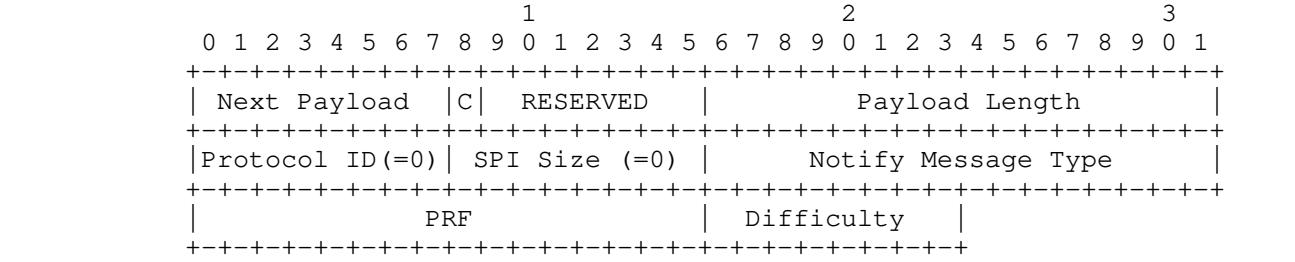
If the Initiator uses IKE Fragmentation, then it is possible, that due to packet loss and/or reordering the Responder could receive non-first IKE Fragment messages before receiving the first one containing the PS payload. In this case the Responder MAY choose to keep the received fragments until the first fragment containing the solution to the puzzle is received. In this case the Responder SHOULD NOT try to verify authenticity of the kept fragments until the first fragment with the PS payload is received and the solution to the puzzle is verified. After successful verification of the puzzle, the Responder

can then calculate the SK_* key and verify authenticity of the collected fragments.

8. Payload Formats

8.1. PUZZLE Notification

The PUZZLE notification is used by the IKE Responder to inform the Initiator about the need to solve the puzzle. It contains the difficulty level of the puzzle and the PRF the Initiator should use.

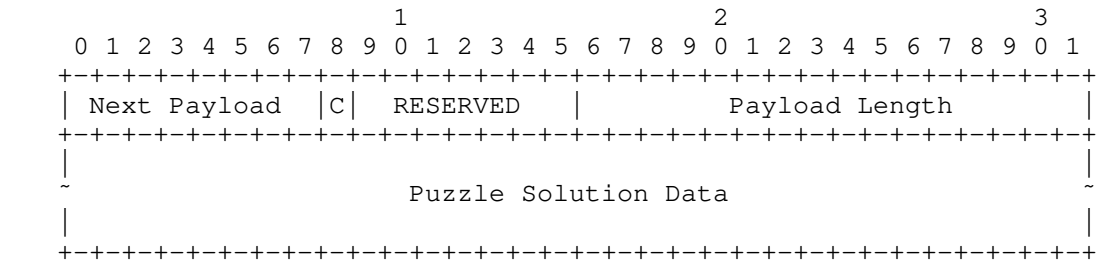


- Protocol ID (1 octet) -- MUST be 0.
- SPI Size (1 octet) - MUST be 0, meaning no Security Parameter Index (SPI) is present.
- Notify Message Type (2 octets) -- MUST be <TBA by IANA>, the value assigned for the PUZZLE notification.
- PRF (2 octets) -- Transform ID of the PRF algorithm that must be used to solve the puzzle. Readers should refer to the section "Transform Type 2 - Pseudo-Random Function Transform IDs" in [IKEV2-IANA] for the list of possible values.
- Difficulty (1 octet) -- Difficulty Level of the puzzle. Specifies the minimum number of trailing zero bits (ZBC), that each of the results of PRF must contain. Value 0 means that the Responder doesn't request any specific difficulty level and the Initiator is free to select an appropriate difficulty level on its own (see Section 7.1.1.1 for details).

This notification contains no data.

8.2. Puzzle Solution Payload

The solution to the puzzle is returned back to the Responder in a dedicated payload, called the Puzzle Solution payload and denoted as PS in this document.



- o Puzzle Solution Data (variable length) -- Contains the solution to the puzzle - four different keys for the selected PRF. This field MUST NOT be empty. All of the keys MUST have the same size, therefore the size of this field is always a mutiple of 4 bytes. If the selected PRF accepts only fixed-size keys, then the size of each key MUST be of that fixed size. If the agreed upon PRF accepts keys of any size, then then the size of each key MUST be between 1 octet and the preferred key length of the PRF (inclusive). It is expected that in most cases the keys will be 4 (or even less) octets in length, however it depends on puzzle difficulty and on the Initiator's strategy to find solutions, and thus the size is not mandated by this specification. The Responder determines the size of each key by dividing the size of the Puzzle Solution Data by 4 (the number of keys). Note that the size of Puzzle Solution Data is the size of Payload (as indicated in Payload Length field) minus 4 - the size of Payload Header.

The payload type for the Puzzle Solution payload is <TBA by IANA>.

9. Operational Considerations

The puzzle difficulty level should be set by balancing the requirement to minimize the latency for legitimate Initiators with making things difficult for attackers. A good rule of thumb is for taking about 1 second to solve the puzzle. A typical Initiator or bot-net member in 2014 can perform slightly less than a million hashes per second per core, so setting the difficulty level to n=20 is a good compromise. It should be noted that mobile Initiators, especially phones are considerably weaker than that. Implementations should allow administrators to set the difficulty level, and/or be able to set the difficulty level dynamically in response to load.

Initiators should set a maximum difficulty level beyond which they won't try to solve the puzzle and log or display a failure message to the administrator or user.

10. Security Considerations

Care must be taken when selecting parameters for the puzzles, in particular the puzzle difficulty. If the puzzles are too easy for the majority of attacker, then the puzzle mechanism wouldn't be able to prevent (D)DoS attacks and would only impose an additional burden on legitimate Initiators. On the other hand, if the puzzles are too hard for the majority of Initiators, then many legitimate users would experience unacceptable delays in IKE SA setup (and unacceptable power consumption on mobile devices), that might cause them to cancel the connection attempt. In this case the resources of the Responder are preserved, however the DoS attack can be considered successful. Thus a sensible balance should be kept by the Responder while choosing the puzzle difficulty - to defend itself and to not over-defend itself. It is RECOMMENDED that the puzzle difficulty be chosen so, that the Responder's load remains close to the maximum it can tolerate. It is also RECOMMENDED to dynamically adjust the puzzle difficulty in accordance to the current Responder's load.

Solving puzzles requires a lot of CPU power that increases power consumption. This additional power consumption can negatively affect battery-powered Initiators, e.g. mobile phones or some IoT devices. If puzzles are too hard, then the required additional power consumption may appear to be unacceptable for some Initiators. The Responder SHOULD take this possibility into consideration while choosing the puzzle difficulty, and while selecting which percentage of Initiators are allowed to reject solving puzzles. See Section 7.1.4 for details.

If the Initiator uses NULL Authentication [RFC7619] then its identity is never verified. This condition may be used by attackers to perform a DoS attack after the IKE SA is established. Responders that allow unauthenticated Initiators to connect must be prepared to deal with various kinds of DoS attacks even after the IKE SA is created. See Section 5 for details.

To prevent amplification attacks implementations must strictly follow the retransmission rules described in Section 2.1 of [RFC7296].

11. IANA Considerations

This document defines a new payload in the "IKEv2 Payload Types" registry:

<TBA>	Puzzle Solution	PS

This document also defines a new Notify Message Type in the "IKEv2 Notify Message Types - Status Types" registry:

<TBA> PUZZLE

12. Acknowledgements

The authors thank Tero Kivinen, Yaron Sheffer, and Scott Fluhrer for their contributions to the design of the protocol. In particular, Tero Kivinen suggested the kind of puzzle where the task is to find a solution with a requested number of zero trailing bits. Yaron Sheffer and Scott Fluhrer suggested a way to make puzzle difficulty less erratic by solving several weaker puzles. The authors also thank David Waltermire and Paul Wouters for their careful reviews of the document, Graham Bartlett for pointing out to the possibility of the "Hash & URL" related attack, and all others who commented the document.

13. References

13.1. 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, <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>.

[RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <http://www.rfc-editor.org/info/rfc7383>.

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

13.2. Informative References

[bitcoins] Nakamoto, S., "Bitcoin: A Peer-to-Peer Electronic Cash System", October 2008, <https://bitcoin.org/bitcoin.pdf>.

[RFC5723] Sheffer, Y. and H. Tschofenig, "Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption", RFC 5723, DOI 10.17487/RFC5723, January 2010, <http://www.rfc-editor.org/info/rfc5723>.

[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, <http://www.rfc-editor.org/info/rfc7619>.

[RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <http://www.rfc-editor.org/info/rfc7696>.

Authors' Addresses

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 6789735
Israel

EMail: ynir.ietf@gmail.com

Valery Smyslov
ELVIS-PLUS
PO Box 81
Moscow (Zelenograd) 124460
Russian Federation

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

Network Working Group
Internet-Draft
Obsoletes: 4307 (if approved)
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: November 14, 2016

Y. Nir
Check Point
T. Kivinen
INSIDE Secure
P. Wouters
Red Hat
D. Migault
Ericsson

May 13, 2016

Algorithm Implementation Requirements and Usage Guidance for IKEv2
draft-ietf-ipsecme-rfc4307bis-09

Abstract

The IPsec series of protocols makes use of various cryptographic algorithms in order to provide security services. The Internet Key Exchange (IKE) protocol is used to negotiate the IPsec Security Association (IPsec SA) parameters, such as which algorithms should be used. To ensure interoperability between different implementations, it is necessary to specify a set of algorithm implementation requirements and usage guidance to ensure that there is at least one algorithm that all implementations support. This document defines the current algorithm implementation requirements and usage guidance for IKEv2. This document does not update the algorithms used for packet encryption using IPsec Encapsulated Security Payload (ESP).

Table of Contents

1. Introduction

The Internet Key Exchange (IKE) protocol [RFC7296] is used to negotiate the parameters of the IPsec SA, such as the encryption and authentication algorithms and the keys for the protected communications between the two endpoints. The IKE protocol itself is also protected by cryptographic algorithms which are negotiated

between the two endpoints using IKE. Different implementations of IKE may negotiate different algorithms based on their individual local policy. To ensure interoperability, a set of "mandatory-to-implement" IKE cryptographic algorithms is defined.

This document describes the parameters of the IKE protocol and updates the IKEv2 specification because it changes the mandatory to implement authentication algorithms of the section 4 of the RFC7296 by saying RSA key lengths of less than 2048 are SHOULD NOT. It does not describe the cryptographic parameters of the AH or ESP protocols.

1.1. Updating Algorithm Implementation Requirements and Usage Guidance

The field of cryptography evolves continuously. New stronger algorithms appear and existing algorithms are found to be less secure then originally thought. Therefore, algorithm implementation requirements and usage guidance need to be updated from time to time to reflect the new reality. The choices for algorithms must be conservative to minimize the risk of algorithm compromise. Algorithms need to be suitable for a wide variety of CPU architectures and device deployments ranging from high end bulk encryption devices to small low-power IoT devices.

The algorithm implementation requirements and usage guidance may need to change over time to adapt to the changing world. For this reason, the selection of mandatory-to-implement algorithms was removed from the main IKEv2 specification and placed in a separate document.

1.2. Updating Algorithm Requirement Levels

The mandatory-to-implement algorithm of tomorrow should already be available in most implementations of IKE by the time it is made mandatory. This document attempts to identify and introduce those algorithms for future mandatory-to-implement status. There is no guarantee that the algorithms in use today may become mandatory in the future. Published algorithms are continuously subjected to cryptographic attack and may become too weak or could become completely broken before this document is updated.

This document only provides recommendations for the mandatory-to-implement algorithms or algorithms too weak that are recommended not to be implemented. As a result, any algorithm listed at the IKEv2 IANA registry not mentioned in this document MAY be implemented. For clarification and consistency with [RFC4307] an algorithm will be denoted here as MAY only when it has been downgraded.

Although this document updates the algorithms to keep the IKEv2 communication secure over time, it also aims at providing

recommendations so that IKEv2 implementations remain interoperable. IKEv2 interoperability is addressed by an incremental introduction or deprecation of algorithms. In addition, this document also considers the new use cases for IKEv2 deployment, such as Internet of Things (IoT).

It is expected that deprecation of an algorithm is performed gradually. This provides time for various implementations to update their implemented algorithms while remaining interoperable. Unless there are strong security reasons, an algorithm is expected to be downgraded from MUST to MUST- or SHOULD, instead of MUST NOT. Similarly, an algorithm that has not been mentioned as mandatory-to-implement is expected to be introduced with a SHOULD instead of a MUST.

The current trend toward Internet of Things and its adoption of IKEv2 requires this specific use case to be taken into account as well. IoT devices are resource constrained devices and their choice of algorithms are motivated by minimizing the footprint of the code, the computation effort and the size of the messages to send. This document indicates "[IoT]" when a specified algorithm is specifically listed for IoT devices. Requirement levels that are marked as "IoT" apply to IoT devices and to server-side implementations that might presumably need to interoperate with them, including any general-purpose VPN gateways.

1.3. Document Audience

The recommendations of this document mostly target IKEv2 implementers as implementations need to meet both high security expectations as well as high interoperability between various vendors and with different versions. Interoperability requires a smooth move to more secure cipher suites. This may differ from a user point of view that may deploy and configure IKEv2 with only the safest cipher suite.

This document does not give any recommendations for the use of algorithms, it only gives implementation recommendations for implementations. The use of algorithms by users is dictated by the security policy requirements for that specific user, and are outside the scope of this document.

IKEv1 is out of scope of this document. IKEv1 is deprecated and the recommendations of this document must not be considered for IKEv1, as most IKEv1 implementations have been "frozen" and will not be able to update the list of mandatory-to-implement algorithms.

2. Conventions Used in This Document

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 [RFC2119].

We define some additional terms here:

SHOULD+	This term means the same as SHOULD. However, it is likely that an algorithm marked as SHOULD+ will be promoted at some future time to be a MUST.
SHOULD-	This term means the same as SHOULD. However, an algorithm marked as SHOULD- may be deprecated to a MAY in a future version of this document.
MUST-	This term means the same as MUST. However, we expect at some point that this algorithm will no longer be a MUST in a future document. Although its status will be determined at a later time, it is reasonable to expect that if a future revision of a document alters the status of a MUST- algorithm, it will remain at least a SHOULD or a SHOULD- level.
IoT	stands for Internet of Things.

3. Algorithm Selection

3.1. Type 1 - IKEv2 Encryption Algorithm Transforms

The algorithms in the below table are negotiated in the SA payload and used for the Encrypted Payload. References to the specification defining these algorithms and the ones in the following subsections are in the IANA registry [IKEV2-IANA]. Some of these algorithms are Authenticated Encryption with Associated Data (AEAD - [RFC5282]). Algorithms that are not AEAD MUST be used in conjunction with an integrity algorithms in Section 3.3.

Name	Status	AEAD?	Comment
ENCR_AES_CBC	MUST-	No	[1]
ENCR_CHACHA20_POLY1305	SHOULD	Yes	
AES-GCM with a 16 octet ICV	SHOULD	Yes	[1]
ENCR_AES_CCM_8	SHOULD	Yes	[1][IoT]
ENCR_3DES	MAY	No	
ENCR_DES	MUST NOT	No	

[1] - This requirement level is for 128-bit keys. 256-bit keys are at SHOULD. 192-bit keys can safely be ignored. [IoT] - This requirement is for interoperability with IoT.

ENCR_AES_CBC is raised from SHOULD+ in [RFC4307] to MUST. It is the only shared mandatory-to-implement algorithm with RFC4307 and as a result it is necessary for interoperability with IKEv2 implementation compatible with RFC4307.

ENCR_CHACHA20_POLY1305 was not ready to be considered at the time of RFC4307. It has been recommended by the CRFG and others as an alternative to AES-CBC and AES-GCM. It is also being standardized for IPsec for the same reasons. At the time of writing, there were not enough IKEv2 implementations supporting ENCR_CHACHA20_POLY1305 to be able to introduce it at the SHOULD+ level.

AES-GCM with a 16 octet ICV was not considered in RFC4307. At the time RFC4307 was written, AES-GCM was not defined in an IETF document. AES-GCM was defined for ESP in [RFC4106] and later for IKEv2 in [RFC5282]. The main motivation for adopting AES-GCM for ESP is encryption performance and key longevity compared to AES-CBC. This resulted in AES-GCM being widely implemented for ESP. As the computation load of IKEv2 is relatively small compared to ESP, many IKEv2 implementations have not implemented AES-GCM. For this reason, AES-GCM is not promoted to a greater status than SHOULD. The reason for promotion from MAY to SHOULD is to promote the slightly more secure AEAD method over the traditional encrypt+auth method. Its status is expected to be raised once widely implemented. As the advantage of the shorter (and weaker) ICVs is minimal, the 8 and 12 octet ICV's remain at the MAY level.

ENCR_AES_CCM_8 was not considered in RFC4307. This document considers it as SHOULD be implemented in order to be able to interact with Internet of Things devices. As this case is not a general use case for non-IoT VPNs, its status is expected to remain as SHOULD. The 8 octet size of the ICV is expected to be sufficient for most use cases of IKEv2, as far less packets are exchanged on those cases, and

IoT devices want to make packets as small as possible. When implemented, ENCR_AES_CCM_8 MUST be implemented for key length 128 and MAY be implemented for key length 256.

ENCR_3DES has been downgraded from RFC4307 MUST- to SHOULD NOT. All IKEv2 implementation already implement ENCR_AES_CBC, so there is no need to keep support for the much slower ENCR_3DES. In addition, ENCR_CHACHA20_POLY1305 provides a more modern alternative to AES.

ENCR_DES can be brute-forced using of-the-shelves hardware. It provides no meaningful security whatsoever and therefor MUST NOT be implemented.

3.2. Type 2 - IKEv2 Pseudo-random Function Transforms

Transform Type 2 algorithms are pseudo-random functions used to generate pseudo-random values when needed.

If an algorithm is selected as the integrity algorithm, it SHOULD also be used as the PRF. When using an AEAD cipher, a choice of PRF needs to be made. The table below lists the recommended algorithms.

Name	Status	Comment
PRF_HMAC_SHA2_256	MUST	
PRF_HMAC_SHA2_512	SHOULD+	
PRF_HMAC_SHA1	MUST-	
PRF_AES128_XCBC	SHOULD	[IoT]
PRF_HMAC_MD5	MUST NOT	

[IoT] - This requirement is for interoperability with IoT

PRF_HMAC_SHA2_256 was not mentioned in RFC4307, as no SHA2 based transforms were mentioned. PRF_HMAC_SHA2_256 MUST be implemented in order to replace SHA1 and PRF_HMAC_SHA1.

PRF_HMAC_SHA2_512 SHOULD be implemented as a future replacement for PRF_HMAC_SHA2_256 or when stronger security is required. PRF_HMAC_SHA2_512 is preferred over PRF_HMAC_SHA2_384, as the additional overhead of PRF_HMAC_SHA2_512 is negligible.

PRF_HMAC_SHA1 has been downgraded from MUST in RFC4307 to MUST- as their is an industry-wide trend to deprecate its usage.

PRF_AES128_XCBC is only recommended in the scope of IoT, as Internet of Things deployments tend to prefer AES based pseudo-random

functions in order to avoid implementing SHA2. For the non-IoT VPN deployment it has been downgraded from SHOULD in RFC4307 to MAY as it has not seen wide adoption.

PRF_HMAC_MD5 has been downgraded from MAY in RFC4307 to MUST NOT. There is an industry-wide trend to deprecate its usage as MD5 support is being removed from cryptographic libraries in general because its non-HMAC use is known to be subject to collision attacks, for example as mentioned in [TRANSCRIPTION].

3.3. Type 3 - IKEv2 Integrity Algorithm Transforms

The algorithms in the below table are negotiated in the SA payload and used for the Encrypted Payload. References to the specification defining these algorithms are in the IANA registry. When an AEAD algorithm (see Section 3.1) is proposed, this algorithm transform type is not in use.

Name	Status	Comment
AUTH_HMAC_SHA2_256_128	MUST	
AUTH_HMAC_SHA2_512_256	SHOULD	
AUTH_HMAC_SHA1_96	MUST-	
AUTH_AES_XCBC_96	SHOULD	[IoT]
AUTH_HMAC_MD5_96	MUST NOT	
AUTH_DES_MAC	MUST NOT	
AUTH_KPDK_MD5	MUST NOT	

[IoT] - This requirement is for interoperability with IoT

AUTH_HMAC_SHA2_256_128 was not mentioned in RFC4307, as no SHA2 based transforms were mentioned. AUTH_HMAC_SHA2_256_128 MUST be implemented in order to replace AUTH_HMAC_SHA1_96.

AUTH_HMAC_SHA2_512_256 SHOULD be implemented as a future replacement of AUTH_HMAC_SHA2_256_128 or when stronger security is required. This value has been preferred over AUTH_HMAC_SHA2_384, as the additional overhead of AUTH_HMAC_SHA2_512 is negligible.

AUTH_HMAC_SHA1_96 has been downgraded from MUST in RFC4307 to MUST- as there is an industry-wide trend to deprecate its usage.

AUTH_AES-XCBC is only recommended in the scope of IoT, as Internet of Things deployments tend to prefer AES based pseudo-random functions in order to avoid implementing SHA2. For the non-IoT VPN deployment,

it has been downgraded from SHOULD in RFC4307 to MAY as it has not been widely adopted.

AUTH_DES_MAC, AUTH_HMAC_MD5_96, and AUTH_KPDK_MD5 were not mentioned in RFC4307 so their default status ware MAY. They have been downgraded to MUST NOT. There is an industry-wide trend to deprecate DES and MD5. MD5 support is being removed from cryptographic libraries in general because its non-HMAC use is known to be subject to collision attacks, for example as mentioned in [TRANSCRIPTION].

3.4. Type 4 - IKEv2 Diffie-Hellman Group Transforms

There are several Modular Exponential (MODP) groups and several Elliptic Curve groups (ECC) that are defined for use in IKEv2. These groups are defined in both the [IKEv2] base document and in extensions documents and are identified by group number. Note that it is critical to enforce a secure Diffie-Hellman exchange as this exchange provides keys for the session. If an attacker can retrieve the private numbers (a, or b) and the public values (g**a, and g**b), then the attacker can compute the secret and the keys used and decrypt the exchange and IPsec SA created inside the IKEv2 SA. Such an attack can be performed off-line on a previously recorded communication, years after the communication happened. This differs from attacks that need to be executed during the authentication which must be performed online and in near real-time.

Number	Description	Status
14	2048-bit MODP Group	MUST
19	256-bit random ECP group	SHOULD
5	1536-bit MODP Group	SHOULD NOT
2	1024-bit MODP Group	SHOULD NOT
1	768-bit MODP Group	MUST NOT
22	1024-bit MODP Group with 160-bit Prime Order Subgroup	SHOULD NOT
23	2048-bit MODP Group with 224-bit Prime Order Subgroup	SHOULD NOT
24	2048-bit MODP Group with 256-bit Prime Order Subgroup	SHOULD NOT

Group 14 or 2048-bit MODP Group is raised from SHOULD+ in RFC4307 as a replacement for 1024-bit MODP Group. Group 14 is widely implemented and considered secure.

Group 19 or 256-bit random ECP group was not specified in RFC4307, as this group were not specified at that time. Group 19 is widely implemented and considered secure.

Group 5 or 1536-bit MODP Group has been downgraded from MAY in RFC4307 to SHOULD NOT. It was specified earlier, but is now considered to be vulnerable to be broken within the next few years by a nation state level attack, so its security margin is considered too narrow.

Group 2 or 1024-bit MODP Group has been downgraded from MUST- in RFC4307 to SHOULD NOT. It is known to be weak against sufficiently funded attackers using commercially available mass-computing resources, so its security margin is considered too narrow. It is expected in the near future to be downgraded to MUST NOT.

Group 1 or 768-bit MODP Group was not mentioned in RFC4307 and so its status was MAY. It can be broken within hours using cheap of-the-shelves hardware. It provides no security whatsoever.

Group 22, 23 and 24 or 1024-bit MODP Group with 160-bit, and 2048-bit MODP Group with 224-bit and 256-bit Prime Order Subgroup have small subgroups, which means that checks specified in the "Additional Diffie-Hellman Test for the IKEv2" [RFC6989] section 2.2 first bullet point MUST be done when these groups are used. These groups are also not safe-primes. The seeds for these groups have not been publicly released, resulting in reduced trust in these groups. These groups were proposed as alternatives for group 2 and 14 but never saw wide deployment. It is expected in the near future to be further downgraded to MUST NOT.

4. IKEv2 Authentication

IKEv2 authentication may involve a signatures verification. Signatures may be used to validate a certificate or to check the signature of the AUTH value. Cryptographic recommendations regarding certificate validation are out of scope of this document. What is mandatory to implement is provided by the PKIX Community. This document is mostly concerned on signature verification and generation for the authentication.

4.1. IKEv2 Authentication Method

Number	Description	Status
1	RSA Digital Signature	MUST
2	Shared Key Message Integrity Code	MUST
3	DSS Digital Signature	SHOULD NOT
9	ECDSA with SHA-256 on the P-256 curve	SHOULD
10	ECDSA with SHA-384 on the P-384 curve	SHOULD
11	ECDSA with SHA-512 on the P-521 curve	SHOULD
14	Digital Signature	SHOULD

RSA Digital Signature is widely deployed and therefore kept for interoperability. It is expected to be downgraded in the future as its signatures are based on the older RSASSA-PKCS1-v1.5 which is no longer recommended. RSA authentication, as well as other specific Authentication Methods, are expected to be replaced with the generic Digital Signature method of [RFC7427]. RSA Digital Signature is not recommended for keys smaller then 2048, but since these signatures only have value in real-time, and need no future protection, smaller keys was kept at SHOULD NOT instead of MUST NOT.

Shared Key Message Integrity Code is widely deployed and mandatory to implement in the IKEv2 in the RFC7296.

ECDSA based Authentication Methods are also expected to be downgraded as it does not provide hash function agility. Instead, ECDSA (like RSA) is expected to be performed using the generic Digital Signature method.

DSS Digital Signature is bound to SHA-1 and has the same level of security as 1024-bit RSA. It is expected to be downgraded to MUST NOT in the future.

Digital Signature [RFC7427] is expected to be promoted as it provides hash function, signature format and algorithm agility.

4.1.1. Recommendations for RSA key length

Description	Status
RSA with key length 2048	MUST
RSA with key length 3072 and 4096	SHOULD
RSA with key length between 2049 and 4095	MAY
RSA with key length smaller than 2048	SHOULD NOT

The IKEv2 RFC7296 mandates support for the RSA keys of size 1024 or 2048 bits, but here we make key sizes less than 2048 SHOULD NOT as there is industry-wide trend to deprecate key lengths less than 2048 bits.

4.2. Digital Signature Recommendations

When Digital Signature authentication method is implemented, then the following recommendations are applied for hash functions:

Number	Description	Status	Comment
1	SHA1	MUST NOT	
2	SHA2-256	MUST	
3	SHA2-384	MAY	
4	SHA2-512	SHOULD	

When Digital Signature authentication method is used with RSA signature algorithm, then RSASSA-PSS MUST be supported and RSASSA-PKCS1-v1.5 MAY be supported.

The following table lists recommendations for authentication methods in RFC7427 [RFC7427] notation. These recommendations are applied only if Digital Signature authentication method is implemented.

Description	Status	Comment
RSASSA-PSS with SHA-256	MUST	
ecdsa-with-sha256	SHOULD	
sha1WithRSAEncryption	MUST NOT	
dsa-with-sha1	MUST NOT	
ecdsa-with-sha1	MUST NOT	
RSASSA-PSS with Empty Parameters	MUST NOT	
RSASSA-PSS with Default Parameters	MUST NOT	

5. Algorithms for Internet of Things

Some algorithms in this document are marked for use with the Internet of Things (IoT). There are several reasons why IoT devices prefer a different set of algorithms from regular IKEv2 clients. IoT devices are usually very constrained, meaning the memory size and CPU power is so limited, that these clients only have resources to implement and run one set of algorithms. For example, instead of implementing

AES and SHA, these devices typically use AES_XCBC as integrity algorithm so SHA does not need to be implemented.

For example, IEEE Std 802.15.4 [IEEE-802-15-4] devices have a mandatory to implement link level security using AES-CCM with 128 bit keys. The IEEE Recommended Practice for Transport of Key Management Protocol (KMP) Datagrams [IEEE-802-15-9] already provide a way to use Minimal IKEv2 [RFC7815] over 802.15.4 to provide link keys for the 802.15.4 layer.

These devices might want to use AES-CCM as their IKEv2 algorithm, so they can reuse the hardware implementing it. They cannot use the AES-CBC algorithm, as the hardware quite often do not include support for AES decryption needed to support the CBC mode. So despite the AES-CCM algorithm requiring AEAD [RFC5282] support, the benefit of reusing the crypto hardware makes AES-CCM the preferred algorithm.

Another important aspect of IoT devices is that their transfer rates are usually quite low (in order of tens of kbits/s), and each bit they transmit has an energy consumption cost associated with it and shortens their battery life. Therefore, shorter packets are preferred. This is the reason for recommending the 8 octet ICV over the 16 octet ICV.

Because different IoT devices will have different constraints, this document cannot specify the one mandatory profile for IoT. Instead, this document points out commonly used algorithms with IoT devices.

6. Security Considerations

The security of cryptographic-based systems depends on both the strength of the cryptographic algorithms chosen and the strength of the keys used with those algorithms. The security also depends on the engineering of the protocol used by the system to ensure that there are no non-cryptographic ways to bypass the security of the overall system.

The Diffie-Hellman Group parameter is the most important one to choose conservatively. Any party capturing all IKE and ESP traffic that (even years later) can break the selected DH group in IKE, can gain access to the symmetric keys used to encrypt all the ESP traffic. Therefore, these groups must be chosen very conservatively. However, specifying an extremely large DH group also puts a considerable load on the device, especially when this is a large VPN gateway or an IoT constrained device.

This document concerns itself with the selection of cryptographic algorithms for the use of IKEv2, specifically with the selection of

"mandatory-to-implement" algorithms. The algorithms identified in this document as "MUST implement" or "SHOULD implement" are not known to be broken at the current time, and cryptographic research so far leads us to believe that they will likely remain secure into the foreseeable future. However, this isn't necessarily forever and it is expected that new revisions of this document will be issued from time to time to reflect the current best practice in this area.

7. IANA Considerations

This document makes no requests of IANA.

8. Acknowledgements

The first version of this document was RFC 4307 by Jeffrey I. Schiller of the Massachusetts Institute of Technology (MIT). Much of the original text has been copied verbatim.

We would like to thank Paul Hoffman, Yaron Sheffer, John Mattsson and Tommy Pauly for their valuable feedback.

9. References

9.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <http://www.rfc-editor.org/info/rfc4106>.

[RFC4307] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", RFC 4307, DOI 10.17487/RFC4307, December 2005, <http://www.rfc-editor.org/info/rfc4307>.

[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>.

[RFC5282] Black, D. and D. McGrew, "Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol", RFC 5282, DOI 10.17487/RFC5282, August 2008, <http://www.rfc-editor.org/info/rfc5282>.

9.2. Informative References

[RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)", RFC 7427, DOI 10.17487/RFC7427, January 2015, <http://www.rfc-editor.org/info/rfc7427>.

[RFC6989] Sheffer, Y. and S. Fluhrer, "Additional Diffie-Hellman Tests for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 6989, DOI 10.17487/RFC6989, July 2013, <http://www.rfc-editor.org/info/rfc6989>.

[RFC7815] Kivinen, T., "Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation", RFC 7815, DOI 10.17487/RFC7815, March 2016, <http://www.rfc-editor.org/info/rfc7815>.

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

[TRANSCRIPTION] Bhargavan, K. and G. Leurent, "Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH", NDSS , feb 2016.

[IEEE-802-15-4] "IEEE Standard for Low-Rate Wireless Personal Area Networks (WPANs)", IEEE Standard 802.15.4, 2015.

[IEEE-802-15-9] "IEEE Recommended Practice for Transport of Key Management Protocol (KMP) Datagrams", IEEE Standard 802.15.9, 2016.

Authors' Addresses

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 6789735
Israel

EMail: ynir.ietf@gmail.com

Tero Kivinen
INSIDE Secure
Eerikinkatu 28
HELSINKI FI-00180
FI

EMail: kivinen@iki.fi

Paul Wouters
Red Hat

EMail: pwouters@redhat.com

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
EMail: daniel.migault@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 5, 2016

Y. Nir
Check Point
S. Josefsson
SJD

February 2, 2016

Curve25519 and Curve448 for IKEv2 Key Agreement
draft-ietf-ipsecme-safecurves-01

Abstract

This document describes the use of Curve25519 and Curve448 for ephemeral key exchange in the Internet Key Exchange (IKEv2) protocol.

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 August 5, 2016.

Copyright Notice

Table of Contents

1. Introduction

[RFC7748] describes two elliptic curves: Curve25519 and Curve448, as well as the X25519 and X448 functions for performing key agreement (Diffie-Hellman) operations with these curves. The curves and functions are designed for both performance and security.

Almost ten years ago [RFC4753] specified the first elliptic curve Diffie-Hellman groups for the Internet Key Exchange protocol (IKEv2 - [RFC7296]). These were the so-called NIST curves. The state of the art has advanced since then. More modern curves allow faster implementations while making it much easier to write constant-time implementations free from time-based side-channel attacks. This document defines two such curves for use in IKE. See [Curve25519] for details about the speed and security of the Curve25519 function.

1.1. Conventions Used in This Document

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 [RFC2119].

2. Curve25519 & Curve448

All cryptographic computations are done using the X25519 and X448 functions defined in [RFC7748]. All related parameters (for example, the base point) and the encoding (in particular, pruning the least/ most significant bits and use of little-endian encoding) are inherited from [RFC7748].

An ephemeral Diffie-Hellman key exchange using Curve25519 or Curve448 goes as follows: Each party picks a secret key d uniformly at random

and computes the corresponding public key. "X" is used below to denote either X25519 or X448, and "G" is used to denote the corresponding base point:

pub_mine = X(d, G)

Parties exchange their public keys (see Section 3.1) and compute a shared secret:

SHARED_SECRET = X(d, pub_peer).

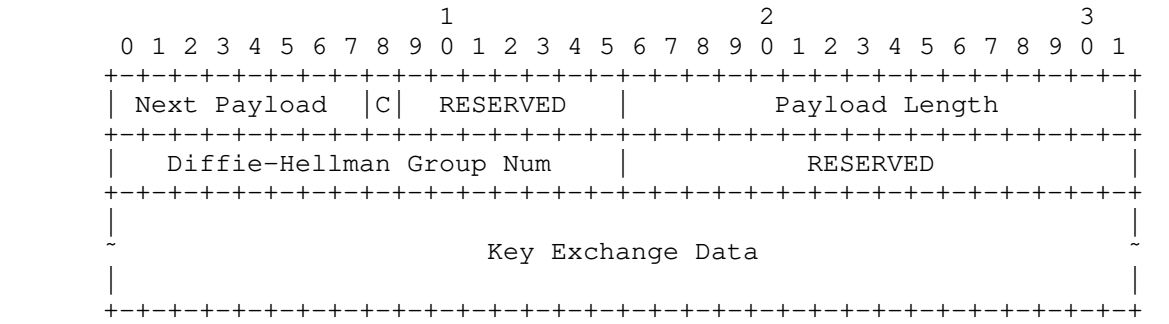
This shared secret is used directly as the value denoted g^ir in section 2.14 of RFC 7296. It is 32 octets when Curve25519 is used, and 56 octets when Curve448 is used.

3. Use and Negotiation in IKEv2

The use of Curve25519 and Curve448 in IKEv2 is negotiated using a Transform Type 4 (Diffie-Hellman group) in the SA payload of either an IKE_SA_INIT or a CREATE_CHILD_SA exchange. The value xx is used for the group defined by Curve25519 and yy is used for the group defined by Curve448. Both are TBA by IANA.

3.1. Key Exchange Payload

The diagram for the Key Exchange Payload from section 3.4 of RFC 7296 is copied below for convenience:



- Payload Length - For Curve25519 the public key is 32 octets, so the Payload Length field will be 40, and for Curve448 the public key is 56 octets, so the Payload Length field will be 64.
- The Diffie-Hellman Group Num is xx for Curve25519, or yy for Curve448 (both TBA by IANA).
- The Key Exchange Data is the 32 or 56 octets as described in section 6 of [RFC7748]

3.2. Recipient Tests

This document matches the discussion in [RFC7748] related to receiving and accepting incompatible point formats. In particular, receiving entities MUST mask the most-significant bit in the final byte for X25519 (but not X448), and implementations MUST accept non-canonical values. See section 5 of [RFC7748] for further discussion.

4. Security Considerations

Curve25519 and Curve448 are designed to facilitate the production of high-performance constant-time implementations. Implementors are encouraged to use a constant-time implementation of the functions. This point is of crucial importance if the implementation chooses to reuse its supposedly ephemeral key pair for many key exchanges, which some implementations do in order to improve performance.

Curve25519 is intended for the ~128-bit security level, comparable to the 256-bit random ECP group (group 19) defined in RFC 4753, also known as NIST P-256 or secp256r1. Curve448 is intended for the ~224-bit security level.

While the NIST curves are advertised as being chosen verifiably at random, there is no explanation for the seeds used to generate them. In contrast, the process used to pick these curves is fully documented and rigid enough so that independent verification has been done. This is widely seen as a security advantage, since it prevents the generating party from maliciously manipulating the parameters.

Another family of curves available in IKE, generated in a fully verifiable way, is the Brainpool curves [RFC6954]. For example, brainpoolP256 (group 28) is expected to provide a level of security comparable to Curve25519 and NIST P-256. However, due to the use of pseudo-random prime, it is significantly slower than NIST P-256, which is itself slower than Curve25519.

5. IANA Considerations

IANA is requested to assign two values from the IKEv2 "Transform Type 4 - Diffie-Hellman Group Transform IDs" registry, with names "Curve25519" and "Curve448" and this document as reference. The Recipient Tests field should also point to this document.

6. Acknowledgements

Curve25519 was designed by D. J. Bernstein and the parameters for Curve448 ("Goldilocks") is by Mike Hamburg. The specification of

algorithms, wire format and other considerations are in RFC 7748 by Adam Langley, Mike Hamburg, and Sean Turner.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, January 2016.

7.2. Informative References

[Curve25519] Bernstein, J., "Curve25519: New Diffie-Hellman Speed Records", LNCS 3958, February 2006, <http://dx.doi.org/10.1007/11745853_14>.

[RFC4753] Fu, D. and J. Solinas, "ECP Groups For IKE and IKEv2", RFC 4753, January 2007.

[RFC6954] Merkle, J. and M. Lochter, "Using the Elliptic Curve Cryptography (ECC) Brainpool Curves for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 6954, July 2013.

Authors' Addresses

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 6789735
Israel

Email: ynir.ietf@gmail.com

Simon Josefsson
SJD AB

Email: simon@josefsson.org

IPSECME
Internet-Draft
Intended status: Standards Track
Expires: December 22, 2017

D. Migault, Ed.
Ericsson
T. Guggemos, Ed.
LMU Munich
Y. Nir
Dell EMC

June 20, 2017

Implicit IV for Counter-based Ciphers in IPsec
draft-mglt-ipsecme-implicit-iv-04

Abstract

IPsec ESP sends an initialization vector (IV) or nonce in each packet, adding 8 or 16 octets. Some algorithms such as AES-GCM, AES-CCM, AES-CTR and ChaCha20-Poly1305 require a unique nonce but do not require an unpredictable nonce. When using such algorithms the packet counter value can be used to generate a nonce, saving 8 octets per packet. This document describes how to do this.

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 December 22, 2017.

Copyright Notice

Table of Contents

1. Requirements notation

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 [RFC2119].

2. Introduction

Counter-based AES modes of operation such as AES-CTR ([RFC3686]), AES-CCM ([RFC4309]), and AES-GCM ([RFC4106]) require the specification of an nonce for each ESP packet. The same applies for ChaCha20-Poly1305 ([RFC7634]. Currently this nonce is sent in each ESP packet ([RFC4303]). This practice is designated in this document as "explicit nonce".

In some context, such as IoT, it may be preferable to avoid carrying the extra bytes associated to the IV and instead generate it locally on each peer. The local generation of the nonce is designated in this document as "implicit IV".

The size of this nonce depends on the specific algorithm, but all of the algorithms mentioned above take an 8-octet nonce.

This document defines how to compute the nonce locally when it is implicit. It also specifies how peers agree with the Internet Key Exchange version 2 (IKEv2 - [RFC7296]) on using an implicit IV versus an explicit IV.

This document limits its scope to the algorithms mentioned above. Other algorithms with similar properties may later be defined to use this extension.

This document does not consider AES-CBC ([RFC3602]) as AES-CBC requires the IV to be unpredictable. Deriving it directly from the packet counter as described below is insecure as mentioned in Security Consideration of [RFC3602] and has led to real world chosen plain-text attack such as BEAST [BEAST].

3. Terminology

- IoT: Internet of Things.
- IV: Initialization Vector.
- Nonce: a fixed-size octet string used only once. This is similar to IV, except that in common usage there is no implication of non-predictability.

4. Implicit IV

With the algorithms listed in Section 2, the 8 byte nonce MUST NOT repeat. The binding between a ESP packet and its nonce is provided using the Sequence Number or the Extended Sequence Number. Figure 1 and Figure 2 represent the IV with a regular 4-byte Sequence Number and with an 8-byte Extended Sequence Number respectively.

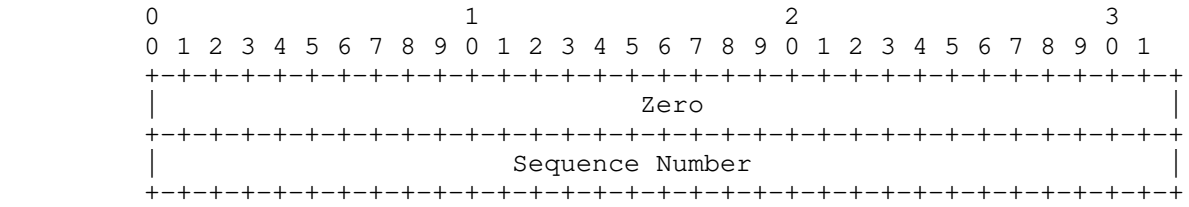


Figure 1: Implicit IV with a 4 byte Sequence Number

- Sequence Number: the 4 byte Sequence Number carried in the ESP packet.
- Zero: a 4 byte array with all bits set to zero.

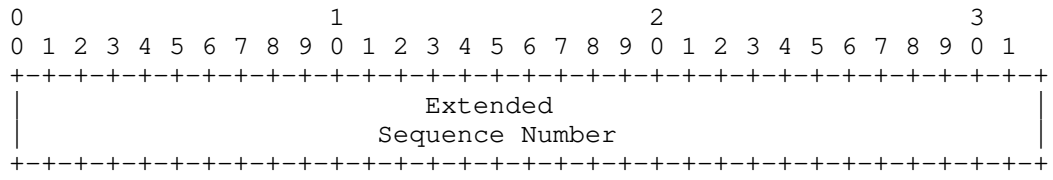


Figure 2: Implicit IV with an 8 byte Extended Sequence Number

- Extended Sequence Number: the 8 byte Extended Sequence Number of the Security Association. The 4 byte low order bytes are carried in the ESP packet.

5. Initiator Behavior

An initiator supporting this feature SHOULD propose implicit IV for all relevant algorithms. To facilitate backward compatibility with non-supporting peers the initiator SHOULD also include those same algorithms without IIV. This may require extra transforms.

6. Responder Behavior

The rules of SA payload processing ensure that the responder will never send an SA payload containing the IIV indicator to an initiator that does not support IIV.

7. Security Consideration

Nonce generation for these algorithms has not been explicitly defined. It has been left to the implementation as long as certain security requirements are met. This document provides an explicit and normative way to generate IVs. The mechanism described in this document meets the IV security requirements of all relevant algorithms.

As the IV MUST NOT repeat for one SPI when Counter-Mode ciphers are used, Implicit IV as described in this document MUST NOT be used in setups with the chance that the Sequence Number overlaps for one SPI. Multicast as described in [RFC5374], [RFC6407] and [I-D.yeung-g-ikev2] is a prominent example, where many senders share one secret and thus one SPI. Section 3.5 of [RFC6407] explains how repetition MAY BE prevented by using a prefix for each group member, which could be prefixed to the Sequence Number. Otherwise, Implicit IV MUST NOT be used in multicast scenarios.

8. IANA Considerations

AES-CTR, AES-CCM, AES-GCM and ChaCha20-Poly1305 are likely to implement the implicit IV described in this document. This section limits assignment of new code points to the recommended suites provided in [I-D.ietf-ipsecme-rfc4307bis] and [I-D.ietf-ipsecme-rfc7321bis], thus the new Transform Type 1 - Encryption Algorithm Transform IDs are as defined below:

- ENCR_AES-CCM_8_IIV
- ENCR_AES-GCM_16_IIV
- ENCR_CHACHA20-POLY1305_IIV

9. References

9.1. Normative References

[I-D.ietf-ipsecme-rfc4307bis]
Nir, Y., Kivinen, T., Wouters, P., and D. Migault, "Algorithm Implementation Requirements and Usage Guidance for IKEv2", draft-ietf-ipsecme-rfc4307bis-18 (work in progress), March 2017.

[I-D.ietf-ipsecme-rfc7321bis]
Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T. Kivinen, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", draft-ietf-ipsecme-rfc7321bis-06 (work in progress), June 2017.

[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>.

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, DOI 10.17487/RFC3602, September 2003, <http://www.rfc-editor.org/info/rfc3602>.

[RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, DOI 10.17487/RFC3686, January 2004, <http://www.rfc-editor.org/info/rfc3686>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <http://www.rfc-editor.org/info/rfc4106>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <http://www.rfc-editor.org/info/rfc4309>.

[RFC5374] Weis, B., Gross, G., and D. Ignjatic, "Multicast Extensions to the Security Architecture for the Internet Protocol", RFC 5374, DOI 10.17487/RFC5374, November 2008, <http://www.rfc-editor.org/info/rfc5374>.

[RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <http://www.rfc-editor.org/info/rfc6407>.

[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>.

[RFC7634] Nir, Y., "ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec", RFC 7634, DOI 10.17487/RFC7634, August 2015, <http://www.rfc-editor.org/info/rfc7634>.

9.2. Informational References

[BEAST] Thai, T. and J. Juliano, "Here Come The xor Ninjas", , May 2011, <https://www.researchgate.net/publication/266529975_Here_Come_The_Ninjas>.

[I-D.yeung-g-ikev2] Weis, B., Nir, Y., and V. Smyslov, "Group Key Management using IKEv2", draft-yeung-g-ikev2-11 (work in progress), March 2017.

Authors' Addresses

Daniel Migault (editor)
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Email: daniel.migault@ericsson.com

Tobias Guggemos (editor)
LMU Munich
Oettingenstr. 67
80538 Munich, Bavaria
Germany

Email: guggemos@mnm-team.org
URI: http://mnm-team.org/~guggemos

Yoav Nir
Dell EMC
9 Andrei Sakharov St
Haifa 3190500
Israel

Email: ynir.ietf@gmail.com

Network Working Group
Internet-Draft
Obsoletes: 7321 (if approved)
Intended status: Standards Track
Expires: March 13, 2017

D. Migault
J. Mattsson
Ericsson
P. Wouters
Red Hat
Y. Nir
Check Point
T. Kivinen
INSIDE Secure

September 9, 2016

Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH) draft-mglt-ipsecme-rfc7321bis-04

Abstract

This document updates the Cryptographic Algorithm Implementation Requirements for ESP and AH. The goal of these document is to enable ESP and AH to benefit from cryptography that is up to date while making IPsec interoperable.

This document obsoletes RFC 7321 on the cryptographic recommendations only.

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 March 13, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Encapsulating Security Payload (ESP) [RFC4303] and the Authentication Header (AH) [RFC4302] are the mechanisms for applying cryptographic protection to data being sent over an IPsec Security Association (SA) [RFC4301].

This document provides guidance and recommendations so that ESP and AH can be used with a cryptographic algorithms that are up to date. The challenge of such document is to make sure that over the time IPsec implementations can use secure and up-to-date cryptographic algorithms while keeping IPsec interoperable.

1.1. Updating Algorithm Implementation Requirements and Usage Guidance

The field of cryptography evolves continuously. New stronger algorithms appear and existing algorithms are found to be less secure then originally thought. Therefore, algorithm implementation

requirements and usage guidance need to be updated from time to time to reflect the new reality. The choices for algorithms must be conservative to minimize the risk of algorithm compromise. Algorithms need to be suitable for a wide variety of CPU architectures and device deployments ranging from high end bulk encryption devices to small low-power IoT devices.

The algorithm implementation requirements and usage guidance may need to change over time to adapt to the changing world. For this reason, the selection of mandatory-to-implement algorithms was removed from the main IKEv2 specification and placed in a separate document.

1.2. Updating Algorithm Requirement Levels

The mandatory-to-implement algorithm of tomorrow should already be available in most implementations of AH/ESP by the time it is made mandatory. This document attempts to identify and introduce those algorithms for future mandatory-to-implement status. There is no guarantee that the algorithms in use today may become mandatory in the future. Published algorithms are continuously subjected to cryptographic attack and may become too weak or could become completely broken before this document is updated.

This document only provides recommendations for the mandatory-to-implement algorithms or algorithms too weak that are recommended not to be implemented. As a result, any algorithm listed at the IPsec IANA registry not mentioned in this document MAY be implemented. As [RFC7321] omitted most of the algorithms mentioned by the IPsec IANA repository, which makes it difficult to define whether non mentioned algorithms are optional to implement or must not be implemented as they are too weak. This document provides explicit guidance for all of them. It is expected that this document will be updated over time and next versions will only mention algorithms which status has evolved. For clarification when an algorithm has been mentioned in [RFC7321], this document states explicitly the update of the status.

Although this document updates the algorithms to keep the AH/ESP communication secure over time, it also aims at providing recommendations so that AH/ESP implementations remain interoperable. AH/ESP interoperability is addressed by an incremental introduction or deprecation of algorithms. In addition, this document also considers the new use cases for AH/ESP deployment, such as Internet of Things (IoT).

It is expected that deprecation of an algorithm is performed gradually. This provides time for various implementations to update their implemented algorithms while remaining interoperable. Unless there are strong security reasons, an algorithm is expected to be

downgraded from MUST to MUST- or SHOULD, instead of MUST NOT. Similarly, an algorithm that has not been mentioned as mandatory-to-implement is expected to be introduced with a SHOULD instead of a MUST.

The current trend toward Internet of Things and its adoption of AH/ESP requires this specific use case to be taken into account as well. IoT devices are resource constrained devices and their choice of algorithms are motivated by minimizing the footprint of the code, the computation effort and the size of the messages to send. This document indicates "[IoT]" when a specified algorithm is specifically listed for IoT devices. Requirement levels that are marked as "IoT" apply to IoT devices and to server-side implementations that might presumably need to interoperate with them, including any general-purpose VPN gateways.

1.3. Document Audience

The recommendations of this document mostly target AH/ESP implementers as implementations need to meet both high security expectations as well as high interoperability between various vendors and with different versions. Interoperability requires a smooth move to more secure cipher suites. This may differ from a user point of view that may deploy and configure AH/ESP with only the safest cipher suite.

This document does not give any recommendations for the use of algorithms, it only gives implementation recommendations for implementations. The use of algorithms by users is dictated by the security policy requirements for that specific user, and are outside the scope of this document.

The algorithms considered here are listed by the IANA as part of the IKEv2 parameters. IKEv1 is out of scope of this document. IKEv1 is deprecated and the recommendations of this document must not be considered for IKEv1, nor IKEv1 parameters be considered by this document.

The IANA registry for Internet Key Exchange Version 2 (IKEv2) Parameters contains some entries that are not for use with ESP or AH. This document does not modify the status of those algorithms.

2. 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 [RFC2119].

Following [RFC4835], we define some additional key words:

MUST- This term means the same as MUST. However, we expect that at some point in the future this algorithm will no longer be a MUST.
SHOULD+ This term means the same as SHOULD. However, it is likely that an algorithm marked as SHOULD+ will be promoted at some future time to be a MUST.

3. ESP Encryption Algorithms

Name	Status	AEAD	Comment
ENCR_DES_IV64	MUST NOT	No	UNSPECIFIED
ENCR_DES	MUST NOT	No	[RFC2405]
ENCR_3DES	SHOULD NOT	No	[RFC2451]
ENCR_BLOWFISH	MUST NOT	No	[RFC2451]
ENCR_3IDEA	MUST NOT	No	UNSPECIFIED
ENCR_DES_IV32	MUST NOT	No	UNSPECIFIED
ENCR_NULL	MUST	No	[RFC2410]
ENCR_AES_CBC	MUST	No	[RFC3602][1]
ENCR_AES_CCM_8	SHOULD	Yes	[RFC4309]IoT]
ENCR_AES_GCM_16	MUST	Yes	[RFC4106][1]
ENCR_CHACHA20_POLY1305	SHOULD	Yes	[RFC7634]

[1] - This requirement level is for 128-bit and 256-bit keys. 192-bit keys remain at MAY level. [IoT] - This requirement is for interoperability with IoT. Only 128-bit keys are at MUST level. 192-bit and 256-bit keys are at the MAY level.

IPsec sessions may have very long life time, and carry multiple packets, so there is a need to move 256-bit keys in the long term. For that purpose requirement level is for 128 bit keys and 256 bit keys are at SHOULD (when applicable). In that sense 256 bit keys status has been raised from MAY in RFC7321 to SHOULD.

IANA has allocated codes for cryptographic algorithms that have not been specified by the IETF. Such algorithms are noted as UNSPECIFIED. Usually, the use of theses algorithms is limited to specific cases, and the absence of specification makes interoperability difficult for IPsec communications. These algorithms were not been mentioned in [RFC7321] and this document clarify that such algorithms MUST NOT be implemented for IPsec communications.

Similarly IANA also allocated code points for algorithms that are not expected to be used to secure IPsec communications. Such algorithms

are noted as Non IPsec. As a result, these algorithms MUST NOT be implemented.

Various older and not well tested and never widely implemented ciphers have been changed to MUST NOT.

ENCR_3DES status has been downgraded from MAY in RFC7321 to SHOULD NOT. ENCR_CHACHA20_POLY1305 is a more modern approach alternative for ENCR_3DES than ENCR_AES_CBC and so it expected to be favored to replace ENCR_3DES.

ENCR_BLOWFISH has been downgraded to MUST NOT as it has been deprecated for years by TWOFISH, which is not standarized for ESP and therefor not listed in this document. Some implementations support TWOFISH using a private range number.

ENCR_NULL status was set to MUST in [RFC7321] and remains a MUST to enable the use of ESP with only authentication which is preferred over AH due to NAT traversal. ENCR_NULL is expected to remain MUST by protocol requirements.

ENCR_AES_CBC status remains to MUST. ENCR_AES_CBC MUST be implemented in order to enable interoperability between implementation that followed RFC7321. However, there is a trend for the industry to move to AEAD encryption, and the overhead of ENCR_AES_CBC remains quite large so it is expected to be replaced by AEAD algorithms in the long term.

ENCR_AES_CCM_8 status was set to MAY in [RFC7321] and has been raised from MAY to SHOULD in order to interact with Internet of Things devices. As this case is not a general use case for VPNs, its status is expected to remain as SHOULD.

ENCR_AES_GCM_16 status has been updated from SHOULD+ to MUST in order to favor the use of authenticated encryption and AEAD algorithms. ENCR_AES_GCM_16 has been widely implemented for ESP due to its increased performance and key longevity compared to ENCR_AES_CBC.

ENCR_CHACHA20_POLY1305 was not ready to be considered at the time of RFC7321. It has been recommended by the CRFG and others as an alternative to ENCR_AES_XCBC and ENCR_AES_GCM_*. It is also being standardized for ESP for the same reasons. At the time of writing, there are not enough ESP implementations of ENCR_CHACHA20_POLY1305 to be able to introduce it at the SHOULD+ level. Its status has been set to SHOULD and is expected to become MUST in the long term.

4. ESP and AH Authentication Algorithms

Encryption without authentication MUST NOT be used. As a result, authentication algorithm recommendations in this section are targeting two types of communications: Firstly authenticated only communications without encryption. Such communications can be ESP with NULL encryption or AH communications. Secondly, communications that are encrypted with non AEAD encryption algorithms mentioned above. In this case, they MUST be combined with an authentication algorithm.

Name	Status	Comment
AUTH_NONE	MUST / MUST NOT	[RFC7296] AEAD
AUTH_HMAC_MD5_96	MUST NOT	[RFC2403][RFC7296]
AUTH_HMAC_SHA1_96	MUST-	[RFC2404][RFC7296]
AUTH_DES_MAC	MUST NOT	[UNSPECIFIED]
AUTH_KPDK_MD5	MUST NOT	[UNSPECIFIED]
AUTH_AES_XCBC_96	SHOULD	[RFC3566][RFC7296] [IoT]
AUTH_AES_128_GMAC	MAY	[RFC4543]
AUTH_AES_256_GMAC	MAY	[RFC4543]
AUTH_HMAC_SHA2_256_128	MUST	[RFC4868]
AUTH_HMAC_SHA2_512_256	SHOULD	[RFC4868]

[IoT] - This requirement is for interoperability with IoT

AUTH_NONE has been downgraded from MAY in RFC7321 to MUST NOT. The only reason NULL is acceptable is when authenticated encryption algorithms are selected from Section 3. In all other case, NULL MUST NOT be selected. As ESP and AH provides both authentication, one may be tempted to combine these protocol to provide authentication. As mentioned by RFC7321, it is NOT RECOMMENDED to use ESP with NULL authentication - with non authenticated encryption - in conjunction with AH; some configurations of this combination of services have been shown to be insecure [PD10]. In addition, NULL authentication cannot be combined with ESP NULL encryption.

AUTH_HMAC_MD5_96 and AUTH_KPDK_MD5 were not mentioned in RFC7321. As MD5 is known to be vulnerable to collisions, these algorithms MUST NOT be used.

AUTH_HMAC_SHA1_96 has been downgraded from MUST in RFC7321 to MUST- as there is an industry-wide trend to deprecate its usage.

AUTH_DES_MAC was not mentioned in RFC7321. As DES is known to be vulnerable, it MUST NOT be used.

AUTH_AES_XCBC_96 is only recommended in the scope of IoT, as Internet of Things deployments tend to prefer AES based HMAC functions in order to avoid implementing SHA2. For the wide VPN deployment, as it has not been widely adopted, it has been downgraded from SHOULD to MAY.

AUTH_AES_128_GMAC status has been downgraded from SHOULD+ to MAY. Along with AUTH_AES_192_GMAC and AUTH_AES_256_GMAC, these algorithms should only be used for AH not for ESP. If using ENCR_NULL, AUTH_HMAC_SHA2_256_128 is recommended for integrity. If using GMAC without authentication, ENCR_NULL_AUTH_AES_GMAC is recommended. Therefore, these ciphers are kept at MAY.

AUTH_HMAC_SHA2_256_128 was not mentioned in RFC7321, as no SHA2 based authentication was mentioned. AUTH_HMAC_SHA2_256_128 MUST be implemented in order to replace AUTH_HMAC_SHA1_96. Note that due to a long standing common implementation bug of this algorithm that truncates the hash at 96-bits instead of 128-bits, it is recommended that implementations prefer AUTH_HMAC_SHA2_512_256 over AUTH_HMAC_SHA2_256_128 if they implement AUTH_HMAC_SHA2_512_256.

AUTH_HMAC_SHA2_512_256 SHOULD be implemented as a future replacement of AUTH_HMAC_SHA2_256_128 or when stronger security is required. This value has been preferred to AUTH_HMAC_SHA2_384, as the additional overhead of AUTH_HMAC_SHA2_512 is negligible.

5. ESP and AH Compression Algorithms

Name	Status	Comment
IPCOMP_OUI	MUST NOT	UNSPECIFIED
IPCOMP_DEFLATE	MAY	[RFC2393]
IPCOMP_LZS	MAY	[RFC2395]
IPCOMP_LZJH	MAY	[RFC3051]

[IoT] - This requirement is for interoperability with IoT

Compression was not mentioned in RFC7321. As it is not widely deployed, it remains optional and at the MAY-level.

6. Summary of Changes from RFC 7321

The following table summarizes the changes from RFC 7321.

RFC EDITOR: PLEASE REMOVE THIS PARAGRAPH AND REPLACE XXXX IN THE TABLE BELOW WITH THE NUMBER OF THIS RFC

Algorithm	RFC 7321	RFC XXXX
ENCR_AES_GCM_16	SHOULD+	MUST
ENCR_AES_CCM_8	MAY	SHOULD
ENCR_AES_CTR	MAY	(*)
ENCR_3DES	MAY	SHOULD NOT
AUTH_HMAC_SHA1_96	MUST	MUST-
AUTH_AES_128_GMAC	SHOULD+	MAY
AUTH_NONE	MAY	MUST / MUST NOT

(*) This algorithm is not mentioned in the above sections, so it defaults to MAY.

7. Acknowledgements

Some of the wording in this document was adapted from [RFC7321], the document that this one obsoletes, which was written by D. McGrew and P. Hoffman.

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

The security of a system that uses cryptography depends on both the strength of the cryptographic algorithms chosen and the strength of the keys used with those algorithms. The security also depends on the engineering and administration of the protocol used by the system to ensure that there are no non-cryptographic ways to bypass the security of the overall system.

This document concerns itself with the selection of cryptographic algorithms for the use of ESP and AH, specifically with the selection of mandatory-to-implement algorithms. The algorithms identified in this document as "MUST implement" or "SHOULD implement" are not known to be broken at the current time, and cryptographic research to date leads us to believe that they will likely remain secure into the foreseeable future. However, this is not necessarily forever.

Therefore, we expect that revisions of that document will be issued from time to time to reflect the current best practice in this area.

10. References

10.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <http://www.rfc-editor.org/info/rfc4301>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <http://www.rfc-editor.org/info/rfc4302>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <http://www.rfc-editor.org/info/rfc4303>.

[RFC7321] McGrew, D. and P. Hoffman, "Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 7321, DOI 10.17487/RFC7321, August 2014, <http://www.rfc-editor.org/info/rfc7321>.

[RFC7634] Nir, Y., "ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec", RFC 7634, DOI 10.17487/RFC7634, August 2015, <http://www.rfc-editor.org/info/rfc7634>.

10.2. Informative References

[PD10] Paterson, K. and J. Degabriele, "On the (in)security of IPsec in MAC-then-encrypt configurations (ACM Conference on Computer and Communications Security, ACM CCS)", 2010.

[RFC2393] Shacham, A., Monsour, R., Pereira, R., and M. Thomas, "IP Payload Compression Protocol (IPComp)", RFC 2393, DOI 10.17487/RFC2393, December 1998, <http://www.rfc-editor.org/info/rfc2393>.

[RFC2395] Friend, R. and R. Monsour, "IP Payload Compression Using LZS", RFC 2395, DOI 10.17487/RFC2395, December 1998, <http://www.rfc-editor.org/info/rfc2395>.

[RFC2403] Madson, C. and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH", RFC 2403, DOI 10.17487/RFC2403, November 1998, <http://www.rfc-editor.org/info/rfc2403>.

[RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, DOI 10.17487/RFC2404, November 1998, <http://www.rfc-editor.org/info/rfc2404>.

[RFC2405] Madson, C. and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm With Explicit IV", RFC 2405, DOI 10.17487/RFC2405, November 1998, <http://www.rfc-editor.org/info/rfc2405>.

[RFC2410] Glenn, R. and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, DOI 10.17487/RFC2410, November 1998, <http://www.rfc-editor.org/info/rfc2410>.

[RFC2451] Pereira, R. and R. Adams, "The ESP CBC-Mode Cipher Algorithms", RFC 2451, DOI 10.17487/RFC2451, November 1998, <http://www.rfc-editor.org/info/rfc2451>.

[RFC3051] Heath, J. and J. Border, "IP Payload Compression Using ITU-T V.44 Packet Method", RFC 3051, DOI 10.17487/RFC3051, January 2001, <http://www.rfc-editor.org/info/rfc3051>.

[RFC3566] Frankel, S. and H. Herbert, "The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec", RFC 3566, DOI 10.17487/RFC3566, September 2003, <http://www.rfc-editor.org/info/rfc3566>.

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, DOI 10.17487/RFC3602, September 2003, <http://www.rfc-editor.org/info/rfc3602>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <http://www.rfc-editor.org/info/rfc4106>.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <http://www.rfc-editor.org/info/rfc4309>.

[RFC4543] McGrew, D. and J. Viega, "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, DOI 10.17487/RFC4543, May 2006, <http://www.rfc-editor.org/info/rfc4543>.

[RFC4835] Manral, V., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4835, DOI 10.17487/RFC4835, April 2007, <http://www.rfc-editor.org/info/rfc4835>.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <http://www.rfc-editor.org/info/rfc4868>.

[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>.

Authors' Addresses

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
Email: daniel.migault@ericsson.com

John Mattsson
Ericsson AB
SE-164 80 Stockholm
Sweden

Email: john.mattsson@ericsson.com

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 6789735
Israel

Email: ynir.ietf@gmail.com

Tero Kivinen
INSIDE Secure
Eerikinkatu 28
HELSINKI FI-00180
FI

Email: kivinen@iki.fi

Network
Internet-Draft
Intended status: Standards Track
Expires: March 25, 2017

T. Pauly
Apple Inc.
P. Wouters
Red Hat

September 21, 2016

Split DNS Configuration for IKEv2
draft-pauly-ipsecme-split-dns-02

Abstract

This document defines two Configuration Payload Attribute Types for the IKEv2 protocol that add support for private DNS domains. These domains should be resolved using DNS servers reachable through an IPsec connection, while leaving all other DNS resolution unchanged. This approach of resolving a subset of domains using non-public DNS servers is referred to as "Split DNS".

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 March 25, 2017.

Table of Contents

1. Introduction

Split DNS is a common configuration for secure tunnels, such as Virtual Private Networks in which host machines private to an organization can only be resolved using internal DNS resolvers [RFC2775]. In such configurations, it is often desirable to only resolve hosts within a set of private domains using the tunnel, while letting resolutions for public hosts be handled by a device's default DNS configuration.

The Internet Key Exchange protocol version 2 [RFC7296] negotiates configuration parameters using Configuration Payload Attribute Types. This document defines two Configuration Payload Attribute Types that add support for trusted Split DNS domains.

The INTERNAL_DNS_DOMAIN attribute type is used to convey one or more DNS domains that should be resolved only using the provided DNS nameserver IP addresses, causing these requests to use the IPsec connection.

The INTERNAL_DNSSEC_TA attribute type is used to convey DNSSEC trust anchors for those domains.

When only a subset of traffic is routed into a private network using an IPsec SA, these Configuration Payload options can be used to define which private domains should be resolved through the IPsec connection without affecting the client's global DNS resolution.

For the purposes of this document, DNS resolution servers accessible through an IPsec connection will be referred to as "internal DNS servers", and other DNS servers will be referred to as "external DNS servers".

A client using these configuration payloads will be able to request and receive Split DNS configurations using the INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA configuration attributes. The client device can use the internal DNS server(s) for any DNS queries within the assigned domains, while routing other DNS queries to its regular external DNS server.

1.1. 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. Background

Split DNS is a common configuration for enterprise VPN deployments, in which only one or a few private DNS domains are accessible and resolvable via an IPsec based VPN connection.

Other tunnel-establishment protocols already support the assignment of Split DNS domains. For example, there are proprietary extensions to IKEv1 that allow a server to assign Split DNS domains to a client. However, the IKEv2 standard does not include a method to configure this option. This document defines a standard way to negotiate this option for IKEv2.

3. Protocol Exchange

In order to negotiate which domains are considered internal to an IKEv2 tunnel, initiators indicate support for Split DNS in their CFG_REQUEST payloads, and responders assign internal domains (and DNSSEC trust anchors) in their CFG_REPLY payloads. When Split DNS has been negotiated, the existing DNS server configuration attributes will be interpreted as internal DNS servers that can resolve hostnames within the internal domains.

3.1. Configuration Request

To indicate support for Split DNS, an initiator sending a CFG_REQUEST payload MAY include one or more INTERNAL_DNS_DOMAIN attributes as defined in Section 4. If an INTERNAL_DNS_DOMAIN attribute is included in the CFG_REQUEST, the initiator SHOULD also include one or both of the INTERNAL_IP4_DNS and INTERNAL_IP6_DNS attributes in its CFG_REQUEST.

If the length of the INTERNAL_DNS_DOMAIN attribute is zero, then the initiator is requesting that the attribute be assigned without restricting the subdomains that it will accept.

If the length of the INTERNAL_DNS_DOMAIN is greater than zero, the value is a single DNS domain. The initiator is indicating that it will only allow this domain and any sub-domains within this domain to be resolved using the internal DNS servers. The list of INTERNAL_DNS_DOMAIN attributes in the CFG_REQUEST defines the full set of domains the intiator is willing to resolve using the internal DNS servers.

The absence of INTERNAL_DNS_DOMAIN attributes in the CFG_REQUEST payload indicates that the initiator does not support or is unwilling to accept Split DNS configuration.

To indicate support for DNSSEC, an initiator sending a CFG_REQUEST payload MAY include one or more INTERNAL_DNS_TA attributes as defined in Section 4. These payloads MUST immediately follow a INTERNAL_DNS_DOMAIN attribute, which binds the DNSSEC trust anchor request to the domain.

An initiator MAY convey its current DNSSEC trust anchors for the domain specified in the INTERNAL_DNS_DOMAIN attribute. If it does not wish to convey this information, it MUST use a length of 0.

The absence of INTERNAL_DNS_TA attributes in the CFG_REQUEST payload indicates that the initiator does not support or is unwilling to accept DNSSEC trust anchor configuration.

3.2. Configuration Reply

Responders MAY send one or more INTERNAL_DNS_DOMAIN attributes in their CFG_REPLY payload if the CFG_REQUEST contained at least one INTERNAL_DNS_DOMAIN attribute. If the CFG_REQUEST did not contain an INTERNAL_DNS_DOMAIN attribute, the responder MUST NOT include an INTERNAL_DNS_DOMAIN attribute in the CFG_REPLY. If an INTERNAL_DNS_DOMAIN attribute is included in the CFG_REPLY, the responder SHOULD also include one or both of the INTERNAL_IP4_DNS and

INTERNAL_IP6_DNS attributes in its CFG_REPLY. These DNS server configurations are necessary to define which servers should receive queries for hostnames in internal domains. If the CFG_REQUEST included an INTERNAL_DNS_DOMAIN attribute, but the CFG_REPLY does not include an INTERNAL_DNS_DOMAIN attribute, the initiator should behave as if Split DNS configurations are not supported by the server.

Each INTERNAL_DNS_DOMAIN represents a domain that the DNS servers address listed in INTERNAL_IP4_DNS and INTERNAL_IP6_DNS can resolve.

If the CFG_REQUEST included INTERNAL_DNS_DOMAIN attributes with non-zero lengths, the CFG_REPLY MUST NOT assign any domains in its INTERNAL_DNS_DOMAIN attributes that are not contained within the requested domains. The initiator SHOULD ignore any domains beyond its requested list.

For each DNS domain specified in an INTERNAL_DNS_DOMAIN attribute, one or more INTERNAL_DNSSEC_TA attributes MAY be included by the responder. This attribute lists the corresponding DSSNEC trust anchor in the DNS wire format of a DS record as specified in [RFC4034]. The INTERNAL_DNSSEC_TA attribute MUST immediately follow the INTERNAL_DNS_DOMAIN attribute that it applies to.

3.3. Mapping DNS Servers to Domains

All DNS servers provided in the CFG_REPLY MUST support resolving hostnames within all INTERNAL_DNS_DOMAIN domains. In other words, the INTERNAL_DNS_DOMAIN attributes in a CFG_REPLY payload form a single list of Split DNS domains that applies to the entire list of INTERNAL_IP4_DNS and INTERNAL_IP6_DNS attributes.

3.4. Example Exchanges

3.4.1. Simple Case

In this example exchange, the initiator requests INTERNAL_IP4_DNS and INTERNAL_DNS_DOMAIN attributes in its CFG_REQUEST, but does not specify any value for either. This indicates that it supports Split DNS, but has no preference for which DNS requests should be routed through the tunnel.

The responder replies with two DNS server addresses, and one internal domain, "example.com".

Any subsequent DNS queries from the initiator for domains such as "www.example.com" should use 198.51.100.2 or 198.51.100.4 to resolve.

```
CP(CFG_REQUEST) =
  INTERNAL_IP4_ADDRESS()
  INTERNAL_IP4_DNS()
  INTERNAL_DNS_DOMAIN()

CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_DNS(198.51.100.2)
  INTERNAL_IP4_DNS(198.51.100.4)
  INTERNAL_DNS_DOMAIN(example.com)
```

3.4.2. Requesting Limited Domains

In this example exchange, the initiator requests INTERNAL_IP4_DNS and INTERNAL_DNS_DOMAIN attributes in its CFG_REQUEST, specifically requesting only "example.com" and "other.com". The responder replies with two DNS server addresses, 198.51.100.2 and 198.51.100.4, and two domains, "example.com" and "city.other.com". Note that one of the domains in the CFG_REPLY, "city.other.com", is a subset of the requested domain, "other.com". This indicates that hosts within "other.com" that are not within "city.other.com" should be resolved using an external DNS server. The CFG_REPLY would not be allowed to respond with "com" or "example.net", however, since these were contained within the limited set of requested domains.

Any subsequent DNS queries from the initiator for domains such as "www.example.com" or "city.other.com" should use 198.51.100.2 or 198.51.100.4 to resolve.

```
CP(CFG_REQUEST) =
  INTERNAL_IP4_ADDRESS()
  INTERNAL_IP4_DNS()
  INTERNAL_DNS_DOMAIN(example.com)
  INTERNAL_DNS_DOMAIN(other.com)

CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_DNS(198.51.100.2)
  INTERNAL_IP4_DNS(198.51.100.4)
  INTERNAL_DNS_DOMAIN(example.com)
  INTERNAL_DNS_DOMAIN(city.other.com)
```

3.4.3. Requesting Domains and DNSSEC trust anchors

In this example exchange, the initiator requests INTERNAL_IP4_DNS, INTERNAL_DNS_DOMAIN and INTERNAL_DNS_TA attributess in its CFG_REQUEST

Any subsequent DNS queries from the initiator for domains such as "www.example.com" or "city.other.com" would be DNSSEC validated using the DNSSEC trust anchor received in the CFG_REPLY

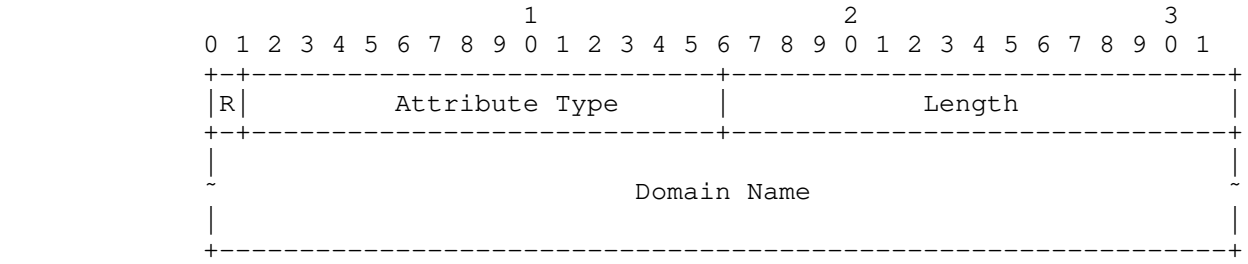
In this example, the initiator has no existing DNSSEC trust anchors would the requested domain. the "example.com" dommain has DNSSEC trust anchors that are returned, while the "other.com" domain has no DNSSEC trust anchors

```
CP(CFG_REQUEST) =
  INTERNAL_IP4_ADDRESS()
  INTERNAL_IP4_DNS()
  INTERNAL_DNS_DOMAIN(example.com)
  INTERNAL_DNS_TA()
  INTERNAL_DNS_DOMAIN(other.com)
  INTERNAL_DNS_TA()

CP(CFG_REPLY) =
  INTERNAL_IP4_ADDRESS(198.51.100.234)
  INTERNAL_IP4_DNS(198.51.100.2)
  INTERNAL_IP4_DNS(198.51.100.4)
  INTERNAL_DNS_DOMAIN(example.com)
  INTERNAL_DNS_TA(43547,8,1,B6225AB2CC613E0DCA7962BDC2342EA4F1B56083)
  INTERNAL_DNS_TA(31406,8,2,F78CF3344F72137235098ECBBD08947C2C90....)
  INTERNAL_DNS_DOMAIN(city.other.com)
```

4. Payload Formats

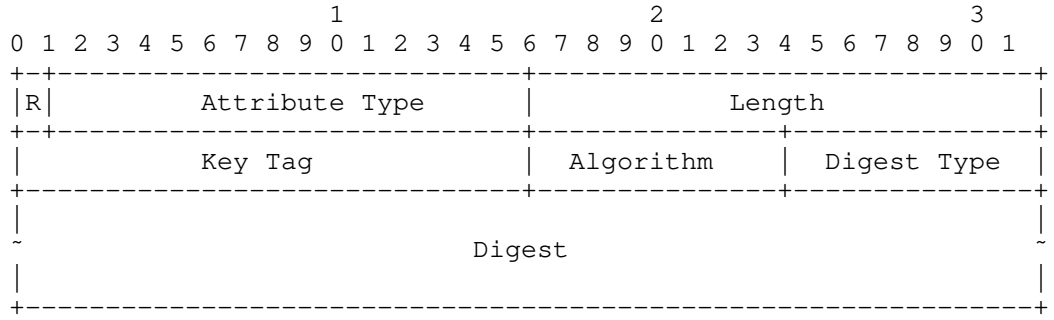
4.1. INTERNAL_DNS_DOMAIN Configuration Attribute Type



- Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- Attribute Type (15 bits) [TBD IANA] - INTERNAL_DNS_DOMAIN.
- Length (2 octets, unsigned integer) - Length of domain name.
- Domain Name (0 or more octets) - A domain or subdomain used for Split DNS rules, such as example.com. This is a string of ASCII

characters with labels separated by dots, with no trailing dot, using IDNA [RFC5890] for non-ASCII DNS domains. The value is NOT null-terminated.

4.2. INTERNAL_DNSSEC_TA Configuration Attribute



- Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- Attribute Type (15 bits) [TBD IANA] - INTERNAL_DNSSEC_TA.
- Length (2 octets, unsigned integer) - Length of DNSSEC Trust Anchor data.
- Key Tag value (0 or 2 octets, unsigned integer) - Key Tag as specified in [RFC4034] Section 5.1
- DNSKEY algorithm (0 or 1 octet) - Value from the IANA DNS Security Algorithm Numbers Registry
- DS algorithm (0 or 1 octet) - Value from the IANA Delegation Signer (DS) Resource Record (RR) Type Digest Algorithms Registry
- Digest (0 or more octets) - The raw digest as specified in [RFC4034] Section 5.1

5. Split DNS Usage Guidelines

If a CFG_REPLY payload contains no INTERNAL_DNS_DOMAIN attributes, the client MAY use the provided INTERNAL_IP4_DNS or INTERNAL_IP6_DNS servers as the default DNS server(s) for all queries.

For each INTERNAL_DNS_DOMAIN entry in a CFG_REPLY payload, the client SHOULD use the provided INTERNAL_IP4_DNS or INTERNAL_IP6_DNS DNS servers as the only resolvers for the listed domains and its sub-domains and it SHOULD NOT attempt to resolve the provided DNS domains using its external DNS servers.

If the initiator host is configured to block DNS answers containing IP addresses from special IP address ranges such as those of [RFC1918], the initiator SHOULD allow the DNS domains listed in the INTERNAL_DNS_DOMAIN attributes to contain these IP addresses.

If a CFG_REPLY contains one or more INTERNAL_DNS_DOMAIN attributes, the client SHOULD configure its DNS resolver to resolve those domains and all their subdomains using only the DNS resolver(s) listed in that CFG_REPLY message. If those resolvers fail, those names SHOULD NOT be resolved using any other DNS resolvers. All other domain names MUST be resolved using some other external DNS resolver(s), configured independently, and SHOULD NOT be sent to the internal DNS resolver(s) listed in that CFG_REPLY message. For example, if the INTERNAL_DNS_DOMAIN attribute specifies "example.com", then "example.com", "www.example.com" and "mail.eng.example.com" MUST be resolved using the internal DNS resolver(s), but "anotherexample.com" and "ample.com" MUST be resolved using the system's external DNS resolver(s).

An initiator SHOULD ignore INTERNAL_DNS_DOMAIN attributes containing domains that are designated Special Use Domain Names in [RFC6761], such as "local", "localhost", "invalid", etc. Although it may explicitly wish to support some Special Use Domain Names.

When an IPsec connection is terminated, the DNS forwarding must be unconfigured. The DNS forwarding itself MUST be be deleted. All cached data of the INTERNAL_DNS_DOMAIN provided DNS domainis MUST be flushed. This includes negative cache entries. Obtained DNSSEC trust anchors MUST be removed from the list of trust anchors. The outstanding DNS request queue MAY be cleared.

A domain that is served via INTERNAL_DNS_DOMAIN MUST NOT have indirect references to DNS records that point to other Split DNS domains that are not served via INTERNAL_DNS_DOMAIN attributes. Indirect reference RRtypes include CNAME, DNAME, MX and SRV RR's.

INTERNAL_DNS_DOMAIN and INTERNAL_DNSSEC_TA attributes SHOULD only be used on split tunnel configurations where only a subset of traffic is routed into a private remote network using the IPsec connection. If all traffic is routed over the IPsec connection, the existing global INTERNAL_IP4_DNS and INTERNAL_IP6_DNS can be used without creating specific DNS exemptions.

6. Security Considerations

The use of Split DNS configurations assigned by an IKEv2 responder is predicated on the trust established during IKE SA authentication. However, if IKEv2 is being negotiated with an anonymous or unknown

endpoint (such as for Opportunistic Security [RFC7435]), the initiator MUST ignore Split DNS configurations assigned by the responder.

If a host connected to an authenticated IKE peer is connecting to another IKE peer that attempts to claim the same domain via the INTERNAL_DNS_DOMAIN attribute, the IKE connection should be terminated.

If the IP address value of the received INTERNAL_IP4_DNS or INTERNAL_IP6_DNS attribute is not covered by the proposed IPsec connection, then the local DNS should not be reconfigured until a CREATE_CHILD Exchange is received that covers these IP addresses.

INTERNAL_DNSSEC_TA directives MUST immediately follow an INTERNAL_DNS_DOMAIN directive. As the INTERNAL_DNSSEC_TA format itself does not contain the domain name, it relies on the preceding INTERNAL_DNS_DOMAIN to provide the domain for which it specifies the trust anchor.

If the initiator is using DNSSEC validation for a domain in its public DNS view, and it requests and receives an INTERNAL_DNS_DOMAIN attribute without an INTERNAL_DNSSEC_TA, it will need to reconfigure its DNS resolver to allow for an insecure delegation. It SHOULD NOT accept insecure delegations for domains that are DNSSEC signed in the public DNS view, for which it has not explicitely requested such deletation by specifying the domain specifically using a INTERNAL_DNS_DOMAIN(domain) request.

7. IANA Considerations

This document defines two new IKEv2 Configuration Payload Attribute Types, which are allocated from the "IKEv2 Configuration Payload Attribute Types" namespace.

Value	Attribute Type	Multi-Valued	Length	Reference
[TBD]	INTERNAL_DNS_DOMAIN	YES	0 or more	[this document]
[TBD]	INTERNAL_DNSSEC_TA	YES	0 or more	[this document]

Figure 1

8. References

8.1. Normative References

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

[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>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <http://www.rfc-editor.org/info/rfc4034>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <http://www.rfc-editor.org/info/rfc5890>.

[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. Informative References

[RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, DOI 10.17487/RFC2775, February 2000, <http://www.rfc-editor.org/info/rfc2775>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <http://www.rfc-editor.org/info/rfc6761>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://www.rfc-editor.org/info/rfc7435>.

Authors' Addresses

Tommy Pauly
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
US

Email: tpauly@apple.com

Paul Wouters
Red Hat

Email: pwouters@redhat.com