

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: October 21, 2017

S. Fluhrer  
D. McGrew  
P. Kampanakis  
Cisco Systems  
April 19, 2017

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on October 21, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

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

## Table of Contents

1. Introduction . . . . .	2
1.1. Changes . . . . .	3
1.2. Requirements Language . . . . .	4
2. Assumptions . . . . .	4
3. Exchanges . . . . .	4
4. PPK ID format . . . . .	7
5. PPK Distribution . . . . .	8
6. Upgrade procedure . . . . .	8
7. Security Considerations . . . . .	8
8. References . . . . .	9
8.1. Normative References . . . . .	9
8.2. Informational References . . . . .	10
Appendix A. Discussion and Rationale . . . . .	10
Appendix B. Acknowledgement . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

It is an open question whether or not it is feasible to build a quantum computer (and if so, when might one be implemented), but if it is, many of the cryptographic algorithms and protocols currently in use would be insecure. A quantum computer would be able to solve DH and ECDH problems, and this would imply that the security of existing IKEv2 systems would be compromised. IKEv1 when used with strong preshared keys is not vulnerable to quantum attacks, because those keys are one of the inputs to the key derivation function. If the preshared key has sufficient entropy and the PRF, encryption and authentication transforms are postquantum secure, then the resulting system is believed to be quantum resistant, that is, believed to be invulnerable to an attacker with a Quantum Computer.

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

The general idea is that we add an additional secret that is shared between the initiator and the responder; this secret is in addition

to the authentication method that is already provided within IKEv2. We stir in this secret into the SK\_d value, which is used to generate the key material (KEYMAT) keys and the SKEYSEED for the child SAs; this secret provides quantum resistance to the IPsec SAs (and any child IKE SAs). We also stir in the secret into the SK\_pi, SK\_pr values; this allows both sides to detect a secret mismatch cleanly.

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

### 1.1. Changes

Changes in this draft from the previous versions

draft-03

- Modified how we stir the PPK into the IKEv2 secret state
- Modified how the use of PPKs is negotiated

draft-02

- Simplified the protocol by stirring in the preshared key into the child SAs; this avoids the problem of having the responder decide which preshared key to use (as it knows the initiator identity at that point); it does mean that someone with a Quantum Computer can recover the initial IKE negotiation.
- Removed positive endorsements of various algorithms. Retained warnings about algorithms known to be weak against a Quantum Computer

draft-01

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

draft-00

- We switched from using vendor ID's to transmit the additional data to notifications
- We added a mandatory cookie exchange to allow the server to communicate to the client before the initial exchange

- We added algorithm agility by having the server tell the client what algorithm to use in the cookie exchange
- We have the server specify the PPK Indicator Input, which allows the server to make a trade-off between the efficiency for the search of the clients PPK, and the anonymity of the client.
- We now use the negotiated PRF (rather than a fixed HMAC-SHA256) to transform the nonces during the KDF

## 1.2. Requirements Language

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

## 2. Assumptions

We assume that each IKE peer has a list of Postquantum Preshared Keys (PPK) along with their identifiers (PPK\_id), and any potential IKE initiator has a selection of which PPK to use with with any specific responder. In addition, the implementation has a configurable flag that determines whether this postquantum preshared key is mandatory. This PPK is independent of the preshared key (if any) that the IKEv2 protocol uses to perform authentication.

## 3. Exchanges

If the initiator is configured to use a postquantum preshared key with the responder (whether or not the use of the PPK is optional), then it will include a notify payload in the initial exchange as follows:

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

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

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

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

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

```

Initiator                               Responder
-----
<--- HDR, SAR1, KEr, Nr, [CERTREQ], N(PPK_SUPPORT)

```

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

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

```

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

```

That is, we use the standard IKE key derivation process except that the three subkeys SK\_d, SK\_pi, SK\_pr are run through the prf again, this time using the PPK as the key.

The initiator then sends the initial encrypted message, including the PPK\_id value as follows:

```

Initiator                               Responder
-----
HDR, SK {IDi, [CERT,] [CERTREQ,]
      [IDr,] AUTH, Sai2,
      TSi, TSr, N(PPK_IDENTITY) (PPK_id)} --->

```

N(PPK\_IDENTITY) is a status notification payload with the type [TBA]; it has a protocol ID of 0, and no SPI and has a notification data that consists of the identifier PPK\_id.

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

```

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

```

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

```

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

```

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

This table summarizes the above logic by the responder

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

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

If the initial exchange had PPK\_SUPPORT sent by both the initiator and the responder, and the initiator does not include a PPK\_NOTIFY notification, then the responder SHOULD fail the exchange.

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

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

#### 4. PPK ID format

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

PPK ID Type	Value
<code>PPK_ID_OPAQUE</code>	0
<code>PPK_ID_FIXED</code>	1
RESERVED TO IANA	2-127
Reserved for private use	128-255

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

For `PPK_ID_FIXED`, the format of the PPK ID and the PPK are fixed octet strings; the remaining bytes of the `PPK_ID` are a configured value. We assume that there is a fixed mapping between `PPK_ID` and PPK, which is configured locally to both the initiator and the responder. The responder can use to do a look up the passed `PPK_id` value to determine the corresponding PPK value. Not all implementations are able to configure arbitrary octet strings; to improve the potential interoperability, it is recommended that, in the `PPK_ID_FIXED` case, both the PPK and the `PPK_ID` strings be limited to the base64 character set, namely the 64 characters 0-9, A-Z, a-z, + and /.

The PPK ID type values 2-127 are reserved for IANA; values 128-255 are for private use among mutually consenting parties.

## 5. PPK Distribution

PPK\_id's of the type PPK\_ID\_FIXED (and the corresponding PPKs) are assumed to be configured within the IKE device in an out-of-band fashion. While the method of distribution is a local matter, one suggestion would be to reuse the format within [RFC6030], with the Key Id field being the PPK\_ID (without the 0x01 prefix for a PPK\_ID\_FIXED), and with the PPK being the secret, and the algorithm as PIN ("Algorithm=urn:ietf:params:xml:ns:keyprov:pskc:pin").

## 6. Upgrade procedure

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

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

- The set of PPKs (and corresponding PPK\_id's) that this node would need to know
- For each peer that this node would initiate to, which PPK that we would use
- That the use of PPK is currently optional

With this configuration, the node will continue to operate with nodes that have not yet been upgraded. This is due to the PPK\_SUPPORT notify; if the initiator has not been upgraded, it will not send the PPK\_SUPPORT notify (and so the responder will know that we will not use a PPK); if the responder has not been upgraded, it will not send the PPK\_SUPPORT notify (and so the initiator will know not to use a PPK). And, if both peers have been upgraded, they will both realize it, and in that case, the link will be quantum secure

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

## 7. Security Considerations

Quantum computers are able to perform Grover's algorithm; that effectively halves the size of a symmetric key. Because of this, the user SHOULD ensure that the postquantum preshared key used has at



least 256 bits of entropy, in order to provide a 128 bit security level.

Although this protocol preserves all the security properties of IKE against adversaries with conventional computers, this protocol allows an adversary with a Quantum Computer to decrypt all traffic encrypted with the initial IKE SA. In particular, it allows the adversary to recover the identities of both sides. If there is IKE traffic other than the identities that need to be protected against such an adversary, one suggestion would be to form an initial IKE SA (which is used to exchange identities), perhaps by using the protocol documented in RFC6023. Then, you would immediately create a child IKE SA (which is used to exchange everything else). Because the child IKE SA keys are a function of SK\_d, which is a function of the PPK (among other things), traffic protected by that SA is secure against Quantum capable adversaries.

In addition, the policy SHOULD be set to negotiate only quantum-resistant symmetric algorithms; while this RFC doesn't claim to give advice as to what algorithms are secure (as that may change based on future cryptographical results), here is a list of defined IKEv2 and IPsec algorithms that should NOT be used, as they are known not to be Quantum Resistant

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

Any ESP Transform with key size <256 bits

PRF\_AES128\_XCBC and PRF\_AES128\_CBC; even though they are defined to be able to use an arbitrary key size, they convert it into a 128 bit key internally

## 8. References

### 8.1. Normative References

- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<http://www.rfc-editor.org/info/rfc2104>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.

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

## 8.2. Informational References

- [RFC6023] Nir, Y., Tschofenig, H., Deng, H., and R. Singh, "A Childless Initiation of the Internet Key Exchange Version 2 (IKEv2) Security Association (SA)", RFC 6023, DOI 10.17487/RFC6023, October 2010, <<http://www.rfc-editor.org/info/rfc6023>>.
- [RFC6030] Hoyer, P., Pei, M., and S. Machani, "Portable Symmetric Key Container (PSKC)", RFC 6030, DOI 10.17487/RFC6030, October 2010, <<http://www.rfc-editor.org/info/rfc6030>>.
- [SPDP] McGrew, D., "A Secure Peer Discovery Protocol (SPDP)", 2001, <<http://www.mindspring.com/~dmcgrew/spdp.txt>>.

## Appendix A. Discussion and Rationale

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

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

A third goal was to be friendly to incremental deployment in operational networks, for which we might not want to have a global shared key, and also if we're rolling this out incrementally. This

is why we specifically try to allow the PPK to be dependent on the peer, and why we allow the PPK to be configured as optional.

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

#### Appendix B. Acknowledgement

We would like to thank Tero Kivine, Valery Smyslov, Paul Wouters and the rest of the ipsecme working group for their feedback and suggestions for the scheme

#### Authors' Addresses

Scott Fluhrer  
Cisco Systems

Email: [sfluhrer@cisco.com](mailto:sfluhrer@cisco.com)

David McGrew  
Cisco Systems

Email: [mcgrew@cisco.com](mailto:mcgrew@cisco.com)

Panos Kampanakis  
Cisco Systems

Email: [pkampana@cisco.com](mailto:pkampana@cisco.com)

IPSecME Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 29, 2018

Y. Nir  
Dell EMC  
October 26, 2017

Using Edwards-curve Digital Signature Algorithm (EdDSA) in the Internet  
Key Exchange (IKEv2)  
draft-ietf-ipsecme-eddsa-04

Abstract

This document describes the use of the Edwards-curve digital signature algorithm in the 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 <https://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on April 29, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Conventions Used in This Document . . . . .	3
2. The "Identity" Hash Identifier . . . . .	3
3. Security Considerations . . . . .	3
4. IANA Considerations . . . . .	3
5. Normative References . . . . .	3
Appendix A. ASN.1 Objects . . . . .	5
A.1. ASN.1 Object for Ed25519 . . . . .	5
A.2. ASN.1 Object for Ed448 . . . . .	5
Author's Address . . . . .	5

## 1. Introduction

The Internet Key Exchange protocol [RFC7296] can use arbitrary signature algorithms as described in [RFC7427]. The latter RFC defines the SIGNATURE\_HASH\_ALGORITHMS notification where each side of the IKE negotiation lists its supported hash algorithms. This assumes that all signature schemes involve a hashing phase followed by a signature phase. This made sense because most signature algorithms either cannot sign messages bigger than their key or truncate messages bigger than their key.

EdDSA ([RFC8032]) defines signature methods that do not require pre-hashing of the message. Unlike other methods, these accept arbitrary-sized messages, so no pre-hashing is required. These methods are called Ed25519 and Ed448, which respectively use the Edwards 25519 and the Edwards 448 ("Goldilocks") curves. Although that document also defines pre-hashed versions of these algorithm, those versions are not recommended for protocols where the entire to-be-signed message is available at once. See section 8.5 or RFC 8032 for that recommendation.

EdDSA defines the binary format of the signatures that should be used in the "Signature Value" field of the Authentication Data Format in section 3. The CURDLE PKIX document ([I.D-curdle-pkix]) defines the object identifiers (OIDs) for these signature methods. For convenience, these OIDs are repeated in Appendix A.

In order to signal within IKE that no hashing needs to be done, we define a new value in the SIGNATURE\_HASH\_ALGORITHMS notification, one that indicates that no hashing is performed.

### 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. The "Identity" Hash Identifier

This document defines a new value called "Identity" (value is 5) in the hash algorithm registry for use in the SIGNATURE\_HASH\_ALGORITHMS notification. Inserting this new value into the notification indicates that the receiver supports at least one signature algorithm that accepts arbitrary-sized messages such as Ed25519 and Ed448.

Ed25519 and Ed448 are only defined with the Identity hash, and MUST NOT be sent to a receiver that has not indicated support for the "Identity" hash.

The pre-hashed versions of Ed25519 and Ed448 (Ed25519ph and Ed448ph respectively) MUST NOT be used in IKE.

## 3. Security Considerations

The new "Identity" value is needed only for signature algorithms that accept an arbitrary-sized input. It MUST NOT be used if none of the supported and configured algorithms have this property. On the other hand there is no good reason to pre-hash the inputs where the signature algorithm has that property. For this reason implementations MUST have the "Identity" value in the SIGNATURE\_HASH\_ALGORITHMS notification when EdDSA is supported and configured. Implementations SHOULD NOT have other hash algorithms in the notification if all supported and configured signature algorithms have this property.

## 4. IANA Considerations

IANA has assigned the value 5 for the algorithm with the name "Identity" in the "IKEv2 Hash Algorithms" registry with this draft as reference.

Upon publication of this document IANA is requested to update the entry with this document as reference.

## 5. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)", RFC 7427, DOI 10.17487/RFC7427, January 2015, <<https://www.rfc-editor.org/info/rfc7427>>.
- [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/info/rfc8032>>.
- [I.D-curdle-pkix] Josefsson, S. and J. Schaad, "Algorithm Identifiers for Ed25519, Ed25519ph, Ed448, Ed448ph, X25519 and X448 for use in the Internet X.509 Public Key Infrastructure", September 2017, <<https://tools.ietf.org/html/draft-ietf-curdle-pkix-06>>.

## Appendix A. ASN.1 Objects

The normative reference for the ASN.1 objects for Ed25519 and Ed448 is in [I.D-curdle-pkix]. They are repeated below for convenience.

## A.1. ASN.1 Object for Ed25519

id-Ed25519 OBJECT IDENTIFIER ::= { 1.3.101.112 }

Parameters are absent. Length is 7 bytes.

Binary encoding: 3005 0603 2B65 70

## A.2. ASN.1 Object for Ed448

id-Ed448 OBJECT IDENTIFIER ::= { 1.3.101.113 }

Parameters are absent. Length is 7 bytes.

Binary encoding: 3005 0603 2B65 71

## Author's Address

Yoav Nir  
Dell EMC  
9 Andrei Sakharov St  
Haifa 3190500  
Israel

EMail: ynir.ietf@gmail.com



Network Working Group  
Internet-Draft  
Obsoletes: 4307 (if approved)  
Updates: 7296 (if approved)  
Intended status: Standards Track  
Expires: September 30, 2017

Y. Nir  
Check Point  
T. Kivinen  
INSIDE Secure  
P. Wouters  
Red Hat  
D. Migault  
Ericsson  
March 29, 2017

Algorithm Implementation Requirements and Usage Guidance for IKEv2  
draft-ietf-ipsecme-rfc4307bis-18

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 updates RFC 7296 and obsoletes RFC 4307 in defining the current algorithm implementation requirements and usage guidance for IKEv2, and does minor cleaning up of the IKEv2 IANA registry. This document does not update the algorithms used for packet encryption using IPsec Encapsulated Security Payload (ESP).

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 September 30, 2017.

## Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Conventions Used in This Document . . . . .	3
1.2. Updating Algorithm Implementation Requirements and Usage Guidance . . . . .	3
1.3. Updating Algorithm Requirement Levels . . . . .	4
1.4. Document Audience . . . . .	5
2. Algorithm Selection . . . . .	5
2.1. Type 1 - IKEv2 Encryption Algorithm Transforms . . . . .	5
2.2. Type 2 - IKEv2 Pseudo-random Function Transforms . . . . .	7
2.3. Type 3 - IKEv2 Integrity Algorithm Transforms . . . . .	8
2.4. Type 4 - IKEv2 Diffie-Hellman Group Transforms . . . . .	9
2.5. Summary of Changes from RFC 4307 . . . . .	10
3. IKEv2 Authentication . . . . .	11
3.1. IKEv2 Authentication Method . . . . .	11
3.1.1. Recommendations for RSA key length . . . . .	12
3.2. Digital Signature Recommendations . . . . .	12
4. Algorithms for Internet of Things . . . . .	13
5. Security Considerations . . . . .	14
6. IANA Considerations . . . . .	15
7. Acknowledgements . . . . .	15
8. References . . . . .	16
8.1. Normative References . . . . .	16
8.2. Informative References . . . . .	16
Authors' Addresses . . . . .	17

## 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. It changes the mandatory to implement authentication algorithms of Section 4 of [RFC7296] by saying RSA key lengths of less than 2048 SHOULD NOT be used. It does not describe the cryptographic parameters of the AH or ESP protocols.

### 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].

When used in the tables in this document, these terms indicate that the listed algorithm MUST, MUST NOT, SHOULD, SHOULD NOT or MAY be implemented as part of an IKEv2 implementation. Additional terms used in this document are:

- 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, it is expected 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.

### 1.2. 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 than 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 this separate document.

### 1.3. 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.

#### 1.4. Document Audience

The recommendations of this document mostly target IKEv2 implementers who need to create implementations that 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 regarding 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. Algorithm Selection

### 2.1. 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 one of the integrity algorithms in Section 2.3.

Name	Status	AEAD?	Comment
ENCR_AES_CBC	MUST	No	(1)
ENCR_CHACHA20_POLY1305	SHOULD	Yes	
ENCR_AES_GCM_16	SHOULD	Yes	(1)
ENCR_AES_CCM_8	SHOULD	Yes	(IoT)
ENCR_3DES	MAY	No	
ENCR_DES	MUST NOT	No	

(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 SHOULD level. 192-bit and 256-bit remain at the MAY level.

ENCR\_AES\_CBC is raised from SHOULD+ for 128-bit keys and MAY for 256-bit keys in [RFC4307] to MUST. 192-bit keys remain at the MAY level. ENCR\_AES\_CBC 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 Crypto Forum Research Group (CFRG) of the IRTF 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.

ENCR\_AES\_GCM\_16 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 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 in those cases, and IoT devices want to make packets as small as possible. The SHOULD level is for 128-bit keys, 256-bit keys remains at MAY level.

ENCR\_3DES has been downgraded from RFC4307 MUST- to MAY. All IKEv2 implementations 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 off-the-shelf hardware. It provides no meaningful security whatsoever and therefore MUST NOT be implemented.

## 2.2. Type 2 - IKEv2 Pseudo-random Function Transforms

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

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

As no SHA2 based transforms were referenced in RFC4307, PRF\_HMAC\_SHA2\_256 was not mentioned in RFC4307. 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 cryptographic attacks against SHA1 are increasing, resulting in 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. Cryptographic attacks against MD5, such as collision attacks mentioned in [TRANSCRIPTION], are resulting in an industry-wide trend to deprecate and remove MD5 (and thus HMAC-MD5) from cryptographic libraries.

### 2.3. 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 2.1) 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 cryptographic attacks against SHA1 are increasing, resulting in an industry-wide trend to deprecate its usage

AUTH\_AES\_XCBC\_96 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 statuses were 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].

#### 2.4. 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 [RFC7296] 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 one of the private numbers ( $a$  or  $b$ ) and the complementary public value ( $g^{**b}$  or  $g^{**a}$ ), 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	MUST 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 to MUST 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 was not defined at that time. Group 19 is widely implemented and considered secure and therefore has been promoted to the SHOULD level.

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 being 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 off-the-shelves hardware. It provides no security whatsoever. It has therefore been downgraded to MUST NOT.

Group 22, 23 and 24 are MODP Groups with Prime Order Subgroups that are 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 has been shown that Group 22 with 1024-bit MODP is too weak and academia have the resources to generate malicious values at this size. This has resulted in Group 22 to be demoted to MUST NOT. Group 23 and 24 have been demoted to SHOULD NOT and are expected to be further downgraded in the near future to MUST NOT. Since Group 23 and 24 have small subgroups, the checks specified in "Additional Diffie-Hellman Test for the IKEv2" [RFC6989] section 2.2 first bullet point MUST be done when these groups are used.

## 2.5. Summary of Changes from RFC 4307

The following table summarizes the changes from RFC 4307.

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

Algorithm	RFC 4307	RFC XXXX
ENCR_3DES	MUST-	MAY
ENCR_NULL	MUST NOT[errata]	MUST NOT
ENCR_AES_CBC	SHOULD+	MUST
ENCR_AES_CTR	SHOULD	(*)
PRF_HMAC_MD5	MAY	MUST NOT
PRF_HMAC_SHA1	MUST	MUST-
PRF_AES128_XCBC	SHOULD+	SHOULD
AUTH_HMAC_MD5_96	MAY	MUST NOT
AUTH_HMAC_SHA1_96	MUST	MUST-
AUTH_AES_XCBC_96	SHOULD+	SHOULD
Group 2 (1024-bit)	MUST-	SHOULD NOT
Group 14 (2048-bit)	SHOULD+	MUST

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

### 3. 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 with signature verification and generation for the authentication.

#### 3.1. 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].

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

ECDSA based Authentication Methods are also expected to be downgraded as these do not provide hash function agility. Instead, ECDSA (like RSA) is expected to be performed using the generic Digital Signature method. It's status is SHOULD.

DSS Digital Signature is bound to SHA-1 and has the same level of security as 1024-bit RSA. It is currently at SHOULD NOT and 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. Its current status is SHOULD.

### 3.1.1. 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 key sizes less than 2048 are updated to SHOULD NOT as there is industry-wide trend to deprecate key lengths less than 2048 bits. Since these signatures only have value in real-time, and need no future protection, smaller keys were kept at SHOULD NOT instead of MUST NOT.

### 3.2. Digital Signature Recommendations

When a Digital Signature authentication method is implemented, 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 the Digital Signature authentication method is used with RSA signature algorithm, 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	(*)

(\*) Empty or Default parameters means it is using SHA1, which is at level MUST NOT.

#### 4. 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.

## 5. 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.

## 6. IANA Considerations

This document renames some of the names in the "Transform Type 1 - Encryption Algorithm Transform IDs" registry of the "Internet Key Exchange Version 2 (IKEv2) Parameters". All the other names have ENCR\_ prefix except 3, and all other entries use names in format of uppercase words separated with underscores except 6. This document changes those names to match others.

This document requests IANA to rename following entries for the AES-GCM cipher [RFC4106] and the Camellia cipher [RFC5529]:

Old name	New name
AES-GCM with a 8 octet ICV	ENCR_AES_GCM_8
AES-GCM with a 12 octet ICV	ENCR_AES_GCM_12
AES-GCM with a 16 octet ICV	ENCR_AES_GCM_16
ENCR_CAMELLIA_CCM with an 8-octet ICV	ENCR_CAMELLIA_CCM_8
ENCR_CAMELLIA_CCM with a 12-octet ICV	ENCR_CAMELLIA_CCM_12
ENCR_CAMELLIA_CCM with a 16-octet ICV	ENCR_CAMELLIA_CCM_16

In addition to add this RFC as reference to both ESP Reference and IKEv2 Reference columns for ENCR\_AES\_GCM entries, keeping the current references there also, and also add this RFC as reference to the ESP Reference column for ENCR\_CAMELLIA\_CCM entries, keeping the current reference there also.

The final registry entries should be:

Number	Name	ESP Reference	IKEv2 Reference
...			
18	ENCR_AES_GCM_8	[RFC4106] [RFCXXXX]	[RFC5282] [RFCXXXX]
19	ENCR_AES_GCM_12	[RFC4106] [RFCXXXX]	[RFC5282] [RFCXXXX]
20	ENCR_AES_GCM_16	[RFC4106] [RFCXXXX]	[RFC5282] [RFCXXXX]
...			
25	ENCR_CAMELLIA_CCM_8	[RFC5529] [RFCXXXX]	-
26	ENCR_CAMELLIA_CCM_12	[RFC5529] [RFCXXXX]	-
27	ENCR_CAMELLIA_CCM_16	[RFC5529] [RFCXXXX]	-

## 7. 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, Tommy Pauly, Eric Rescorla and Pete Resnick for their valuable feedback and reviews.

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<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>>.

### 8.2. 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>>.
- [RFC5529] Kato, A., Kanda, M., and S. Kanno, "Modes of Operation for Camellia for Use with IPsec", RFC 5529, DOI 10.17487/RFC5529, April 2009, <<http://www.rfc-editor.org/info/rfc5529>>.
- [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

EEmail: [ynir.ietf@gmail.com](mailto:ynir.ietf@gmail.com)

Tero Kivinen  
INSIDE Secure  
Eerikinkatu 28  
HELSINKI FI-00180  
FI

EEmail: [kivinen@iki.fi](mailto: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  
Obsoletes: 7321 (if approved)  
Intended status: Standards Track  
Expires: December 21, 2017

P. Wouters  
Red Hat  
D. Migault  
J. Mattsson  
Ericsson  
Y. Nir  
Check Point  
T. Kivinen  
INSIDE Secure  
June 19, 2017

Cryptographic Algorithm Implementation Requirements and Usage Guidance  
for Encapsulating Security Payload (ESP) and Authentication Header (AH)  
draft-ietf-ipsecme-rfc7321bis-06

#### 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.

#### 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 21, 2017.

#### Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Updating Algorithm Implementation Requirements and Usage Guidance . . . . .	3
1.2. Updating Algorithm Requirement Levels . . . . .	3
1.3. Document Audience . . . . .	4
2. Requirements Language . . . . .	4
3. Manual Keying . . . . .	5
4. Encryption must be Authenticated . . . . .	5
5. ESP Encryption Algorithms . . . . .	6
6. ESP and AH Authentication Algorithms . . . . .	8
7. ESP and AH Compression Algorithms . . . . .	9
8. Summary of Changes from RFC 7321 . . . . .	10
9. Acknowledgements . . . . .	10
10. IANA Considerations . . . . .	10
11. Security Considerations . . . . .	10
12. References . . . . .	11
12.1. Normative References . . . . .	11
12.2. Informative References . . . . .	11
Authors' Addresses . . . . .	13

## 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 than 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 and 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. 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].

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. Manual Keying

Manual Keying SHOULD NOT be used as it is inherently dangerous. Without any secure keying protocol such as IKE, IPsec does not offer Perfect Forward Secrecy ("PFS") protection and there is no entity to ensure refreshing of session keys, tracking SPI uniqueness and ensuring nonces, IVs and counters are never re-used. This document was written for deploying ESP/AH using IKE ([RFC7296]) and assumes that keying happens using IKE version 2 or higher.

If Manual Keying is used regardless, Counter Mode algorithms such as ENCR\_AES\_CTR, ENCR\_AES\_CCM, ENCR\_AES\_GCM and ENCR\_CHACHA20\_POLY1305 MUST NOT be used as it is incompatible with a secure and persistent handling of the counter, as explained in the Security Considerations Section of [RFC3686]. This particularly applies to IoT devices that have no state across reboots. As of publication date of this document, ENCR\_AES\_CBC is the only Mandatory-To-Implement encryption algorithm suitable for Manual Keying.

### 4. Encryption must be Authenticated

Encryption without authentication is not effective and MUST NOT be used. IPsec offers three ways to provide both encryption and authentication:

- o ESP with an AEAD cipher
- o ESP with a non-AEAD cipher + authentication
- o ESP with a non-AEAD cipher + AH with authentication

The fastest and most modern method is to use ESP with a combined mode cipher such as an AEAD cipher that handles encryption/decryption and authentication in a single step. In this case, the AEAD cipher is set as the encryption algorithm and the authentication algorithm is set to none. Examples of this are ENCR\_AES\_GCM\_16 and ENCR\_CHACHA20\_POLY1305.

A more traditional approach is to use ESP with an encryption and an authentication algorithm. This approach is slower, as the data has to be processed twice, once for encryption/decryption and once for authentication. An example of this is ENCR\_AES\_CBC combined with AUTH\_HMAC\_SHA2\_512\_256.

The last method that can be used is ESP+AH. This method is NOT RECOMMENDED. It is the slowest method and also takes up more octets due to the double header of ESP+AH, resulting in a smaller effective MTU for the encrypted data. With this method, ESP is only used for confidentiality without an authentication algorithm and a second IPsec protocol of type AH is used for authentication. An example of this is ESP with ENCR\_AES\_CBC with AH with AUTH\_HMAC\_SHA2\_512\_256.

## 5. 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 (IoT)	Yes	[RFC4309]
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 the given level.

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



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 these 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 standardized for ESP and therefore 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 at MUST. ENCR\_AES\_CBC MUST be implemented in order to enable interoperability between implementations 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 AES-CBC and 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.

## 6. ESP and AH Authentication Algorithms

Authentication algorithm recommendations in this section are targeting two types of communications:

- o Authenticated only communications without encryption, such as ESP with NULL encryption or AH communications.
- o Communications that are encrypted with non-AEAD algorithm which 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 case where AUTH\_NONE is acceptable is when authenticated encryption algorithms are selected from Section 5. In all other cases, AUTH\_NONE MUST NOT be selected. As ESP and AH both provide authentication, one may be tempted to combine these protocols 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, AUTH\_NONE 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 set as SHOULD only 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 and not for ESP. If using ENCR\_NULL, AUTH\_HMAC\_SHA2\_256\_128 is recommended for integrity. If using AES-GMAC in ESP 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.

## 7. 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.

## 8. 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.

## 9. 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.

## 10. IANA Considerations

This document has no IANA actions.

## 11. 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.

## 12. References

### 12.1. 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>>.

### 12.2. 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>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.

#### Authors' Addresses

Paul Wouters  
Red Hat

Email: [pwouters@redhat.com](mailto:pwouters@redhat.com)

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

Phone: +1 514-452-2160  
Email: [daniel.migault@ericsson.com](mailto:daniel.migault@ericsson.com)

John Mattsson  
Ericsson AB  
SE-164 80 Stockholm  
Sweden

Email: [john.mattsson@ericsson.com](mailto:john.mattsson@ericsson.com)

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

Email: [ynir.ietf@gmail.com](mailto:ynir.ietf@gmail.com)

Tero Kivinen  
INSIDE Secure  
Eerikinkatu 28  
HELSINKI FI-00180  
FI

Email: [kivinen@iki.fi](mailto:kivinen@iki.fi)



Network  
Internet-Draft  
Intended status: Standards Track  
Expires: September 12, 2019

T. Pauly  
Apple Inc.  
P. Wouters  
Red Hat  
March 11, 2019

Split DNS Configuration for IKEv2  
draft-ietf-ipsecme-split-dns-17

Abstract

This document defines two Configuration Payload Attribute Types (INTERNAL\_DNS\_DOMAIN and INTERNAL\_DNSSEC\_TA) for the Internet Key Exchange Protocol Version 2 (IKEv2). These payloads add support for private (internal-only) DNS domains. These domains are intended to be resolved using non-public DNS servers that are only reachable through the IPsec connection. DNS resolution for other domains remains unchanged. These Configuration Payloads only apply to split tunnel configurations.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 12, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Requirements Language . . . . .	3
2. Applicability . . . . .	3
3. Protocol Exchange . . . . .	5
3.1. Configuration Request . . . . .	5
3.2. Configuration Reply . . . . .	6
3.3. Mapping DNS Servers to Domains . . . . .	6
3.4. Example Exchanges . . . . .	6
3.4.1. Simple Case . . . . .	6
3.4.2. Requesting Domains and DNSSEC trust anchors . . . . .	7
4. Payload Formats . . . . .	8
4.1. INTERNAL_DNS_DOMAIN Configuration Attribute Type Request and Reply . . . . .	8
4.2. INTERNAL_DNSSEC_TA Configuration Attribute . . . . .	9
5. INTERNAL_DNS_DOMAIN Usage Guidelines . . . . .	10
6. INTERNAL_DNSSEC_TA Usage Guidelines . . . . .	11
7. Security Considerations . . . . .	12
8. IANA Considerations . . . . .	14
9. References . . . . .	14
9.1. Normative References . . . . .	14
9.2. Informative References . . . . .	15
Authors' Addresses . . . . .	15

## 1. Introduction

Split tunnel Virtual Private Network ("VPN") configurations only send packets with a specific destination IP range, usually chosen from [RFC1918], via the VPN. All other traffic is not sent via the VPN. This allows an enterprise deployment to offer Remote Access VPN services without needing to accept and forward all the non-enterprise related network traffic generated by their remote users. Resources within the enterprise can be accessed by the user via the VPN, while all other traffic generated by the user is not sent over the VPN.

These internal resources tend to only have internal-only DNS names and require the use of special internal-only DNS servers to get resolved. Split DNS [RFC2775] is a common configuration that is part of split tunnel VPN configurations to support configuring Remote Access users to use these special internal-only domain names.

The IKEv2 protocol [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 that the specified DNS domain MUST be resolved using the provided DNS nameserver IP addresses as specified in the INTERNAL\_IP4\_DNS and INTERNAL\_IP6\_DNS Configuration Payloads, causing these requests to use the IPsec connection.

The INTERNAL\_DNSSEC\_TA attribute type is used to convey a DNSSEC trust anchor for such a domain. This is required if the external view uses DNSSEC that would prove the internal view does not exist or would expect a different DNSSEC key on the different versions (internal and external) of the enterprise domain.

If an INTERNAL\_DNS\_DOMAIN is sent by the responder, the responder MUST also include one or more INTERNAL\_IP4\_DNS or INTERNAL\_IP6\_DNS attributes that contain the IPv4 or IPv6 address of the internal DNS server.

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".

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.

### 1.1. Requirements Language

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

## 2. Applicability

If the negotiated IPsec connection is not a split tunnel configuration, the INTERNAL\_DNS\_DOMAIN and INTERNAL\_DNSSEC\_TA Configuration Payloads MUST be ignored. This prevents generic (non-

enterprise) VPN services from overriding the public DNS hierarchy, which could lead to malicious overrides of DNS and DNSSEC.

Such configurations SHOULD instead use only the INTERNAL\_IP4\_DNS and INTERNAL\_IP6\_DNS Configuration Payloads to ensure all of the user's DNS traffic is sent through the IPsec connection and does not leak unencrypted onto the local network, as the local network is often explicitly exempted from IPsec encryption.

For split tunnel configurations, an enterprise can require one or more DNS domains to be resolved via internal DNS servers. This can be a special domain, such as "corp.example.com" for an enterprise that is publicly known to use "example.com". In this case, the remote user needs to be informed what the internal-only domain names are and what the IP addresses of the internal DNS servers are. An enterprise can also run a different version of its public domain on its internal network. In that case, the VPN client is instructed to send DNS queries for the enterprise public domain (eg "example.com") to the internal DNS servers. A configuration for this deployment scenario is referred to as a Split DNS configuration.

Split DNS configurations are often preferable to sending all DNS queries to the enterprise. This allows the remote user to only send DNS queries for the enterprise to the internal DNS servers. The enterprise remains unaware of all non-enterprise (DNS) activity of the user. It also allows the enterprise DNS servers to only be configured for the enterprise DNS domains which removes the legal and technical responsibility of the enterprise to resolve every DNS domain potentially asked for by the remote user.

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. These attributes MUST be accompanied by one or more INTERNAL\_IP4\_DNS or INTERNAL\_IP6\_DNS configuration attributes. The client device can then use the internal DNS server(s) for any DNS queries within the assigned domains. DNS queries for other domains SHOULD be sent to the regular DNS service of the client unless it prefers to use the IPsec tunnel for all its DNS queries. For example, the client could trust the IPsec provided DNS servers more than the locally provided DNS servers especially in the case of connecting to unknown or untrusted networks (eg coffee shops or hotel networks). Or the client could prefer the IPsec based DNS servers because those provide additional features over the local DNS servers.

### 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 INTERNAL\_IP4\_DNS and INTERNAL\_IP6\_DNS 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 includes one or more INTERNAL\_DNS\_DOMAIN attributes as defined in Section 4 as part of the CFG\_REQUEST payload. If an INTERNAL\_DNS\_DOMAIN attribute is included in the CFG\_REQUEST, the initiator MUST also include one or more INTERNAL\_IP4\_DNS or INTERNAL\_IP6\_DNS attributes in the CFG\_REQUEST.

The INTERNAL\_DNS\_DOMAIN attribute sent by the initiator is usually empty but MAY contain a suggested domain name.

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 receiving DNSSEC trust anchors for Split DNS domains, an initiator includes one or more INTERNAL\_DNSSEC\_TA attributes as defined in Section 4 as part of the CFG\_REQUEST payload. If an INTERNAL\_DNSSEC\_TA attribute is included in the CFG\_REQUEST, the initiator MUST also include one or more INTERNAL\_DNS\_DOMAIN attributes in the CFG\_REQUEST. If the initiator includes an INTERNAL\_DNSSEC\_TA attribute, but does not include an INTERNAL\_DNS\_DOMAIN attribute, the responder MAY still respond with both INTERNAL\_DNSSEC\_TA and INTERNAL\_DNS\_DOMAIN attributes.

An initiator MAY convey its current DNSSEC trust anchors for the domain specified in the INTERNAL\_DNS\_DOMAIN attribute. A responder can use this information to determine that it does not need to send a different trust anchor. If the initiator does not wish to convey this information, it MUST use a length of 0.

The absence of INTERNAL\_DNSSEC\_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 an `INTERNAL_DNS_DOMAIN` attribute is included in the `CFG_REPLY`, the responder MUST also include one or both of the `INTERNAL_IP4_DNS` and `INTERNAL_IP6_DNS` attributes in the `CFG_REPLY`. These DNS server configurations are necessary to define which servers can 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 MUST behave as if Split DNS configurations are not supported by the server, unless the initiator has been configured with local policy to define a set of Split DNS domains to use by default.

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 content MAY be ignored or be interpreted as a suggestion by the responder.

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 internal DNSSEC trust anchor information of a DS record (see [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`, `INTERNAL_IP6_DNS`, and `INTERNAL_DNS_DOMAIN` attributes in the `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 will be routed through the tunnel.

The responder replies with two DNS server addresses, and two internal domains, "example.com" and "city.other.test".

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_IP6_ADDRESS()  
  INTERNAL_IP6_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_IP6_ADDRESS(2001:DB8:0:1:2:3:4:5/64)  
  INTERNAL_IP6_DNS(2001:DB8:99:88:77:66:55:44)  
  INTERNAL_DNS_DOMAIN(example.com)  
  INTERNAL_DNS_DOMAIN(city.other.test)
```

#### 3.4.2. Requesting Domains and DNSSEC trust anchors

In this example exchange, the initiator requests INTERNAL\_IP4\_DNS, INTERNAL\_IP6\_DNS, INTERNAL\_DNS\_DOMAIN and INTERNAL\_DNSSEC\_TA attributes in the CFG\_REQUEST.

Any subsequent DNS queries from the initiator for domains such as "www.example.com" or "city.other.test" 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" domain has DNSSEC trust anchors that are returned, while the "other.test" domain has no DNSSEC trust anchors.

```

CP(CFG_REQUEST) =
    INTERNAL_IP4_ADDRESS()
    INTERNAL_IP4_DNS()
    INTERNAL_IP6_ADDRESS()
    INTERNAL_IP6_DNS()
    INTERNAL_DNS_DOMAIN()
    INTERNAL_DNSSEC_TA()

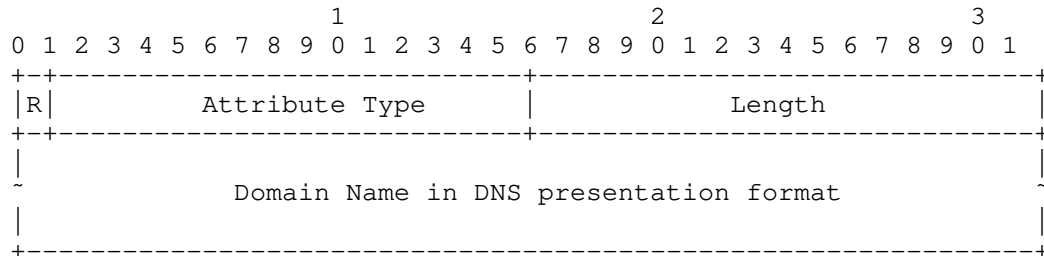
CP(CFG_REPLY) =
    INTERNAL_IP4_ADDRESS(198.51.100.234)
    INTERNAL_IP4_DNS(198.51.100.2)
    INTERNAL_IP4_DNS(198.51.100.4)
    INTERNAL_IP6_ADDRESS(2001:DB8:0:1:2:3:4:5/64)
    INTERNAL_IP6_DNS(2001:DB8:99:88:77:66:55:44)
    INTERNAL_DNS_DOMAIN(example.com)
    INTERNAL_DNSSEC_TA(43547,8,1,B6225AB2CC613E0DCA7962BDC2342EA4...)
    INTERNAL_DNSSEC_TA(31406,8,2,F78CF3344F72137235098ECBBD08947C...)
    INTERNAL_DNS_DOMAIN(city.other.test)

```

#### 4. Payload Formats

All multi-octet fields representing integers are laid out in big endian order (also known as "most significant byte first", or "network byte order").

##### 4.1. INTERNAL\_DNS\_DOMAIN Configuration Attribute Type Request and Reply



- o Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- o Attribute Type (15 bits) set to value 25 for INTERNAL\_DNS\_DOMAIN.
- o Length (2 octets) - Length of domain name.
- o Domain Name (0 or more octets) - A Fully Qualified Domain Name used for Split DNS rules, such as "example.com", in DNS presentation format and using IDNA A-label [RFC5890] for Internationalized Domain Names. Implementors need to be careful that this value is not null-terminated.



## 4.2. INTERNAL\_DNSSEC\_TA Configuration Attribute

An `INTERNAL_DNSSEC_TA` Configuration Attribute can either be empty, or it can contain one Trust Anchor by containing a non-zero Length with a DNSKEY Key Tag, DNSKEY Algorithm, Digest Type and Digest Data fields.

An empty INTERNAL\_DNSSEC\_TA CFG attribute:

										1											2											3
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	
R										Attribute Type										Length (set to 0)												

- o Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- o Attribute Type (15 bits) set to value 26 for INTERNAL\_DNSSEC\_TA.
- o Length (2 octets) - Set to 0 for an empty attribute.

A non-empty `INTERNAL_DNSSEC_TA` CFG attribute:

										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----																																							

- o Reserved (1 bit) - Defined in IKEv2 RFC [RFC7296].
- o Attribute Type (15 bits) set to value 26 for INTERNAL\_DNSSEC\_TA.
- o Length (2 octets) - Length of DNSSEC Trust Anchor data (4 octets plus the length of the Digest Data).
- o DNSKEY Key Tag value (2 octets) - Delegation Signer (DS) Key Tag as specified in [RFC4034] Section 5.1.

- o DNSKEY Algorithm (1 octet) - DNSKEY algorithm value from the IANA DNS Security Algorithm Numbers Registry.
- o Digest Type (1 octet) - DS algorithm value from the IANA Delegation Signer (DS) Resource Record (RR) Type Digest Algorithms Registry.
- o Digest Data (1 or more octets) - The DNSKEY digest as specified in [RFC4034] Section 5.1 in presentation format.

Each INTERNAL\_DNSSEC\_TA attribute in the CFG\_REPLY payload MUST immediately follow a corresponding INTERNAL\_DNS\_DOMAIN attribute. 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. Any INTERNAL\_DNSSEC\_TA attribute that is not immediately preceded by an INTERNAL\_DNS\_DOMAIN or another INTERNAL\_DNSSEC\_TA attribute applying to the same domain name MUST be ignored.

#### 5. INTERNAL\_DNS\_DOMAIN 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.

If a client is configured by local policy to only accept a limited set of INTERNAL\_DNS\_DOMAIN values, the client MUST ignore any other INTERNAL\_DNS\_DOMAIN values.

For each INTERNAL\_DNS\_DOMAIN entry in a CFG\_REPLY payload that is not prohibited by local policy, the client MUST 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 MUST NOT attempt to resolve the provided DNS domains using its external DNS servers. Other domain names SHOULD be resolved using some other external DNS resolver(s), configured independently from IKE. Queries for these other domains MAY be sent to the internal DNS resolver(s) listed in that CFG\_REPLY message, but have no guarantee of being answered. For example, if the INTERNAL\_DNS\_DOMAIN attribute specifies "example.test", then "example.test", "www.example.test" and "mail.eng.example.test" MUST be resolved using the internal DNS resolver(s), but "otherexample.test" and "ple.test" MUST NOT be resolved using the internal resolver and MUST use the system's external DNS resolver(s).

The initiator SHOULD allow the DNS domains listed in the INTERNAL\_DNS\_DOMAIN attributes to resolve to special IP address ranges, such as those of [RFC1918], even if the initiator host is

otherwise configured to block DNS answer containing these special IP address ranges.

When an IKE SA is terminated, the DNS forwarding MUST be unconfigured. This includes deleting the DNS forwarding rules; flushing all cached data for DNS domains provided by the INTERNAL\_DNS\_DOMAIN attribute, including negative cache entries; removing any obtained DNSSEC trust anchors from the list of trust anchors; and clearing the outstanding DNS request queue.

INTERNAL\_DNS\_DOMAIN 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 or DNSSEC exemptions.

## 6. INTERNAL\_DNSSEC\_TA Usage Guidelines

DNS records can be used to publish specific records containing trust anchors for applications. The most common record type is the TLSA record specified in [RFC6698]. This DNS record type publishes which Certificate Authority (CA) certificate or End Entity (EE) certificate to expect for a certain host name. These records are protected by DNSSEC and thus are trustable by the application. Whether to trust TLSA records instead of the traditional WebPKI depends on the local policy of the client. By accepting an INTERNAL\_DNSSEC\_TA trust anchor via IKE from the remote IKE server, the IPsec client might be allowing the remote IKE server to override the trusted certificates for TLS. Similar override concerns apply to other public key or fingerprint-based DNS records, such as OPENPGPKEY, SMIMEA or IPSECKEY records.

Thus, installing an INTERNAL\_DNSSEC\_TA trust anchor can be seen as the equivalent of installing an Enterprise CA certificate. It allows the remote IKE/IPsec server to modify DNS answers including DNSSEC cryptographic signatures by overriding existing DNS information with trust anchor conveyed via IKE and (temporarily) installed on the IKE client. Of specific concern is the overriding of [RFC6698] based TLSA records, which represent a confirmation or override of an existing WebPKI TLS certificate. Other DNS record types that convey cryptographic materials (public keys or fingerprints) are OPENPGPKEY, SMIMEA, SSHP and IPSECKEY records.

IKE clients willing to accept INTERNAL\_DNSSEC\_TA attributes MUST use a whitelist of one or more domains that can be updated out of band. IKE clients with an empty whitelist MUST NOT use any INTERNAL\_DNSSEC\_TA attributes received over IKE. Such clients MAY

interpret receiving an `INTERNAL_DNSSEC_TA` attribute for a non-whitelisted domain as an indication that their local configuration may need to be updated out of band.

IKE clients should take care to only whitelist domains that apply to internal or managed domains, rather than to generic Internet traffic. The DNS root zone (".") MUST be ignored if it appears in a whitelist. Other generic or public domains, such as top-level domains (TLDs), similarly MUST be ignored if these appear in a whitelist unless the entity actually is the operator of the TLD. To determine this, an implementation MAY interactively ask the user when a VPN profile is installed or activated to confirm this. Alternatively, it MAY provide a special override keyword in its provisioning configuration to ensure non-interactive agreement can be achieved only by the party provisioning the VPN client, who presumably is a trusted entity by the end-user. Similarly, an entity might be using a special domain name, such as ".internal", for its internal-only view and might wish to force its provisioning system to accept such a domain in a Split DNS configuration.

Any updates to this whitelist of domain names MUST happen via explicit human interaction or by a trusted automated provision system to prevent malicious invisible installation of trust anchors in case of aIKE server compromise.

IKE clients SHOULD accept any `INTERNAL_DNSSEC_TA` updates for subdomain names of the whitelisted domain names. For example, if "example.net" is whitelisted, then `INTERNAL_DNSSEC_TA` received for "antartica.example.net" SHOULD be accepted.

IKE clients MUST ignore any received `INTERNAL_DNSSEC_TA` attributes for a FQDN for which it did not receive and accept an `INTERNAL_DNS_DOMAIN` Configuration Payload.

In most deployment scenarios, the IKE client has an expectation that it is connecting, using a split-network setup, to a specific organisation or enterprise. A recommended policy would be to only accept `INTERNAL_DNSSEC_TA` directives from that organization's DNS names. However, this might not be possible in all deployment scenarios, such as one where the IKE server is handing out a number of domains that are not within one parent domain.

## 7. Security Considerations

As stated in Section 2, if the negotiated IPsec connection is not a split tunnel configuration, the `INTERNAL_DNS_DOMAIN` and `INTERNAL_DNSSEC_TA` Configuration Payloads MUST be ignored.

Otherwise, generic VPN service providers could maliciously override DNSSEC based trust anchors of public DNS domains.

An initiator **MUST** only accept `INTERNAL_DNSSEC_TAs` for which it has a whitelist, since this mechanism allows the credential used to authenticate an IKEv2 association to be leveraged into authenticating credentials for other connections. Initiators should ensure that they have sufficient trust in the responder when using this mechanism. An initiator **MAY** treat a received `INTERNAL_DNSSEC_TA` for an non-whitelisted domain as a signal to update the whitelist via a non-IKE provisioning mechanism. See Section 6 for additional security considerations for DNSSEC trust anchors.

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** only process the DNS information if the two connections are part of the same logical entity. Otherwise, the client **SHOULD** refuse the DNS information and potentially warn the end-user. For example, if a VPN profile for "Example Corporation" is installed that provides two IPsec connections, one covering 192.168.100.0/24 and one covering 10.13.14.0/24 it could be that both connections negotiate the same `INTERNAL_DNS_DOMAIN` and `INTERNAL_DNSSEC_TA` values. Since these are part of the same remote organisation (or provisioning profile), the Configuration Payloads can be used. However, if a user installs two VPN profiles from two different unrelated independent entities, both of these could be configured to use the same domain, for example ".internal". These two connections **MUST NOT** be allowed to be active at the same time.

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 explicitly requested such delegation by specifying the domain specifically using a `INTERNAL_DNS_DOMAIN` request.

Deployments that configure `INTERNAL_DNS_DOMAIN` domains should pay close attention to their use of indirect reference RRTypes in their

internal-only domain names. Examples of such RRtypes are NS, CNAME, DNAME, MX or SRV records. For example, if the MX record for "internal.example.com" points to "mx.internal.example.net", then both "internal.example.com" and "internal.example.net" should be sent using an INTERNAL\_DNS\_DOMAIN Configuration Payload.

IKE clients MAY want to require whitelisted domains for Top Level Domains (TLDs) and Second Level Domains (SLDs) to further prevent malicious DNS redirections for well known domains. This prevents users from unknowingly giving DNS queries to third parties. This is even more important if those well known domains are not deploying DNSSEC, as the VPN service provider could then even modify the DNS answers without detection.

The content of INTERNAL\_DNS\_DOMAIN and INTERNAL\_DNSSEC\_TA may be passed to another (DNS) program for processing. As with any network input, the content SHOULD be considered untrusted and handled accordingly.

## 8. 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
25	INTERNAL_DNS_DOMAIN	YES	0 or more	[this document]
26	INTERNAL_DNSSEC_TA	YES	0 or more	[this document]

Figure 1

## 9. References

### 9.1. 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, <<https://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, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc5890>>.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 9.2. Informative References

- [RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, DOI 10.17487/RFC2775, February 2000, <<https://www.rfc-editor.org/info/rfc2775>>.
- [RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <<https://www.rfc-editor.org/info/rfc7435>>.

## Authors' Addresses

Tommy Pauly  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014  
US

Email: [tpauly@apple.com](mailto:tpauly@apple.com)

Paul Wouters  
Red Hat

Email: [pwouters@redhat.com](mailto:pwouters@redhat.com)



Network  
Internet-Draft  
Intended status: Standards Track  
Expires: December 1, 2017

T. Pauly  
Apple Inc.  
S. Touati  
Ericsson  
R. Mantha  
Cisco Systems  
May 30, 2017

TCP Encapsulation of IKE and IPsec Packets  
draft-ietf-ipsecme-tcp-encaps-10

Abstract

This document describes a method to transport IKE and IPsec packets over a TCP connection for traversing network middleboxes that may block IKE negotiation over UDP. This method, referred to as TCP encapsulation, involves sending both IKE packets for Security Association establishment and ESP packets over a TCP connection. This method is intended to be used as a fallback option when IKE cannot be negotiated over UDP.

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 1, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Prior Work and Motivation . . . . .	3
1.2. Terminology and Notation . . . . .	4
2. Configuration . . . . .	5
3. TCP-Encapsulated Header Formats . . . . .	5
3.1. TCP-Encapsulated IKE Header Format . . . . .	6
3.2. TCP-Encapsulated ESP Header Format . . . . .	6
4. TCP-Encapsulated Stream Prefix . . . . .	7
5. Applicability . . . . .	7
5.1. Recommended Fallback from UDP . . . . .	8
6. Connection Establishment and Teardown . . . . .	8
7. Interaction with NAT Detection Payloads . . . . .	10
8. Using MOBIKE with TCP encapsulation . . . . .	10
9. Using IKE Message Fragmentation with TCP encapsulation . . . . .	11
10. Considerations for Keep-alives and DPD . . . . .	11
11. Middlebox Considerations . . . . .	12
12. Performance Considerations . . . . .	12
12.1. TCP-in-TCP . . . . .	12
12.2. Added Reliability for Unreliable Protocols . . . . .	13
12.3. Quality of Service Markings . . . . .	13
12.4. Maximum Segment Size . . . . .	13
12.5. Tunnelling ECN in TCP . . . . .	14
13. Security Considerations . . . . .	14
14. IANA Considerations . . . . .	15
15. Acknowledgments . . . . .	15
16. References . . . . .	15
16.1. Normative References . . . . .	15
16.2. Informative References . . . . .	16
Appendix A. Using TCP encapsulation with TLS . . . . .	17
Appendix B. Example exchanges of TCP Encapsulation with TLS . . . . .	17
B.1. Establishing an IKE session . . . . .	17
B.2. Deleting an IKE session . . . . .	19
B.3. Re-establishing an IKE session . . . . .	20
B.4. Using MOBIKE between UDP and TCP Encapsulation . . . . .	21
Authors' Addresses . . . . .	23

## 1. Introduction

IKEv2 [RFC7296] is a protocol for establishing IPsec Security Associations (SAs), using IKE messages over UDP for control traffic, and using Encapsulating Security Payload (ESP) messages for encrypted data traffic. Many network middleboxes that filter traffic on public hotspots block all UDP traffic, including IKE and IPsec, but allow TCP connections through since they appear to be web traffic. Devices on these networks that need to use IPsec (to access private enterprise networks, to route voice-over-IP calls to carrier networks, or because of security policies) are unable to establish IPsec SAs. This document defines a method for encapsulating both the IKE control messages as well as the IPsec data messages within a TCP connection.

Using TCP as a transport for IPsec packets adds a third option to the list of traditional IPsec transports:

1. Direct. Currently, IKE negotiations begin over UDP port 500. If no NAT is detected between the Initiator and the Responder, then subsequent IKE packets are sent over UDP port 500 and IPsec data packets are sent using ESP [RFC4303].
2. UDP Encapsulation [RFC3948]. If a NAT is detected between the Initiator and the Responder, then subsequent IKE packets are sent over UDP port 4500 with four bytes of zero at the start of the UDP payload and ESP packets are sent out over UDP port 4500. Some peers default to using UDP encapsulation even when no NAT are detected on the path as some middleboxes do not support IP protocols other than TCP and UDP.
3. TCP Encapsulation. If both of the other two methods are not available or appropriate, both IKE negotiation packets as well as ESP packets can be sent over a single TCP connection to the peer.

Direct use of ESP or UDP Encapsulation should be preferred by IKE implementations due to performance concerns when using TCP Encapsulation Section 12. Most implementations should use TCP Encapsulation only on networks where negotiation over UDP has been attempted without receiving responses from the peer, or if a network is known to not support UDP.

### 1.1. Prior Work and Motivation

Encapsulating IKE connections within TCP streams is a common approach to solve the problem of UDP packets being blocked by network middleboxes. The goal of this document is to promote

interoperability by providing a standard method of framing IKE and ESP message within streams, and to provide guidelines for how to configure and use TCP encapsulation.

Some previous alternatives include:

Cellular Network Access Interworking Wireless LAN (IWLAN) uses IKEv2 to create secure connections to cellular carrier networks for making voice calls and accessing other network services over Wi-Fi networks. 3GPP has recommended that IKEv2 and ESP packets be sent within a TLS connection to be able to establish connections on restrictive networks.

ISAKMP over TCP Various non-standard extensions to ISAKMP have been deployed that send IPsec traffic over TCP or TCP-like packets.

SSL VPNs Many proprietary VPN solutions use a combination of TLS and IPsec in order to provide reliability. These often run on TCP port 443.

IKEv2 over TCP IKEv2 over TCP as described in [I-D.nir-ipsecme-ike-tcp] is used to avoid UDP fragmentation.

The goal of this specification is to provide a standardized method for using TCP streams to transport IPsec that is compatible with the current IKE standard, and avoids the overhead of other alternatives that always rely on TCP or TLS.

## 1.2. Terminology and Notation

This document distinguishes between the IKE peer that initiates TCP connections to be used for TCP encapsulation and the roles of Initiator and Responder for particular IKE messages. During the course of IKE exchanges, the role of IKE Initiator and Responder may swap for a given SA (as with IKE SA Rekeys), while the initiator of the TCP connection is still responsible for tearing down the TCP connection and re-establishing it if necessary. For this reason, this document will use the term "TCP Originator" to indicate the IKE peer that initiates TCP connections. The peer that receives TCP connections will be referred to as the "TCP Responder". If an IKE SA is rekeyed one or more times, the TCP Originator MUST remain the peer that originally initiated the first IKE SA.

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. Configuration

One of the main reasons to use TCP encapsulation is that UDP traffic may be entirely blocked on a network. Because of this, support for TCP encapsulation is not specifically negotiated in the IKE exchange. Instead, support for TCP encapsulation must be pre-configured on both the TCP Originator and the TCP Responder.

Implementations **MUST** support TCP encapsulation on TCP port 4500, which is reserved for IPsec NAT Traversal.

Beyond a flag indicating support for TCP encapsulation, the configuration for each peer can include the following optional parameters:

- o Alternate TCP ports on which the specific TCP Responder listens for incoming connections. Note that the TCP Originator may initiate TCP connections to the TCP Responder from any local port.
- o An extra framing protocol to use on top of TCP to further encapsulate the stream of IKE and IPsec packets. See Appendix A for a detailed discussion.

Since TCP encapsulation of IKE and IPsec packets adds overhead and has potential performance trade-offs compared to direct or UDP-encapsulated SAs (as described in Performance Considerations, Section 12), implementations **SHOULD** prefer ESP direct or UDP encapsulated SAs over TCP encapsulated SAs when possible.

## 3. TCP-Encapsulated Header Formats

Like UDP encapsulation, TCP encapsulation uses the first four bytes of a message to differentiate IKE and ESP messages. TCP encapsulation also adds a length field to define the boundaries of messages within a stream. The message length is sent in a 16-bit field that precedes every message. If the first 32-bits of the message are zeros (a Non-ESP Marker), then the contents comprise an IKE message. Otherwise, the contents comprise an ESP message. Authentication Header (AH) messages are not supported for TCP encapsulation.

Although a TCP stream may be able to send very long messages, implementations **SHOULD** limit message lengths to typical UDP datagram ESP payload lengths. The maximum message length is used as the effective MTU for connections that are being encrypted using ESP, so the maximum message length will influence characteristics of inner connections, such as the TCP Maximum Segment Size (MSS).

Note that this method of encapsulation will also work for placing IKE and ESP messages within any protocol that presents a stream abstraction, beyond TCP.

### 3.1. TCP-Encapsulated IKE Header Format

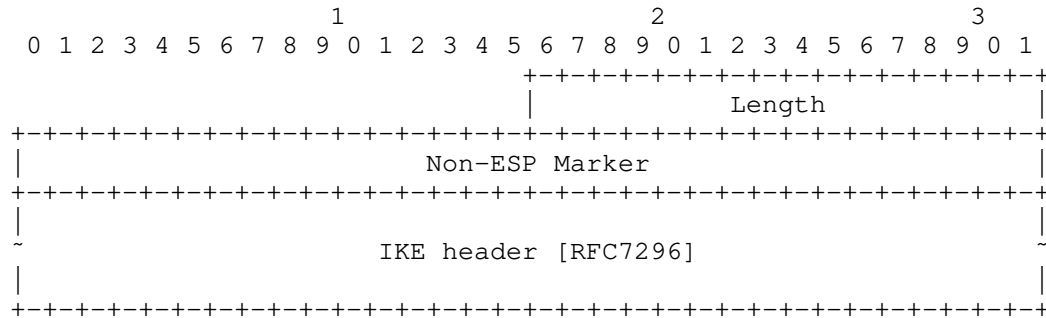


Figure 1

The IKE header is preceded by a 16-bit length field in network byte order that specifies the length of the IKE message (including the Non-ESP marker) within the TCP stream. As with IKE over UDP port 4500, a zeroed 32-bit Non-ESP Marker is inserted before the start of the IKE header in order to differentiate the traffic from ESP traffic between the same addresses and ports.

- o Length (2 octets, unsigned integer) - Length of the IKE packet including the Length Field and Non-ESP Marker.

### 3.2. TCP-Encapsulated ESP Header Format

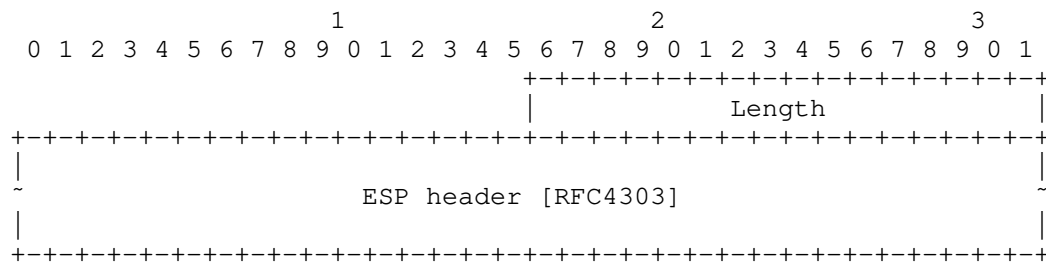


Figure 2

The ESP header is preceded by a 16-bit length field in network byte order that specifies the length of the ESP packet within the TCP stream.

The SPI field in the ESP header MUST NOT be a zero value.

- o Length (2 octets, unsigned integer) - Length of the ESP packet including the Length Field.

#### 4. TCP-Encapsulated Stream Prefix

Each stream of bytes used for IKE and IPsec encapsulation MUST begin with a fixed sequence of six bytes as a magic value, containing the characters "IKETCP" as ASCII values. This value is intended to identify and validate that the TCP connection is being used for TCP encapsulation as defined in this document, to avoid conflicts with the prevalence of previous non-standard protocols that used TCP port 4500. This value is only sent once, by the TCP Originator only, at the beginning of any stream of IKE and ESP messages.

If other framing protocols are used within TCP to further encapsulate or encrypt the stream of IKE and ESP messages, the Stream Prefix must be at the start of the TCP Originator's IKE and ESP message stream within the added protocol layer [Appendix A]. Although some framing protocols do support negotiating inner protocols, the stream prefix should always be used in order for implementations to be as generic as possible and not rely on other framing protocols on top of TCP.

0	1	2	3	4	5
0x49	0x4b	0x45	0x54	0x43	0x50

Figure 3

#### 5. Applicability

TCP encapsulation is applicable only when it has been configured to be used with specific IKE peers. If a Responder is configured to use TCP encapsulation, it MUST listen on the configured port(s) in case any peers will initiate new IKE sessions. Initiators MAY use TCP encapsulation for any IKE session to a peer that is configured to support TCP encapsulation, although it is recommended that Initiators should only use TCP encapsulation when traffic over UDP is blocked.

Since the support of TCP encapsulation is a configured property, not a negotiated one, it is recommended that if there are multiple IKE endpoints representing a single peer (such as multiple machines with different IP addresses when connecting by Fully-Qualified Domain Name, or endpoints used with IKE redirection), all of the endpoints equally support TCP encapsulation.

If TCP encapsulation is being used for a specific IKE SA, all messages for that IKE SA and its Child SAs MUST be sent over a TCP connection until the SA is deleted or MOBIKE is used to change the SA endpoints and/or encapsulation protocol. See Section 8 for more details on using MOBIKE to transition between encapsulation modes.

#### 5.1. Recommended Fallback from UDP

Since UDP is the preferred method of transport for IKE messages, implementations that use TCP encapsulation should have an algorithm for deciding when to use TCP after determining that UDP is unusable. If an Initiator implementation has no prior knowledge about the network it is on and the status of UDP on that network, it SHOULD always attempt negotiate IKE over UDP first. IKEv2 defines how to use retransmission timers with IKE messages, and IKE\_SA\_INIT messages specifically [RFC7296]. Generally, this means that the implementation will define a frequency of retransmission, and the maximum number of retransmissions allowed before marking the IKE SA as failed. An implementation can attempt negotiation over TCP once it has hit the maximum retransmissions over UDP, or slightly before to reduce connection setup delays. It is recommended that the initial message over UDP is retransmitted at least once before falling back to TCP, unless the Initiator knows beforehand that the network is likely to block UDP.

#### 6. Connection Establishment and Teardown

When the IKE Initiator uses TCP encapsulation, it will initiate a TCP connection to the Responder using the configured TCP port. The first bytes sent on the stream MUST be the stream prefix value [Section 4]. After this prefix, encapsulated IKE messages will negotiate the IKE SA and initial Child SA [RFC7296]. After this point, both encapsulated IKE Figure 1 and ESP Figure 2 messages will be sent over the TCP connection. The TCP Responder MUST wait for the entire stream prefix to be received on the stream before trying to parse out any IKE or ESP messages. The stream prefix is sent only once, and only by the TCP Originator.

In order to close an IKE session, either the Initiator or Responder SHOULD gracefully tear down IKE SAs with DELETE payloads. Once the SA has been deleted, the TCP Originator SHOULD close the TCP connection if it does not intend to use the connection for another IKE session to the TCP Responder. If the connection is left idle, and the TCP Responder needs to clean up resources, the TCP Responder MAY close the TCP connection.

An unexpected FIN or a RST on the TCP connection may indicate either a loss of connectivity, an attack, or some other error. If a DELETE



payload has not been sent, both sides SHOULD maintain the state for their SAs for the standard lifetime or time-out period. The TCP Originator is responsible for re-establishing the TCP connection if it is torn down for any unexpected reason. Since new TCP connections may use different ports due to NAT mappings or local port allocations changing, the TCP Responder MUST allow packets for existing SAs to be received from new source ports.

A peer MUST discard a partially received message due to a broken connection.

Whenever the TCP Originator opens a new TCP connection to be used for an existing IKE SA, it MUST send the stream prefix first, before any IKE or ESP messages. This follows the same behavior as the initial TCP connection.

If a TCP connection is being used to resume a previous IKE session, the TCP Responder can recognize the session using either the IKE SPI from an encapsulated IKE message or the ESP SPI from an encapsulated ESP message. If the session had been fully established previously, it is suggested that the TCP Originator send an UPDATE\_SA\_ADDRESSES message if MOBIKE is supported, or an INFORMATIONAL message (a keepalive) otherwise.

The TCP Responder MUST NOT accept any messages for the existing IKE session on a new incoming connection unless that connection begins with the stream prefix. If either the TCP Originator or TCP Responder detects corruption on a connection that was started with a valid stream prefix, it SHOULD close the TCP connection. The connection can be determined as corrupted if there are too many subsequent messages that cannot be parsed as valid IKE messages or ESP messages with known SPIs, or if the authentication check for an ESP message with a known SPI fails. Implementations SHOULD NOT tear down a connection if only a single ESP message has an unknown SPI, since the SPI databases may be momentarily out of sync. If there is instead a syntax issue within an IKE message, an implementation MUST send the INVALID\_SYNTAX notify payload and tear down the IKE SA as usual, rather than tearing down the TCP connection directly.

An TCP Originator SHOULD only open one TCP connection per IKE SA, over which it sends all of the corresponding IKE and ESP messages. This helps ensure that any firewall or NAT mappings allocated for the TCP connection apply to all of the traffic associated with the IKE SA equally.

Similarly, a TCP Responder SHOULD at any given time send packets for an IKE SA and its Child SAs over only one TCP connection. It SHOULD choose the TCP connection on which it last received a valid and

decryptable IKE or ESP message. In order to be considered valid for choosing a TCP connection, an IKE message must be successfully decrypted and authenticated, not be a retransmission of a previously received message, and be within the expected window for IKE message IDs. Similarly, an ESP message must pass authentication checks and be decrypted, not be a replay of a previous message.

Since a connection may be broken and a new connection re-established by the TCP Originator without the TCP Responder being aware, a TCP Responder SHOULD accept receiving IKE and ESP messages on both old and new connections until the old connection is closed by the TCP Originator. A TCP Responder MAY close a TCP connection that it perceives as idle and extraneous (one previously used for IKE and ESP messages that has been replaced by a new connection).

Multiple IKE SAs MUST NOT share a single TCP connection, unless one is a rekey of an existing IKE SA, in which case there will temporarily be two IKE SAs on the same TCP connection.

## 7. Interaction with NAT Detection Payloads

When negotiating over UDP port 500, IKE\_SA\_INIT packets include NAT\_DETECTION\_SOURCE\_IP and NAT\_DETECTION\_DESTINATION\_IP payloads to determine if UDP encapsulation of IPsec packets should be used. These payloads contain SHA-1 digests of the SPIs, IP addresses, and ports as defined in [RFC7296]. IKE\_SA\_INIT packets sent on a TCP connection SHOULD include these payloads with the same content as when sending over UDP, and SHOULD use the applicable TCP ports when creating and checking the SHA-1 digests.

If a NAT is detected due to the SHA-1 digests not matching the expected values, no change should be made for encapsulation of subsequent IKE or ESP packets, since TCP encapsulation inherently supports NAT traversal. Implementations MAY use the information that a NAT is present to influence keep-alive timer values.

If a NAT is detected, implementations need to handle transport mode TCP and UDP packet checksum fixup as defined for UDP encapsulation in [RFC3948].

## 8. Using MOBIKE with TCP encapsulation

When an IKE session that has negotiated MOBIKE [RFC4555] is transitioning between networks, the Initiator of the transition may switch between using TCP encapsulation, UDP encapsulation, or no encapsulation. Implementations that implement both MOBIKE and TCP encapsulation MUST support dynamically enabling and disabling TCP encapsulation as interfaces change.

When a MOBIKE-enabled Initiator changes networks, the UPDATE\_SA\_ADDRESSES notification SHOULD be sent out first over UDP before attempting over TCP. If there is a response to the UPDATE\_SA\_ADDRESSES notification sent over UDP, then the ESP packets should be sent directly over IP or over UDP port 4500 (depending on if a NAT was detected), regardless of if a connection on a previous network was using TCP encapsulation. Similarly, if the Responder only responds to the UPDATE\_SA\_ADDRESSES notification over TCP, then the ESP packets should be sent over the TCP connection, regardless of if a connection on a previous network did not use TCP encapsulation.

#### 9. Using IKE Message Fragmentation with TCP encapsulation

IKE Message Fragmentation [RFC7383] is not required when using TCP encapsulation, since a TCP stream already handles the fragmentation of its contents across packets. Since fragmentation is redundant in this case, implementations might choose to not negotiate IKE fragmentation. Even if fragmentation is negotiated, an implementation SHOULD NOT send fragments when going over a TCP connection, although it MUST support receiving fragments.

If an implementation supports both MOBIKE and IKE fragmentation, it SHOULD negotiate IKE fragmentation over a TCP encapsulated session in case the session switches to UDP encapsulation on another network.

#### 10. Considerations for Keep-alives and DPD

Encapsulating IKE and IPsec inside of a TCP connection can impact the strategy that implementations use to detect peer liveness and to maintain middlebox port mappings. Peer liveness should be checked using IKE Informational packets [RFC7296].

In general, TCP port mappings are maintained by NATs longer than UDP port mappings, so IPsec ESP NAT keep-alives [RFC3948] SHOULD NOT be sent when using TCP encapsulation. Any implementation using TCP encapsulation MUST silently drop incoming NAT keep-alive packets, and not treat them as errors. NAT keep-alive packets over a TCP encapsulated IPsec connection will be sent with a length value of 1 byte, whose value is 0xFF Figure 2.

Note that depending on the configuration of TCP and TLS on the connection, TCP keep-alives [RFC1122] and TLS keep-alives [RFC6520] may be used. These MUST NOT be used as indications of IKE peer liveness.

## 11. Middlebox Considerations

Many security networking devices such as Firewalls or Intrusion Prevention Systems, network optimization/acceleration devices and Network Address Translation (NAT) devices keep the state of sessions that traverse through them.

These devices commonly track the transport layer and/or the application layer data to drop traffic that is anomalous or malicious in nature. While many of these devices will be more likely to pass TCP-encapsulated traffic as opposed to UDP-encapsulated traffic, some may still block or interfere with TCP-encapsulated IKE and IPsec.

A network device that monitors the transport layer will track the state of TCP sessions, such as TCP sequence numbers. TCP encapsulation of IKE should therefore use standard TCP behaviors to avoid being dropped by middleboxes.

## 12. Performance Considerations

Several aspects of TCP encapsulation for IKE and IPsec packets may negatively impact the performance of connections within a tunnel-mode IPsec SA. Implementations should be aware of these performance impacts and take these into consideration when determining when to use TCP encapsulation. Implementations SHOULD favor using direct ESP or UDP encapsulation over TCP encapsulation whenever possible.

### 12.1. TCP-in-TCP

If the outer connection between IKE peers is over TCP, inner TCP connections may suffer effects from using TCP within TCP. Running TCP within TCP is discouraged, since the TCP algorithms generally assume that they are running over an unreliable datagram layer.

If the outer (tunnel) TCP connection experiences packet loss, this loss will be hidden from any inner TCP connections, since the outer connection will retransmit to account for the losses. Since the outer TCP connection will deliver the inner messages in order, any messages after a lost packet may have to wait until the loss is recovered. This means that loss on the outer connection will be interpreted only as delay by inner connections. The burstiness of inner traffic can increase, since a large number of inner packets may be delivered across the tunnel at once. The inner TCP connection may interpret a long period of delay as a transmission problem, triggering a retransmission timeout, which will cause spurious retransmissions. The sending rate of the inner connection may be unnecessarily reduced if the retransmissions are not detected as spurious in time.

The inner TCP connection's round-trip-time estimation will be affected by the burstiness of the outer TCP connection if there are long delays when packets are retransmitted by the outer TCP connection. This will make the congestion control loop of the inner TCP traffic less reactive, potentially permanently leading to a lower sending rate than the outer TCP would allow for.

TCP-in-TCP can also lead to increased buffering, or bufferbloat. This can occur when the window size of the outer TCP connection is reduced, and becomes smaller than the window sizes of the inner TCP connections. This can lead to packets backing up in the outer TCP connection's send buffers. In order to limit this effect, the outer TCP connection should have limits on its send buffer size, and on the rate at which it reduces its window size.

Note that any negative effects will be shared between all flows going through the outer TCP connection. This is of particular concern for any latency-sensitive or real-time applications using the tunnel. If such traffic is using a TCP encapsulated IPsec connection, it is recommended that the number of inner connections sharing the tunnel be limited as much as possible.

#### 12.2. Added Reliability for Unreliable Protocols

Since ESP is an unreliable protocol, transmitting ESP packets over a TCP connection will change the fundamental behavior of the packets. Some application-level protocols that prefer packet loss to delay (such as Voice over IP or other real-time protocols) may be negatively impacted if their packets are retransmitted by the TCP connection due to packet loss.

#### 12.3. Quality of Service Markings

Quality of Service (QoS) markings, such as DSCP and Traffic Class, should be used with care on TCP connections used for encapsulation. Individual packets SHOULD NOT use different markings than the rest of the connection, since packets with different priorities may be routed differently and cause unnecessary delays in the connection.

#### 12.4. Maximum Segment Size

A TCP connection used for IKE encapsulation SHOULD negotiate its maximum segment size (MSS) in order to avoid unnecessary fragmentation of packets.

### 12.5. Tunnelling ECN in TCP

Since there is not a one-to-one relationship between outer IP packets and inner ESP/IP messages when using TCP encapsulation, the markings for Explicit Congestion Notification (ECN) [RFC3168] cannot be simply mapped. However, any ECN Congestion Experienced (CE) marking on inner messages should be preserved through the tunnel.

Implementations SHOULD follow the ECN compatibility mode as described in [RFC6040]. In compatibility mode, the outer TCP connection SHOULD mark its packets as not ECN-capable, and MUST NOT clear any ECN markings on inner packets. Note that outer packets may be ECN marked even though the outer connection did not negotiate support for ECN. If an implementation receives such an outer packet, it MAY propagate the markings as described in the Default Tunnel Egress Behaviour [RFC6040] for any inner packet contained within a single outer TCP packet, or simply apply the rules as if the outer packet were Not-ECT if the inner packet spans multiple outer packets.

### 13. Security Considerations

IKE Responders that support TCP encapsulation may become vulnerable to new Denial-of-Service (DoS) attacks that are specific to TCP, such as SYN-flooding attacks. TCP Responders should be aware of this additional attack-surface.

TCP Responders should be careful to ensure that the stream prefix "IKETCP" uniquely identifies incoming streams as ones that use the TCP encapsulation protocol, and they are not running any other protocols on the same listening port that could conflict with this.

Attackers may be able to disrupt the TCP connection by sending spurious RST packets. Due to this, implementations SHOULD make sure that IKE session state persists even if the underlying TCP connection is torn down.

If MOBIKE is being used, all of the security considerations outlined for MOBIKE apply [RFC4555].

Similarly to MOBIKE, TCP encapsulation requires a TCP Responder to handle changing of source address and port due to network or connection disruption. The successful delivery of valid IKE or ESP messages over a new TCP connection is used by the TCP Responder to determine where to send subsequent responses. If an attacker is able to send packets on a new TCP connection that pass the validation checks of the TCP Responder, it can influence which path future packets take. For this reason, the validation of messages on the TCP Responder must include decryption, authentication, and replay checks.

Since TCP provides a reliable, in-order delivery of ESP messages, the ESP Anti-Replay Window size SHOULD be set to 1. See [RFC4303] for a complete description of the ESP Anti-Replay Window. This increases the protection of implementations against replay attacks.

#### 14. IANA Considerations

TCP port 4500 is already allocated to IPsec for NAT Traversal. This port SHOULD be used for TCP encapsulated IKE and ESP as described in this document.

This document updates the reference for TCP port 4500:

Keyword	Decimal	Description	Reference
-----	-----	-----	-----
ipsec-nat-t	4500/tcp	IPsec NAT-Traversal	[RFC-this-rfc]

Figure 4

#### 15. Acknowledgments

The authors would like to acknowledge the input and advice of Stuart Cheshire, Delziel Fernandes, Yoav Nir, Christoph Paasch, Yaron Sheffer, David Schinazi, Graham Bartlett, Byju Pularikkal, March Wu, Kingwel Xie, Valery Smyslov, Jun Hu, and Tero Kivinen. Special thanks to Eric Kinnear for his implementation work.

#### 16. References

##### 16.1. 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>>.
- [RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <<http://www.rfc-editor.org/info/rfc3948>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<http://www.rfc-editor.org/info/rfc6040>>.

- [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>>.

## 16.2. Informative References

- [I-D.nir-ipsecme-ike-tcp] Nir, Y., "A TCP transport for the Internet Key Exchange", draft-nir-ipsecme-ike-tcp-01 (work in progress), July 2012.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<http://www.rfc-editor.org/info/rfc1122>>.
- [RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, DOI 10.17487/RFC2817, May 2000, <<http://www.rfc-editor.org/info/rfc2817>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<http://www.rfc-editor.org/info/rfc3168>>.
- [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006, <<http://www.rfc-editor.org/info/rfc4555>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension", RFC 6520, DOI 10.17487/RFC6520, February 2012, <<http://www.rfc-editor.org/info/rfc6520>>.
- [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>>.



## Appendix A. Using TCP encapsulation with TLS

This section provides recommendations on how to use TLS in addition to TCP encapsulation.

When using TCP encapsulation, implementations may choose to use TLS [RFC5246] on the TCP connection to be able to traverse middle-boxes, which may otherwise block the traffic.

If a web proxy is applied to the ports used for the TCP connection, and TLS is being used, the TCP Originator can send an HTTP CONNECT message to establish an SA through the proxy [RFC2817].

The use of TLS should be configurable on the peers, and may be used as the default when using TCP encapsulation, or else be a fallback when basic TCP encapsulation fails. The TCP Responder may expect to read encapsulated IKE and ESP packets directly from the TCP connection, or it may expect to read them from a stream of TLS data packets. The TCP Originator should be pre-configured to use TLS or not when communicating with a given port on the TCP Responder.

When new TCP connections are re-established due to a broken connection, TLS must be re-negotiated. TLS Session Resumption is recommended to improve efficiency in this case.

The security of the IKE session is entirely derived from the IKE negotiation and key establishment and not from the TLS session (which in this context is only used for encapsulation purposes), therefore when TLS is used on the TCP connection, both the TCP Originator and TCP Responder SHOULD allow the NULL cipher to be selected for performance reasons.

Implementations should be aware that the use of TLS introduces another layer of overhead requiring more bytes to transmit a given IKE and IPsec packet. For this reason, direct ESP, UDP encapsulation, or TCP encapsulation without TLS should be preferred in situations in which TLS is not required in order to traverse middle-boxes.

## Appendix B. Example exchanges of TCP Encapsulation with TLS

## B.1. Establishing an IKE session

```

                        Client                               Server
                    -----
1)  ----- TCP Connection -----
    (IP_I:Port_I -> IP_R:Port_R)
    TcpSyn                      ----->

```

```

                                <-----
                                ----->
                                TcpSyn,Ack

    TcpAck

2)  ----- TLS Session -----
    ClientHello ----->
                                ServerHello
                                Certificate*
                                ServerKeyExchange*
                                ServerHelloDone
    ClientKeyExchange <-----
    CertificateVerify*
    [ChangeCipherSpec]
    Finished ----->
                                [ChangeCipherSpec]
                                <-----
                                Finished

3)  ----- Stream Prefix -----
    "IKETCP" ----->

4)  ----- IKE Session -----
    Length + Non-ESP Marker ----->
    IKE_SA_INIT
    HDR, SAi1, KEi, Ni,
    [N(NAT_DETECTION*_IP)]
                                <----- Length + Non-ESP Marker
                                IKE_SA_INIT
                                HDR, SAr1, KEr, Nr,
                                [N(NAT_DETECTION*_IP)]
    Length + Non-ESP Marker ----->
    first IKE_AUTH
    HDR, SK {IDi, [CERTREQ]
    CP(CFG_REQUEST), IDr,
    SAi2, TSi, TSr, ...}
                                <----- Length + Non-ESP Marker
                                first IKE_AUTH
                                HDR, SK {IDr, [CERT], AUTH,
                                EAP, SAR2, TSi, TSr}
    Length + Non-ESP Marker ----->
    IKE_AUTH + EAP
    repeat 1..N times
                                <----- Length + Non-ESP Marker
                                IKE_AUTH + EAP
    Length + Non-ESP Marker ----->
    final IKE_AUTH
    HDR, SK {AUTH}
                                <----- Length + Non-ESP Marker
                                final IKE_AUTH
                                HDR, SK {AUTH, CP(CFG_REPLY),
                                SA, TSi, TSr, ...}

```

```

----- IKE and IPsec SAs Established -----
Length + ESP frame          ----->

```

Figure 5

1. Client establishes a TCP connection with the server on port 4500, or an alternate pre-configured port that the server is listening on.
2. If configured to use TLS, the client initiates a TLS handshake. During the TLS handshake, the server SHOULD NOT request the client's certificate, since authentication is handled as part of IKE negotiation.
3. Client send the Stream Prefix for TCP encapsulated IKE Section 4 traffic to signal the beginning of IKE negotiation.
4. Client and server establish an IKE connection. This example shows EAP-based authentication, although any authentication type may be used.

#### B.2. Deleting an IKE session

	Client -----		Server -----
1)	----- IKE Session -----		-----
	Length + Non-ESP Marker ----->		
	INFORMATIONAL		
	HDR, SK {[N,] [D,]		
	[CP,] ...}		
		<-----	Length + Non-ESP Marker
			INFORMATIONAL
			HDR, SK {[N,] [D,]
			[CP,] ...}
2)	----- TLS Session -----		-----
	close_notify ----->		
		<-----	close_notify
3)	----- TCP Connection -----		-----
	TcpFin ----->		
		<-----	Ack
		<-----	TcpFin
	Ack ----->		
	----- IKE SA Deleted -----		-----

Figure 6

1. Client and server exchange INFORMATIONAL messages to notify IKE SA deletion.
2. Client and server negotiate TLS session deletion using TLS CLOSE\_NOTIFY.
3. The TCP connection is torn down.

The deletion of the IKE SA should lead to the disposal of the underlying TLS and TCP state.

### B.3. Re-establishing an IKE session

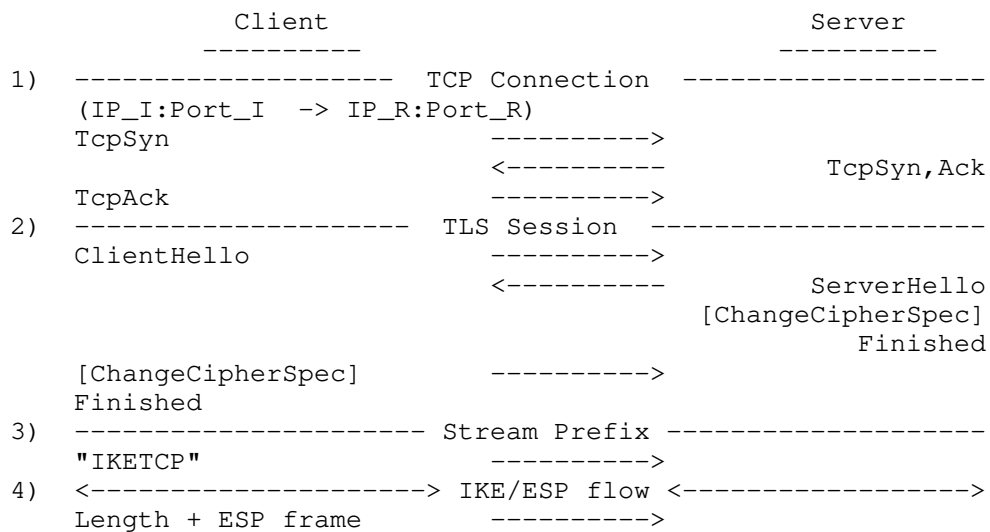


Figure 7

1. If a previous TCP connection was broken (for example, due to a RST), the client is responsible for re-initiating the TCP connection. The TCP Originator's address and port (IP\_I and Port\_I) may be different from the previous connection's address and port.
2. In ClientHello TLS message, the client SHOULD send the Session ID it received in the previous TLS handshake if available. It is up to the server to perform either an abbreviated handshake or full handshake based on the session ID match.

3. After TCP and TLS are complete, the client sends the Stream Prefix for TCP encapsulated IKE traffic Section 4.
4. The IKE and ESP packet flow can resume. If MOBIKE is being used, the Initiator SHOULD send UPDATE\_SA\_ADDRESSES.

#### B.4. Using MOBIKE between UDP and TCP Encapsulation

	Client -----		Server -----
	(IP_I1:UDP500 -> IP_R:UDP500)		
1)	----- IKE_SA_INIT Exchange -----		
	(IP_I1:UDP4500 -> IP_R:UDP4500)		
	Non-ESP Marker ----->		
	Initial IKE_AUTH		
	HDR, SK { IDi, CERT, AUTH,		
	CP(CFG_REQUEST),		
	SAi2, TSi, TSr,		
	N(MOBIKE_SUPPORTED) }		
		<-----	Non-ESP Marker
			Initial IKE_AUTH
			HDR, SK { IDr, CERT, AUTH,
			EAP, SAr2, TSi, TSr,
			N(MOBIKE_SUPPORTED) }
	<-----		
	----- IKE SA establishment ----->		
2)	----- MOBIKE Attempt on new network -----		
	(IP_I2:UDP4500 -> IP_R:UDP4500)		
	Non-ESP Marker ----->		
	INFORMATIONAL		
	HDR, SK { N(UPDATE_SA_ADDRESSES),		
	N(NAT_DETECTION_SOURCE_IP),		
	N(NAT_DETECTION_DESTINATION_IP) }		
3)	----- TCP Connection -----		
	(IP_I2:Port_I -> IP_R:Port_R)		
	TcpSyn ----->		
		<-----	TcpSyn, Ack
	TcpAck ----->		
4)	----- TLS Session -----		
	ClientHello ----->		
			ServerHello
			Certificate*
			ServerKeyExchange*
		<-----	ServerHelloDone

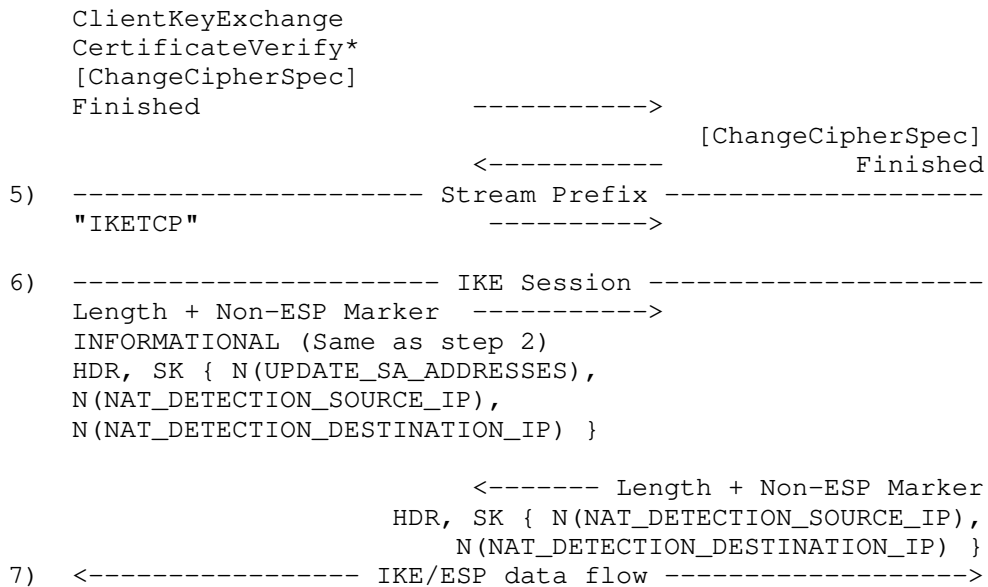


Figure 8

1. During the IKE\_SA\_INIT exchange, the client and server exchange MOBIKE\_SUPPORTED notify payloads to indicate support for MOBIKE.
2. The client changes its point of attachment to the network, and receives a new IP address. The client attempts to re-establish the IKE session using the UPDATE\_SA\_ADDRESSES notify payload, but the server does not respond because the network blocks UDP traffic.
3. The client brings up a TCP connection to the server in order to use TCP encapsulation.
4. The client initiates and TLS handshake with the server.
5. The client sends the Stream Prefix for TCP encapsulated IKE traffic Section 4.
6. The client sends the UPDATE\_SA\_ADDRESSES notify payload on the TCP encapsulated connection. Note that this IKE message is the same as the one sent over UDP in step 2, and should have the same message ID and contents.
7. The IKE and ESP packet flow can resume.

Authors' Addresses

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

Email: [tpauly@apple.com](mailto:tpauly@apple.com)

Samy Touati  
Ericsson  
2755 Augustine  
Santa Clara, California 95054  
US

Email: [samy.touati@ericsson.com](mailto:samy.touati@ericsson.com)

Ravi Mantha  
Cisco Systems  
SEZ, Embassy Tech Village  
Panathur, Bangalore 560 037  
India

Email: [ramantha@cisco.com](mailto:ramantha@cisco.com)

ipsecme  
Internet-Draft  
Intended status: Standards Track  
Expires: December 31, 2017

D. Migault, Ed.  
Ericsson  
T. Guggemos, Ed.  
LMU Munich  
C. Bormann  
Universitaet Bremen TZI  
June 29, 2017

ESP Header Compression and Diet-ESP  
draft-mglt-ipsecme-diet-esp-04.txt

Abstract

ESP Header Compression (EHC) defines a flexible framework to compress communications protected with IPsec/ESP. Compression and decompression is defined by EHC Rules orchestrated by EHC Strategies.

The document specifies the Diet-ESP EHC Strategy and associated EHC Rules. Diet-ESP compresses up to 32 bytes per packet for traditional IPv6 VPN and up to 66 bytes for IPv6 VPN set over a single TCP or UDP session.

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 31, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of



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

## Table of Contents

1. Requirements notation . . . . .	3
2. Introduction . . . . .	3
3. Terminology . . . . .	3
4. Protocol Overview . . . . .	4
5. Diet-ESP EHC Context . . . . .	5
5.1. Diet-ESP Context Parameters for ESP . . . . .	6
5.2. Diet-ESP Context Parameters for Inner IP . . . . .	6
5.3. Diet-ESP Context Parameters for Transport Protocol . . . . .	8
6. Diet-ESP EHC Rules . . . . .	8
6.1. EHC Rules for ESP . . . . .	10
6.2. EHC Rules for inner IPv4 . . . . .	12
6.3. EHC Rules for inner IPv6 . . . . .	14
6.4. EHC Rules for UDP . . . . .	16
6.5. EHC Rules for UDP-Lite . . . . .	17
6.6. EHC Rules for TCP . . . . .	18
7. Diet-ESP EHC Strategy . . . . .	19
7.1. Outbound Packet Processing . . . . .	19
7.2. Inbound Packet Processing . . . . .	21
8. IANA Considerations . . . . .	24
9. Security Considerations . . . . .	24
10. Privacy Considerations . . . . .	25
11. Acknowledgment . . . . .	25
12. References . . . . .	26
12.1. Normative References . . . . .	26
12.2. Informational References . . . . .	26
Appendix A. Illustrative Examples . . . . .	27
A.1. Single UDP Session IoT VPN . . . . .	27
A.2. Single TCP session IoT VPN . . . . .	30
A.3. Traditional VPN . . . . .	33
Appendix B. EHC Classification (Informative) . . . . .	40
B.1. ESP . . . . .	41
B.2. IPv6 (Inner) . . . . .	42
B.3. IPv4 (Inner) . . . . .	43
B.4. UDP . . . . .	45
B.5. UDP-Lite . . . . .	45
B.6. TCP . . . . .	46
Appendix C. Document Change Log . . . . .	47
Authors' Addresses . . . . .	47

## 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

IPsec/ESP [RFC4303] secures communications either using end-to-end security or by building a VPN where the traffic is carried to a secure domain via the security gateway.

IPsec/ESP was not designed to reduce the networking overhead of the communications. In fact, reducing bandwidth often adds computational overhead that may negatively impact large infrastructures in which bandwidth usage is not a constraint. On the other hand, IoT devices have completely different constraints. In IoT communications, sending any extra bytes can significantly impact the battery life of devices. These devices are also often expected to be sleeping nodes, for which IPsec sessions have a very different meaning.

This document defines ESP Header Compression (EHC), a framework that compresses ESP protected communications. EHC is highly flexible to address any use case where compression is necessary. EHC takes advantage of the negotiation between the communication endpoint to agree on the cryptographic parameters. In some cases, the agreement already includes parameters that remain constant during the communications (like port value, or IP address value). EHC takes advantage of these already agreed parameters, and defines additional parameters that could be agreed for the purpose of compression. Similarly, EHC also defines EHC Rules which define how fields may be compressed and decompressed given the provided parameters. Finally, EHC defines EHC Strategy which defines how a set of EHC Rule is coordinated.

The document specifies the Diet-ESP EHC Strategy and associated EHC Rules. Diet-ESP compresses up to 32 bytes per packet for traditional VPN and up to 66 bytes for VPN set over a single TCP or UDP session.

## 3. Terminology

This document uses the following terminology:

- IoT     Internet of Things
- IP     If not stated otherwise, IP means IPv6.
- LSB    Least Significant Bytes
- MSB    Most Significant Bytes
- SAD    IPsec Security Association Database

- SA IPsec Security Association
- SPD IPsec Security Policy Database
- TS IPsec Traffic Selector
- SPI ESP Security Parameter Index
- SN ESP Sequence Number
- PAD ESP Padding
- PL ESP Pad Length
- NH Next Header
- IV Initialization Vector
- IIV Implicit Initialization Vector
- ICV Integrity Check Value
- VPN Virtual Private Network

#### 4. Protocol Overview

ESP Header Compression (EHC) compresses IPsec ESP packets, thus reducing the size of the packet sent on the wire, while carrying an equivalent level of information with an equivalent level of security.

The primarily motivation for payload size reduction was IoT were the cost of sending extra bytes largely overcomes additional computations and thus considerably reduces the life time of battery powered devices. As a result, IoT communication rather favor expensive compression over additional bandwidth. Standard IPsec VPN may also consider reduction of their bandwidth, but on the other hand, the acceptable computation overhead must remain very low. The ESP Header Compression designated in this document as Diet-ESP attempts to reach theses two goals.

ESP Header Compression compresses the standard ESP payload by compressing different fields with a specific compression rules performed in the ESP stack. Concerned fields include fields of the ESP protocol, as well as other protocols in the ESP payload such as the IP header when the tunnel mode is used, the UDP or the TCP header. In fact non ESP fields may be compressed by ESP under certain circumstances, and ESP Header Compression is not intended to provide a generic way, outside of ESP to compress these protocols. Further compression of the ESP payload may be performed by generic mechanism and outside ESP with more generic mechanisms such as for example ROHCoverIPsec [RFC5858] or SCHC [I-D.toutain-6lpwa-ipv6-static-context-hc] which are orthogonal to ESP Header Compression.

As depicted in Figure 1, in order to compress the ESP packets, the two peers are expected to agree on the EHC Strategy - Diet-ESP in our case - as well as some extra parameters needed to derive the EHC Rules and EHC Context.

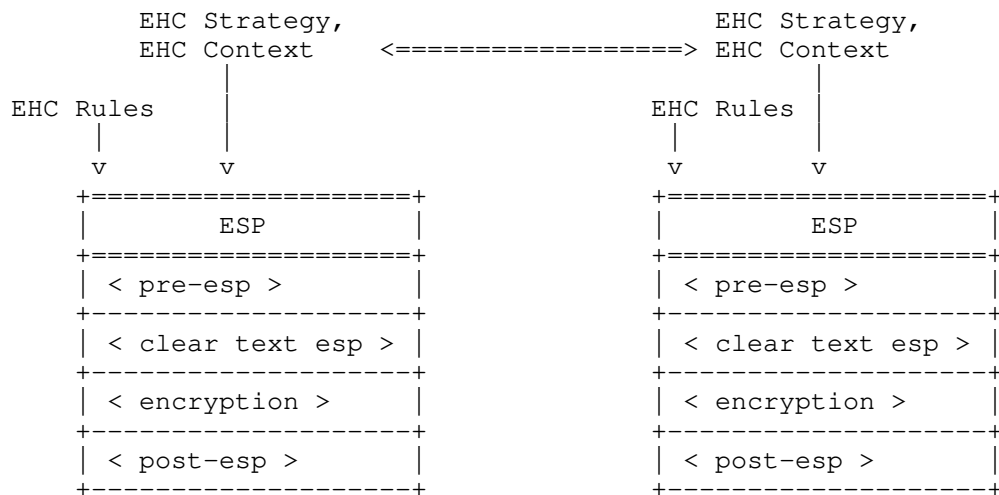


Figure 1: ESP Header Compression Overview

In Figure 1, the ESP stack is represented by various sub layers describing the packet processing inside the ESP. The "pre-esp" layer represents treatment performed to a non ESP packet, i.e. before ESP encapsulation or decapsulation is being proceeded. "clear text esp" designates the ESP encapsulation / decapsulation processing performed on an non encrypted ESP packet. "encryption" designates the encryption/decryption phase and "post-esp" the processing performed on an ESP encrypted packet. EHC Rules may be processed at any of these layers - except for "encryption" layer, and thus impact differently the standard ESP. More specifically, EHC Rules performed at the "pre-esp" or "post-esp" layer does not require the current ESP stack to be updated and can simply be appended to the current ESP stack. On the other hand, EHC Rules at the "clear text esp" may require modification of the current ESP stack.

The set of EHC rules described in this document as well as the EHC Strategies may be extended in the future. There is nothing to prevent such EHC Rules and Strategies to be updated.

## 5. Diet-ESP EHC Context

The EHC Context provides the necessary information so the two peers can proceed to the appropriated compression and decompression defined by the EHC Strategy. As this document is limited to the Diet-ESP strategy, the EHC Context in this section is also designated as Diet-ESP Context and is used by the Diet-ESP Strategy to activate specific EHC Rules as well as to execute the EHC Rule by providing the necessary parameters..

The Diet-ESP Context is defined on a per-SA basis. It is composed of attributes that are not Diet-ESP specific, as well as attributes that are Diet-ESP specific. Attributes that are not Diet-ESP specific are already stored in some form in the SA. Such attributes are designated by "Yes" in the "In SA" column. Diet-ESP specific attributes may need to be specified so Diet-ESP can be executed properly.

#### 5.1. Diet-ESP Context Parameters for ESP

Context Attribute	In SA	Possible Values
esp_mode	Yes	"Tunnel" or "Transport"
outer_version	Yes	"IPv4" "IPv6"
esp_spi	Yes	ESP SPI
esp_spi_lsb	No	0, 1, 2, 3, 4
esp_sn	Yes	ESP Sequence Number
esp_sn_lsb	No	0, 1, 2, 3, 4
esp_sn_gen	No	"Time", "Incremental"
esp_align	No	8, 16, 24, 32
esp_encr	Yes	ESP Encryption Algorithm

#### 5.2. Diet-ESP Context Parameters for Inner IP

Parameters associated to the Inner IP addresses are only specified when the SA has been configured with the tunnel mode. As a result when esp\_mode is set to "Transport" the parameters below MUST NOT be considered and are considered as "Undefined"

Context Attribute	In SA	Possible Values
ip_version	Yes	"IPv4" "IPv6"

##### 5.2.1. Diet-ESP Context Parameters for inner IPv6

Context Attribute	In SA	Possible Values
ip6_tcfl_comp	No	"Outer", "Value", "UnComp"
ip6_tc	No	IPv6 Traffic Class
ip6_fl	No	IPv6 Flow Label
ip6_hl_comp	No	"Outer", "Value", "UnComp"
ip6_hl	No	Hop Limit Value
ip6_src	Yes	IPv6 Source Address
ip6_dst	Yes	IPv6 Destination Address

ip6\_tcfl\_comp indicates how Traffic Class and Flow Label fields of the inner IP Header are expected to be compressed. When set to "UnComp", or "Outer", values associated to ip6\_tc and ip6\_fl MUST NOT be considered and are considered as "Undefined". Values associated to ip6\_tc and ip6\_fl are only considered when ip6\_tcfl\_comp is set to "Provide Values".

ip6\_hl\_comp indicates how Hop Limit field of the inner IP Header is not expected to be compressed. When set to "Outer" or "UnComp", values associated to ip6\_hl MUST NOT be considered and is considered as "Undefined". ip6\_hl is only considered when ip6\_hl\_comp is set to "Value".

ip6\_dst designates the Destination IPv6 Address of the inner IP header. The IP address is provided by the TS, and can be defined as a range of IP addresses. Compression is only considered when ip6\_dst indicates a single IP Address. When the TS defines more than a single IP address ip6\_dst is considered as "Unspecified" and its value MUST NOT be considered for compression.

#### 5.2.2. Diet-ESP Context Parameters for inner IPv4

Context Attribute	In SA	Possible Values
ip4_options	No	"Options", "No_Options"
ip4_id	No	IPv4 Identification
ip4_id_lsb	No	0,1,2
ip4_ttl_compression	No	"Outer", "Value", "UnComp"
ip4_ttl	No	IPv4 Time To Live
ip4_src	yes	IPv4 Source Address
ip4_dst	yes	IPv4 Destination Address
ip4_frag_enable	No	"True", "False"

### 5.3. Diet-ESP Context Parameters for Transport Protocol

The following parameters are provided by the SA but the SA may specify single value or a range of values. When the SA specifies a range of values, these parameters MUST NOT be considered and are considered as Unspecified.

Context Attribute	In SA	Possible Values
l4_proto	Yes	IPv6/ESP Next Header, IPv4 Protocol
l4_src	Yes	UDP/UDP-Lite/TCP Source Port
l4_dst	Yes	UDP/UDP-Lite/TCP Destination Port

#### 5.3.1. Diet-ESP Context Parameters for UDP-Lite

Context Attribute	In SA	Possible Values
udplite_coverage	No	8-16535, "Length", "UnComp"

#### 5.3.2. Diet-ESP Context Parameters for TCP

Context Attribute	In SA	Possible Values
tcp_sn	No	TCP Sequence Number
tcp_ack	No	TCP Acknowledgment Number
tcp_lsb	No	0, 1, 2, 3, 4
tcp_options	No	"True" "False"
tcp_urgent	No	"True" "False"

## 6. Diet-ESP EHC Rules

This section describes the EHC Rules involved in Diet-ESP. The EHC Rules defined by Diet-ESP may be used in the future by EHC Strategies other than Diet-ESP, so they are described in an independent way.

EHC Rule defines the compression and decompression of one or more fields and EHC Rules are represented this way:

EHC Rule	Field	Action	Parameters
EHC_RULE_NAME	f1	a1	p1_1, ... p1_n
	...	...	...
	fm	am	pm_1, ... pm_n

Figure 2: EHC Rules

The EHC Rule is designated by a name (EHC\_RULE\_NAME) which indicates the concerned Fields (f1, ..., fm). Each field compression and decompression is represented by an Action (a1, ..., am). Parameters indicates the necessary parameters for the action to be completed, i.e, to perform both the compression and the decompression.

The table below provides a high level description of the Actions used by Diet-ESP. As these Action may take different arguments and may operate differently for each field a complete description is provided in the next sections as part of the EHC Rule description.

Function	Compression	Decompression
send-value	No	No
elided	Not send	Get from EHC Context
lsb(_lsb_size)	Sent LSB	Get from EHC Context
lower	Not send	Get from lower layer
checksum	Not send	Compute checksum.
padding(_align)	Compute padding	Get padding

- send-value designates an action that does not perform any compression or decompression of a field.
- elided designates an action where both peers have a local value of the field. The compression of the field consists in removing the field, and the decompression consists in retrieving the field value from a known local value. The local value may be stored in a EHC Context or defined by the EHC Rule (like a zero value for example).
- lsb designates an action where both peers have a local value of the field, but the compression consists in sending only the LSB bytes instead of the whole field. The decompression consists in retrieving the field from the LSB sent as well as some other additional local values.



- d. lower designates an action where the compression consists in not sending the field. The decompression consists in retrieving the field from the lower layers of the packet. A typical example is when both IP and UDP carry the length of the payload, then the length of the UDP payload can be inferred from the one of the IP layer.
- e. checksum designates an action where the compression consists in not sending a checksum field. The decompression consists in re-computing the checksum. ESP provides an integrity-check based on signature of the ESP payload (ICV). This makes removing checksum possible, without harming the checksum mechanism.
- f. padding designates an action that computes the padding of the ESP packet. The function is specific to the ESP.

For all actions, the function can be performed only when the appropriated parameters and fields are provided. When a field or a parameters does not have an appropriated value its value is designated as "Unspecified". Specifically some fields such as inner IP addresses, ports or transport protocols are agreed during the SA negotiation and are specified by the SA. Their value in the SA may take various values that are not appropriated to enable a compression. For example, when these fields are defined as a range of values, or by selectors such as OPAQUE or ANY these fields cannot be retrieved from a local value. Instead, when they are defined as a "Single" value (i.e a single IP address, or a single port number or a single transport protocol number) compression and decompression can be performed. These SA related fields are considered as "Unspecified" when not limited to a "Single" value.

When a field or a parameter is "Unspecified", the EHC Rule MUST NOT be activated. This is the purpose of the EHC Strategy to avoid ending in such case. In any case, when one of these condition is not met, the EHC Rule MUST NOT perform any compression or decompression action and the packet MUST be discarded. When possible, an error SHOULD be raised and logged.

#### 6.1. EHC Rules for ESP

This section describes the EHC Rules for ESP which are summed up in the table below.

EHC Rule	Field	Action	Parameters
ESP_SPI	SPI	lsb	esp_spi_lsb, esp_spi
ESP_SN	Sequence Number	lsb	esp_sn_lsb, esp_sn_gen, esp_sn
ESP_NH	Next Header	elided	l4_proto, ipsec_mode
ESP_PAD	Pad Length, Padding	padding	esp_align, esp_encr

ESP\_SPI designates the EHC Rule compressing / decompressing the SPI. ESP\_SPI is performed in the "post-esp" phase. The SPI is compressed using "lsb". The sending peer only places the LSB bytes of the SPI and the receiving peer retrieve the SPI from the LSB bytes carried in the packets as well as from the SPI value stored in the SA. The SPI MUST be retrieved as its full value is included in the signature check. The two peers MUST agree on the number of LSB bytes to be sent: "esp\_spi\_lsb". Upon agreeing on "esp\_spi\_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full SPI.

ESP\_SN designates the EHC Rule compressing / decompressing the ESP Sequence Number. ESP\_SN is performed in the "post-esp" phase. ESP\_SN is only activated if the SN ("esp\_sn"), the LSB significant bytes ("esp\_sn\_lsb") and the method used to generate the SN ("esp\_sn\_gen") are defined. The Sequence Number is compressed using "lsb". Similarly to the SPI, the Sequence Number MUST be retrieved in order to complete the signature check of the ESP packet. Unlike the SPI, the Sequence Number is not agreed by the peers, but is changing for every packet. As a result, in order to retrieve the Sequence Number from the LSB "esp\_sn\_lsb", the peers MUST agree on generating Sequence Number in a similar way. This is negotiated with "esp\_sn\_gen" and the receiver MUST ensure that "esp\_sn\_lsb" is big enough to absorb minor packet losses or time differences between the peers.

ESP\_NH designates the EHC Rule compressing / decompressing the ESP Next Header. ESP\_NH is performed in the "clear text esp" phase. ESP\_NH is only activated if the Next Header is specified. The Next Header can be specified as IP (IPv4 or IPv6) when the IPsec tunnel mode is used ("esp\_mode" set to "Tunnel") or when the transport mode is used when the Traffic Selectors defines a "Single" Protocol ID ("l4\_proto"). The Next Header, is compressed using "elided". The Next Header indicates the Header in the Payload Data. When the Tunnel mode is chosen, the type of the header is known to be an IP header. Similarly, the TS may also hold transport layer protocol,

which specifies the Next Header value for Transport mode. The Next Header value is only there to provide sufficient information for decapsulating ESP. In other words decompressing this fields would occur in the "clear text esp" phase and striped but directly removed again by the ESP stack. For these reasons, implementation may simply omit decompressing this field.

ESP\_PAD designates the EHC Rule compressing / decompressing the Pad Length and Padding fields. ESP\_PAD is performed in the "clear text esp" phase. Pad Length and Padding define the padding. The purpose of padding is to respect a 32 bit alignment for ESP or block sizes of the used cryptographic suite. As the ESP trailer is encrypted, Padding and Pad Length MUST to be performed by ESP and not by the encryption algorithm. Thus, ESP\_PAD always needs to respect the cipher alignment ("esp\_encr"), if applicable. Compression may be performed especially when device support alignment smaller than 32 bit. Such alignment is designated as "esp\_align" and the padding bytes are the necessary bytes so the ESP packet has a length that is a multiple of "esp\_align".

When "esp\_align" is set to an 8-bit alignment padding bytes are not necessary, and Padding as well as Pad Length are removed. For values that are different from 8-bit alignment, padding bytes needs to be computed according to the ESP packet length why ESP\_PAD MUST be the last action of "clear text esp". The resulting number of padding byte is then expressed in Padding and Pad Length fields with Pad Length set to padding bytes number - 1 and Padding is generated as described in [RFC4303].

Combining the Pad Length and Padding fields could potentially add an overhead on fixed size padding. In fact some applications may only send the same type of fixed size data, in which case the Pad Length would not be necessary to be specified. However, the only corner case Pad Length fields would actually add an overhead is when padding is expected to be of zero size. In this case, specifying an 8-bit alignment solve this issue.

## 6.2. EHC Rules for inner IPv4

All IPv4 EHC Rules MUST be performed during the "clear text esp" phase. The EHC Rules are only defined for compressing the inner IPv4 header and thus can only be used when the SA is using the Tunnel mode.

EHC Rule	Field	Action	Parameters
IP4_OPT_DIS	Version	elided	ip_version
	Header Length	elided	
IP4_LENGTH	Total Length	lower	
IP4_ID	Identification	lsb	ip4_id, ip4_id_lsb
IP4_FRAG_DIS	Flags	elided	
	Fragment Offset	elided	
IP4_TTL_OUTER	Time To Live	elided	ip4-ttl
IP4_TTL_VALUE	Time To Live	elided	ip4-ttl
IP4_PROT	Protocol	elided	l4_proto
IP4_CHECK	Header Checksum	checksum	
IP4_SRC	Source Address	elided	ipv4-source
IP4_DST	Dest. Address	elided	ipv4-dest

IP4\_OPT\_DIS designates that the IPv4 header does not include any options and indicates if the first byte of the IPv4 header - consisting of IP version and IPv4 Header Length, are compressed. The Version "ip\_version" is defined by the SA and is thus compressed using "elided". The Header Length is static, if the header does not contain any options, thus it is compressed with "elided" and decompressed "20", the default length of the IPv4 header.

IP4\_LENGTH designates the EHC Rule compressing / decompressing the Total Length Field of the inner IPv4 header. The Total Length is compressed by the sender and not sent. The receiver decompresses it by recomputing the Total Length from the outer IP header. The outer IP header can be IPv4 or IPv6 and IP4\_LENGTH MUST support both versions if both versions are supported by the device. Note that the length of the inner IP payload may also be subject to updates if decompression of the upper layers occurs.

IP4\_ID designates the EHC Rule compressing / decompressing the Identification Field. IP4\_ID is only activated if the ID ("ip4\_id"), the LSB significant bytes ("ip4\_id\_lsb") are defined. Upon agreeing on "ip4\_id\_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full IP Identification. Note also that unlike the ESP SN, the IPv4 Identification is not part of the SA. As a result, when the ID is compressed, its value MUST be stored in the EHC Context. The reserved attribute for that is "ip4\_id"

IP4\_FRAG\_DIS designates that the inner IPv4 header does not support fragmentation. If activated, IP4\_FRAG\_DIS indicates compression of Flags and Fragment Offset field in the IPv4 header which consists of 2 bytes. Both fields are compressed with "elided" and decompressed

with their default value according to [RFC0791], which is 0b010 for Flags and 0 for Fragment Offset.

IP4\_TTL\_OUTER designates an EHC Rule compressing / decompressing the Time To Live field of the inner IP header. IP4\_TTL\_OUTER is only activated when both the outer and inner IP header are IPv6 header. The Time To Live field is compressed / decompressed using the "lower". More specifically, the field is not sent. The receiver decompresses them by reading their value from the outer IPv4 header.

IP4\_TTL\_VALUE designates an EHC Rule compressing / decompressing the Time To Live field of the inner IP header. IP4\_TTL\_VALUE is only activated when the Hop Limit ("ip4\_ttl") has been agreed. Time To Live is compressed / decompressed using the "elided" method.

IP4\_PROTO designates the EHC Rule compressing / decompressing the Protocol field of the inner IPv4 header. IP4\_PROTO is only activated if the Protocol is specified, that is when the Traffic Selectors defines a "Single" Protocol ID ("l4\_proto"). When the Protocol ID identified by the SA has a "Single" value, the Protocol is compressed and decompressed using the "elided" method.

IP4\_CHECK designates the EHC rule compressing / decompressing the Header Checksum field of the inner IPv4 header. The IPv4 header checksum is not sent by the sender and the receiver computes from the decompressed inner IPv4 header. IP4\_CHECK MUST compute the checksum and not fill the checksum field with zeros. As a result, IP4\_CHECK is the last decompressing EHC Rule to be performed on the decompressed IPv4 header.

IP4\_SRC compresses the source IP address of the inner IPv4 header. IP4\_SRC\_IP is only be activated when the Traffic Selectors agreed by the SA defines a "Single" source IP address ("ip4\_src"). The Source IP address is compressed / decompressed using the "elided" method.

IP4\_DST works in a similar way as IP4\_SRC\_IP but for the destination IP address ("ip4\_dst")

### 6.3. EHC Rules for inner IPv6

All IPv6 EHC Rules MUST be performed during the "clear text esp" phase. The EHC Rules are only defined for compressing the inner IPv6 header and thus can only be used when the SA is using the Tunnel mode.

EHC Rule	Field	Action	Parameters
IP6_OUTER	Version Traffic Class Flow Label	elided lower lower	ip_version
IP6_VALUE	Version Traffic Class Flow Label	elided elided elided	ip_version ip6_tc ip6_fl
IP6_LENGTH	Payload Length	lower	
IP6_NH	Next Header	elided	l4_proto
IP6_HL_OUTER	Hop Limit	lower	
IP6_HL_VALUE	Hop Limit	elided	ip6_hl
IP6_SRC	Source Address	elided	ip6_source
IP6_DST	Dest. Address	elided	ip6_dest

IP6\_OUTER designates an EHC Rule for compressing / decompressing the first 32 bits of the inner IPv6 header formed by the Version, Traffic Class and Flow Label. IP6\_OUTER is only activated when both the outer and inner IP header are IPv6 header. The Version "ip\_version" is defined by the SA and is thus compressed using "elided". The other parameters Traffic Class and Flow Label are compressed using "lower". More specifically, the fields are not sent. The receiver decompresses them by reading their value from the outer IPv6 header.

IP6\_VALUE designates an EHC Rule for compressing / decompressing the first 32 bits of the inner IPv6 header formed by the Version, Traffic Class and Flow Label. IP6\_VALUE is only activated if the Version of the inner IP header agreed by the SA is set to "Version 6" ("ip\_version" set to "Version 6") and the specific values of the Traffic Class ("ip6\_tc") and the Flow Label ("ip6\_fl") are specified. With IP6\_VALUE all fields are compressed and decompressed using "elided". Version is provided by the SA ("ip\_version") while other fields are explicitly provided (ip6\_tc, ip6\_fl).

IP6\_LENGTH designates the EHC Rule compressing / decompressing the Payload Length Field of the inner IPv6 header. The Payload Length is compressed by the sender and is not sent. The receiver decompress it by recomputing the Payload Length from the outer IP header. The IP header can be IPv4 or IPv6 and IP6\_LENGTH MUST support both versions if both versions are supported by the device. Note that the length of the inner IP payload may also be subject to updates if decompression of the upper layers occurs.

IP6\_NH designates the EHC Rule compressing / decompressing the Next Header field of the inner IPv6 header. IP6\_NH is only activated if the Next Header is specified, that is when the Traffic Selectors

defines a "Single" Protocol ID ("l4\_proto"). When the Protocol ID identified by the SA has a "Single" value, the Next Header is compressed and decompressed using the "elided" method.

IP6\_HL\_OUTER designates an EHC Rule compressing / decompressing the Hop Limit field of the inner IP header. IP6\_HL\_OUTER is only activated when both the outer and inner IP header are IPv6 header. The Hop Limit field is compressed / decompressed using the "lower". More specifically, the fields are not sent. The receiver decompresses them by reading their value from the outer IPv6 header.

IP6\_HL\_VALUE designates an EHC Rule compressing / decompressing the Hop Limit field of the inner IP header. IP6\_HL\_VALUE is only activated when the Hop Limit ("ip6\_hl") has been agreed. The Hop Limit is compressed / decompressed using the "elided" method.

IP6\_SRC compresses the source IP address of the inner IP header. IP6\_SRC\_IP is only be activated when the Traffic Selectors agreed by the SA defines a "Single" source IP address ("ip6\_src"). The Source IP address is compressed / decompressed using the "elided" method.

IP6\_DST works in a similar way as IP6\_SRC\_IP but for the destination IP address ("ip6\_dst")

#### 6.4. EHC Rules for UDP

All UDP EHC Rules MUST be performed during the "pre-esp" phase. The EHC Rules are only defined when the Traffic Selectors agreed during the SA negotiation results in "Single" Protocol ID ("l4\_proto") which is set to UDP (17).

EHC Rule	Field	Action	Parameters
UDP_SRC	Source Port	elided	l4_source
UDP_DST	Dest. Port	elided	l4_dest
UDP_LENGTH	Length	lower	
UDP_CHECK	UDP Checksum	checksum	

UDP\_SRC designates the EHC Rule that compresses / decompresses the UDP Source Port. UDP\_SRC is only activated when the Source Port agreed by the SA negotiation ("l4\_src") is "Single". The Source Port is then compressed / decompressed using the "elided" method.

UDP\_DST works in a similar way as UDP\_SRC but for the Destination Port ("l4\_dst").

UDP\_LENGTH designates the EHC Rule compressing / decompressing the Length Field of the UDP header. The length is compressed by the sender and is not sent. The receiver decompresses it by recomputing the Length from the IP address header. The IP address can be IPv4 or IPv6 and UDP\_LENGTH MUST support both versions if both versions are supported by the device.

UDP\_CHECK designates the EHC Rule compressing / decompressing the UDP Checksum. The UDP Checksum is not sent by the sender and the receiver computes from the decompressed UDP payload. UDP\_CHECK MUST compute the checksum and not fill the checksum field with zeros. As a result, UDP\_CHECK is the last decompressing EHC Rule to be performed on the decompressed UDP Payload.

#### 6.5. EHC Rules for UDP-Lite

All UDP-lite EHC Rules MUST be performed during the "pre-esp" phase. The EHC Rules are only defined when the Traffic Selectors agreed during the SA negotiation results in a "Single" Protocol ID ("l4\_proto") which is set to UDPLite (136).

EHC Rule	Field	Action	Parameters
UDP-LITE_SRC	Source Port	elided	l4_source
UDP-LITE_DST	Dest. Port	elided	l4_dest
UDP-LITE_COVERAGE	Checksum Coverage	elided	udplite_coverage
UDP-LITE_CHECK	UDP-Lite Checksum	checksum	

UDP-LITE\_SRC works similarly to UDP\_SRC

UDP-LITE\_DST works similarly to UDP\_DST

UDP-LITE\_COVERAGE designates the EHC Rule compressing / decompressing the UDP-Lite Coverage field. UDP-LITE\_COVERAGE is only activated when the Coverage ("udplite\_coverage") has been agreed with a valid value. The Coverage is compressed / decompressed using the "elided" method.

UDP-LITE\_CHECK designates the EHC Rule compressing / decompressing the UDP-Lite checksum. UDP-LITE\_CHECK is only activated if the Coverage is defined either elided or sent. UDP-LITE\_CHECK computes the checksum using "checksum" according to the uncompressed UDP packet and the value of the Coverage.



## 6.6. EHC Rules for TCP

All TCP EHC Rules MUST be performed during the "pre-esp" phase. The EHC Rules are only defined when the Traffic Selectors agreed during the SA negotiation results in a "Single" Protocol ID ("l4\_proto") which is set to TCP (6).

EHC Rule	Field	Action	Parameters
TCP_SRC	Source Port	elided	l4_source
TCP_DST	Dest. Port	elided	l4_dest
TCP_SN	Sequence Number	lsb	tcp_sn, tcp_ls
TCP_ACK	Acknowledgment Number	lsb	tcp_ack, tcp_lsb
TCP_OPTIONS	Data Offset Reserved Bits	elided elided	tcp_options
TCP_CHECK	TCP Checksum	checksum	
TCP_URGENT	elided	tcp_urgent	

TCP\_SRC works similarly to UDP\_SRC.

TCP\_DST works similarly to UDP\_DST.

TCP\_SN designates the EHC Rule compressing / decompressing the TCP Sequence Number. TCP\_SN is only activated if the SN ("tcp\_sn") and the LSB significant bytes ("tcp\_lsb") are defined. The TCP SN is compressed using "lsb". The sending peer only places the LSB bytes of the TCP SN ("tcp\_sn") and the receiving peer retrieve the TCP SN from the LSB bytes carried in the packets as well as from the TCP SN value stored in EHC Context ("tcp\_sn"). The two peers MUST agree on the number of LSB bytes to be sent: "tcp\_lsb". Upon agreeing on "tcp\_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full TCP SN. Note also that unlike the ESP SN, the TCP SN is not part of the SA. As a result, when the SN is compressed, the value of the TCP SN MUST be stored in the EHC Context. The reserved attribute for that is "tcp\_sn"

TCP\_ACK designates the EHC Rule compressing / decompressing the TCP Acknowledgment Number and works similarly to TCP SN. Note that "tcp\_lsb" is agreed for both TCP SN and TCP Acknowledgment. Similarly the value of the complete TCP Acknowledgment Number MUST be stored in the "tcp\_ack" attribute of the EHC Context.

TCP\_OPTIONS designates the EHC Rule compressing / decompressing TCP options related fields such as Data Offset and Reserved Bits. TCP\_OPTION can only be activated when the TCP Option ("tcp\_options")

is defined. When "tcp\_options" is set to "False" and indicates there are no TCP Options, the Data Offsets and Reserved Bits are compressed / decompressed using the "elided" method with Data Offset and Reserved Bits set to zero.

TCP\_CHECK designates the EHC Rule compressing / decompressing the TCP Checksum. TCP\_CHECK works similarly as UDP\_CHECK.

TCP\_URGENT designates the EHC Rule compressing / decompressing the urgent related information. When "tcp\_urgent" is set to "False" and indicates there are no TCP Urgent related information, the Urgent Pointer is then "elided" and filled with zeros.

## 7. Diet-ESP EHC Strategy

From the attributes of the EHC Context, Diet-ESP, defines as an EHC Strategy which EHC Rules to apply. The EHC Strategy is defined both for outbound packets which compresses the packet as well as for inbound packet where the decompression occurs.

Implementation may differ from the description below. However, the outcome MUST remain the same.

### 7.1. Outbound Packet Processing

Diet-ESP compression is defined as follows:

1. In phase "pre-esp": Match the inbound packet with the SA and determine if the Diet-ESP EHC Strategy has been activated. If the Diet-ESP HEC Strategy has been activated proceed to next step, otherwise skip all steps associated to Diet-ESP and proceed to the standard ESP as defined in [RFC4303]
2. In phase "pre-esp": If "l4\_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), proceed to the compression of the specific layer. Otherwise, the transport layer is not compressed.
3. In phase "clear text esp": If "esp\_mode" is set to "Tunnel" mode, determine "ip\_version" the IP version of the inner IP addresses and proceed to the appropriated inner IP address compression.
4. In phase "clear text esp" and "post-esp": Proceed to the ESP compression.

UDP compression is defined as below:

1. If the "l4\_src" designates a "Single" Source Port, apply UDP\_SRC to compress the Source Port.
2. If the "l4\_dst" designates a "Single" Destination Port, apply UDP\_DST to compress the Destination Port.

3. Apply UDP\_CHECK to compress the Checksum.
4. Apply UDP\_LENGTH to compress the Length.

UDP-lite compression is defined as below:

1. If the "l4\_src" designates a "Single" Source Port, apply the UDP-LITE\_SRC to compress the Source Port.
2. If the "l4\_dst" designates a "Single" Destination Port, apply the UDP-LITE\_DST, to compress the Destination Port.
3. If "udplite\_coverage" is specified, apply the UDP-LITE\_COVERAGE, to compress the Coverage.
4. Apply UDP-LITE\_CHECK to compress the Checksum.

TCP compression is defined as below:

1. If the "l4\_src" designates a "Single" Source Port than apply the TCP\_SRC to compress the Source Port.
2. If the "l4\_dst" designates a "Single" Destination Port than apply the TCP\_DST to compress the Destination Port.
3. If "tcp\_lsb" is lower than 4, then "tcp\_sn" "tcp\_ack" attributes of the Diet-ESP Context are updated with the value provided from the packet before applying the TCP\_SN and the TCP\_ACK EHC Rules.
4. If "tcp\_options" is set to "False" apply the TCP\_OPTIONS EHC Rule.
5. If "tcp\_urgent" is set to "False" apply the TCP\_URGENT EHC Rule.
6. Apply TCP\_CHECK to compress the Checksum.

Inner IPv6 Header compression is defined as below:

1. If the "ip6\_src" designates a "Single" Source IP address, apply the IP6\_SRC to compress the IPv6 Source Address.
2. If the "ip6\_dst" designates a "Single" Destination IP address, apply the IP6\_DST to decompress the IPv6 Destination Address.
3. Hop Limit compression is performed as follows:
  1. If "outer\_version" is set to "IPv6" and "ip6\_hl\_comp" is set to "Outer" apply IP6\_HL\_OUTER.
  2. If "outer\_version" is set to "IPv4" and "ip6\_hl\_comp" is set to "Outer" raise an error and discard the packet.
  3. If "ip6\_hl\_comp" is set to "Value" apply IP6\_HL\_VALUE.
4. If "l4\_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP6\_NH to compress the Next Header.
5. Apply, IP6\_LENGTH to compress the Length.
6. Version, Traffic Class and Flow Label are compressed as follows:
  1. If "outer\_version" is set to "IPv6" and "ip6\_tcfl\_comp" is set to "Outer" apply IP6\_OUTER.

2. If "outer\_version" is set to "IPv4" and "ip6\_tcfl\_comp" is set to "Outer" raise an error and discard the packet.
3. If "ip6\_tcfl\_comp" is set to "Value" apply IP6\_VALUE.

ESP compression is defined as below:

1. In phase "clear text esp": If "esp\_mode" is set to "Tunnel" or "l4\_proto" is set to a "Single value - eventually different from TCP, UDP or UDP-Lite, apply ESP\_NH, to compress the Next Header.
2. In phase "clear text esp": If "esp\_encr" specify an encryption algorithm that does not provide padding, then apply ESP\_ALIGN to compress the Pad Length and Padding.
3. Proceed to the ESP encryption as defined in [RFC4303].
4. In phase "post-esp": If "esp\_sn\_lsb" is different from 4, then apply ESP\_SN. To compress the ESP SN.
5. In phase "post-esp": If "esp\_spi\_lsb" is different from 4, then apply ESP\_SPI to compress the SPI.

## 7.2. Inbound Packet Processing

Diet-ESP decompression is defined as follows:

1. Match the inbound packet with the SA and determine if the Diet-ESP EHC Strategy has been activated. When Diet-ESP is activated this means that the "esp\_spi\_lsb" are sufficient to index the SA and proceed to next step, otherwise skip all steps associated to Diet-ESP and proceed to the standard ESP as defined in [RFC4303]
2. In phase "clear text esp" and "post-esp": Proceed to the ESP decompression.
3. In phase "clear text esp": If "esp\_mode" is set to "Tunnel" mode, determine "ip\_version" the IP version of the inner IP addresses and proceed to the appropriated inner IP address decompression, except for the computation of the checksums and length.
4. In phase "pre-esp": If "l4\_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), proceed to the decompression of the specific layer, except for the computation of the checksums and length replaced by zero fields.
5. In phase "pre-esp": Proceed to the decompression of the checksums and length.

ESP decompression is defined as follows:

1. In phase "post-esp": If "esp\_spi\_lsb" is different from 4, then apply ESP\_SPI to decompress the SPI.
2. In phase "post-esp": If "esp\_sn\_lsb" is different from 4, then apply ESP\_SN. To decompress the ESP SN.
3. Proceed to the ESP signature validation and decryption as defined in [RFC4303].

4. In phase "clear text esp": If "esp\_mode" is set to "Tunnel" or "l4\_proto" is set to a "Single" value - eventually different from TCP, UDP or UDP-Lite, apply ESP\_NH, to decompress the Next Header.
5. In phase "clear text esp": If "esp\_encr" specify an encryption algorithm that does not provide padding, then apply ESP\_ALIGN to compress the Pad Length and Padding.
6. Extract the ESP Data Payload and apply decompression EHC Rule to the ESP Data Payload.

Inner IPv6 decompression is defined as follows:

1. Version, Traffic Class and Flow Label are decompressed as follows:
  1. If "outer\_version" is set to "IPv6" and "ip6\_tcfl\_comp" is set to "Outer" apply IP6\_OUTER to decompress Version, Traffic Class and Flow Label.
  2. If "outer\_version" is set to "IPv4" and "ip6\_tcfl\_comp" is set to "Outer" raise an error and discard the packet.
  3. If "ip6\_tcfl\_comp" is set to "Value" apply IP6\_VALUE to Version, Traffic Class and Flow Label.
  4. If "ip6\_tcfl\_comp" is set to "UnComp", Version, Traffic Class and Flow Label are already provided in the packet.
2. Set the Length to zero.
3. If "l4\_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP6\_NH to decompress the Next Header.
4. Hop Limit decompression is performed as follows:
  1. If "outer\_version" is set to "IPv6" and "ip6\_hl\_comp" is set to "Outer" apply IP6\_HL\_OUTER to decompress Hop Limit.
  2. If "outer\_version" is set to "IPv4" and "ip6\_hl\_comp" is set to "Outer" raise an error and discard the packet.
  3. If "ip6\_hl\_comp" is set to "Value" apply IP6\_HL\_VALUE to decompress the Hop Limit.
  4. If "ip6\_hl\_comp" is set to "UnComp", Hop Limit is already provided in the packet.
5. If the "ip6\_src" designates a "Single" Source IP address, apply the IP6\_SRC to decompress the IPv6 Source Address.
6. If the "ip6\_dst" designates a "Single" Destination IP address then apply the IP6\_DST to decompress the IPv6 Destination Address.
7. Apply, IP6\_LENGTH to provide the replace the zero length value by its appropriated value. The Length value considers the length provided by the lower layers to which are added the additional bytes due to the decompression, minus the length of the inner IP6 Header. The value computed from the lower layer will have to be overwritten in case further decompression occurs.

UDP decompression is defined as follows:

1. If the "l4\_src" designates a "Single" Source Port, apply UDP\_SRC to decompress the Source Port.
2. If the "l4\_dst" designates a "Single" Destination Port, apply UDP\_DST to decompress the Destination Port.
3. Apply UDP\_LENGTH to compress the Length. The length value is computed from the length provided by the lower layer, with the additional added bytes during the UDP decompression including the length size.
4. Apply UDP\_CHECK to decompress the Checksum.
5. Update the Length of the lower layers:
  1. If "esp\_mode" is set to "Transport" mode, update the Length of the outer IP header (IPv4 or IPv6). The Length is incremented by the number of bytes generated by the decompression of the transport layer.
  2. If "esp\_mode" is set to "Tunnel" mode, update the Length of the inner IP address (IPv4 or IPv6) as well as the outer IP header (IPv4 or IPv6). The Length is incremented by the number of bytes generated by the decompression of the transport layer.

UDP-Lite decompression is defined as follows:

1. If the "l4\_src" designates a "Single" Source Port, apply the UDP-LITE\_SRC to decompress the Source Port.
2. If the "l4\_dst" designates a "Single" Destination Port, apply the UDP-LITE\_DST, to decompress the Destination Port.
3. If "udplite\_coverage" is specified, apply the UDP-LITE\_COVERAGE, to decompress the Coverage.
4. Apply UDP-LITE\_CHECK to compress the Checksum.
5. Update the Length of the lower layers as defined in UDP.

TCP decompression is defined as follows:

1. If the "l4\_src" designates a "Single" Source Port than apply the TCP\_SRC to decompress the Source Port.
2. If the "l4\_dst" designates a "Single" Destination Port than apply the TCP\_DST to decompress the Destination Port.
3. If "tcp\_lsb" is lower than 4, apply TCP\_SN and the TCP\_ACK to decompress the TCP Sequence Number and the TCP Acknowledgment Number.
4. If "tcp\_options" is set to "False" apply TCP\_OPTIONS to decompress Data Offset and Reserved Bits.
5. If "tcp\_urgent" is set to "False" apply the TCP\_URGENT to decompress the Urgent Pointer.
6. Apply TCP\_CHECK to decompress the Checksum.

## 8. IANA Considerations

There are no IANA consideration for this document.

## 9. Security Considerations

This section lists security considerations related to the Diet-ESP protocol.

### Security Parameter Index (SPI):

The Security Parameter Index (SPI) is used by the receiver to index the Security Association that contains appropriated cryptographic material. If the SPI is not found, the packet is rejected as no further checks can be performed. In EHC, the value of the SPI is not reduced, but compressed why the SPI value may not be fully provided between the compressor and the de-compressor. On the other hand, its uncompressed value is provided to the ESP-procession and no weakness is introduced to ESP itself. On an implementation perspective, it is strongly recommended that decompression is deterministic. Compression and decompression adds some additional treatment to the ESP packet, which might be used by an attacker. In order to minimize the load associated to decompression, decompression is expected to be deterministic. The incoming compressed SPI with the associated IP addresses should output a single and unique uncompressed SPI value. If an uncompressed SPI values have to be considered, then the receiver could end in n signature checks which may be used by an attacker for a DoS attack.

### Sequence Number (SN):

The Sequence Number (SN) is used as an anti-replay attack mechanism. Compression and decompression of the SN is already part of the standard ESP namely the Extended Sequence Number (ESN). The SN in a standard ESP packet is 32 bit long, whether EHC enables to reduce it to 0 bytes and the main limitation to the compression a deterministic decompression. SN compression consists in indicating the least significant bits of the uncompressed SN on the wire. The size of the compressed SN must consider the maximum reordering index such that the probability that a later sent packet arrives before an earlier one. In addition the size of SN should also consider maximum consecutive packets lost during transmission. In the case of ESP, this number is set to  $2^{32}$  which is, in most real world case, largely over-provisioned. When the compression of the SN is not appropriately provisioned, the most significant bit value may be de-synchronized between the sending and receiving parties. Although IKEv2 provides some re-synchronization mechanisms, in case of IoT the de-synchronization will most likely result in a renegotiation and thus DoS possibilities. Note that IoT communication may also use

some external parameters, i.e. other than the compressed SN, to define whether a packet be considered or not and eventually derive the SN. One such scenario may be the use of time windows. Suppose a device is expected to send some information every hour or every week. In this case, for example, the SN may be compressed to zero bytes. Instead the SN may be derived by incrementing the SN every hour after the last received valid packet. Considering the time the packet is received make it possible to consider the time derivation of the sensor clock. If TIME is used as the method to generate the SN, the receiver MUST ensure that the `esp_sn_lsb` is big enough to resist time differences between the nodes. Note also that the anti-replay mechanism needs to define the size of the anti-replay window. [RFC4303] provides guidance to set the window size and are similar to those used to define the size of the compressed SN.

## 10. Privacy Considerations

### Security Parameter Index (SPI):

Until Diet-ESP is not deployed outside the scope of IoT and small devices, the use of a compressed SPI may provide an indication that one of the endpoint is a sensor. Such information may be used, for example, to evaluate the number of appliances deployed, or - in addition with other information, such as the time interval, the geographic location - be used to derive the type of data transmitted.

Sequence Number (SN): If incremented for each ESP packet, the SN may leak some information like the amount of transmitted data or the age of the sensor. The age of the sensor may be correlated with the software used and the potential bugs. On the other hand, re-keying will re-initialize the SN, but the cost of a re-keying may not be negligible and thus, frequent re-keying can be considered. In addition to the re-key operation, the SN may be generated in order to reduce the accuracy of the information leaked. In fact, the SN does not have to be incremented by one for each packet it just has to be an increasing function. Using a function such as a TIME may prevent characterizing the age or the use of the sensor. Note that the use of such function may also impact the compression efficiency and result in larger compressed SN.

## 11. Acknowledgment

We thank Orange and Universitee Pierre et Marie Curie for initiating the work on Diet-ESP. We Would like to thank Sylvain Killian for implementing an open source Diet-ESP on Contiki and testing it on the FIT IoT-LAB [fit-iot-lab] funded by the French Ministry of Higher Education and Research. We thank the IoT-Lab Team and the INRIA for



maintaining the FIT IoT-LAB platform and for providing feed backs in an efficient way.

We would like to thank Rob Moskowitz for not copyrighting Diet HIP. The "Diet" terminology is from him.

We would like to thank those we received many useful feed backs among others: Dominique Bartel, Anna Minaburo, Suresh Krishnan, Samita Chakrabarti, Michael Richardson, Tero Kivinen.

## 12. References

### 12.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<http://www.rfc-editor.org/info/rfc791>>.
- [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>>.
- [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>>.
- [RFC5225] Pelletier, G. and K. Sandlund, "RObust Header Compression Version 2 (ROHCv2): Profiles for RTP, UDP, IP, ESP and UDP-Lite", RFC 5225, DOI 10.17487/RFC5225, April 2008, <<http://www.rfc-editor.org/info/rfc5225>>.
- [RFC5858] Ertekin, E., Christou, C., and C. Bormann, "IPsec Extensions to Support Robust Header Compression over IPsec", RFC 5858, DOI 10.17487/RFC5858, May 2010, <<http://www.rfc-editor.org/info/rfc5858>>.

### 12.2. Informational References

- [I-D.toutain-6lpwa-ipv6-static-context-hc] Minaburo, A. and L. Toutain, "6LPWA Static Context Header Compression (SCHC) for IPV6 and UDP", draft-toutain-6lpwa-ipv6-static-context-hc-01 (work in progress), June 2016.

[I-D.mglt-ipsecme-implicit-iv]

Migault, D., Guggemos, T., and Y. Nir, "Implicit IV for Counter-based Ciphers in IPsec", draft-mglt-ipsecme-implicit-iv-04 (work in progress), June 2017.

[fit-iot-lab]

"Future Internet of Things (FIT) IoT-LAB",  
<<https://www.iot-lab.info>>.

## Appendix A. Illustrative Examples

### A.1. Single UDP Session IoT VPN

This section considers a IoT IPv6 probe hosting a UDP application. The probe is dedicated to a single application and establishes a single UDP session. As a result, inner IP addresses and UDP Ports have a "Single" value and can be easily compressed. The probes sets an IPsec VPN using IPv6 addresses in order to connect its secure domain - typically a Home Gateway. The use of IPv6 for inner and outer IP addresses, enables to infer inner IP fields from the outer IP address. The probes encrypts with AES-CCM\_8 [RFC4309]. AES-CCM does not have padding, so the padding is performed by ESP. The probes uses an 8 bit alignment which enables to fully compress the ESP Trailer. In addition, as the probe SA is indexed using the outer IP addresses (or eventually the radio identifiers) which enables to fully compress the SPI. As the probe provides information every hour, the Sequence Number using time can be derived from the received time, which enables to fully compress the SN.

Figure 3 represents the original UDP packet and Figure 4 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 61 bytes overhead. With IPv4 inner IP addressed Diet-ESP results in an 45 byte overhead reduction.

Further compression may be done for example by using an implicit IV [I-D.mglt-ipsecme-implicit-iv] and by compressing the outer IP addresses (not represented) on the figure. In addition, application data may also be compressed with mechanisms outside of the scope of Diet-ESP.

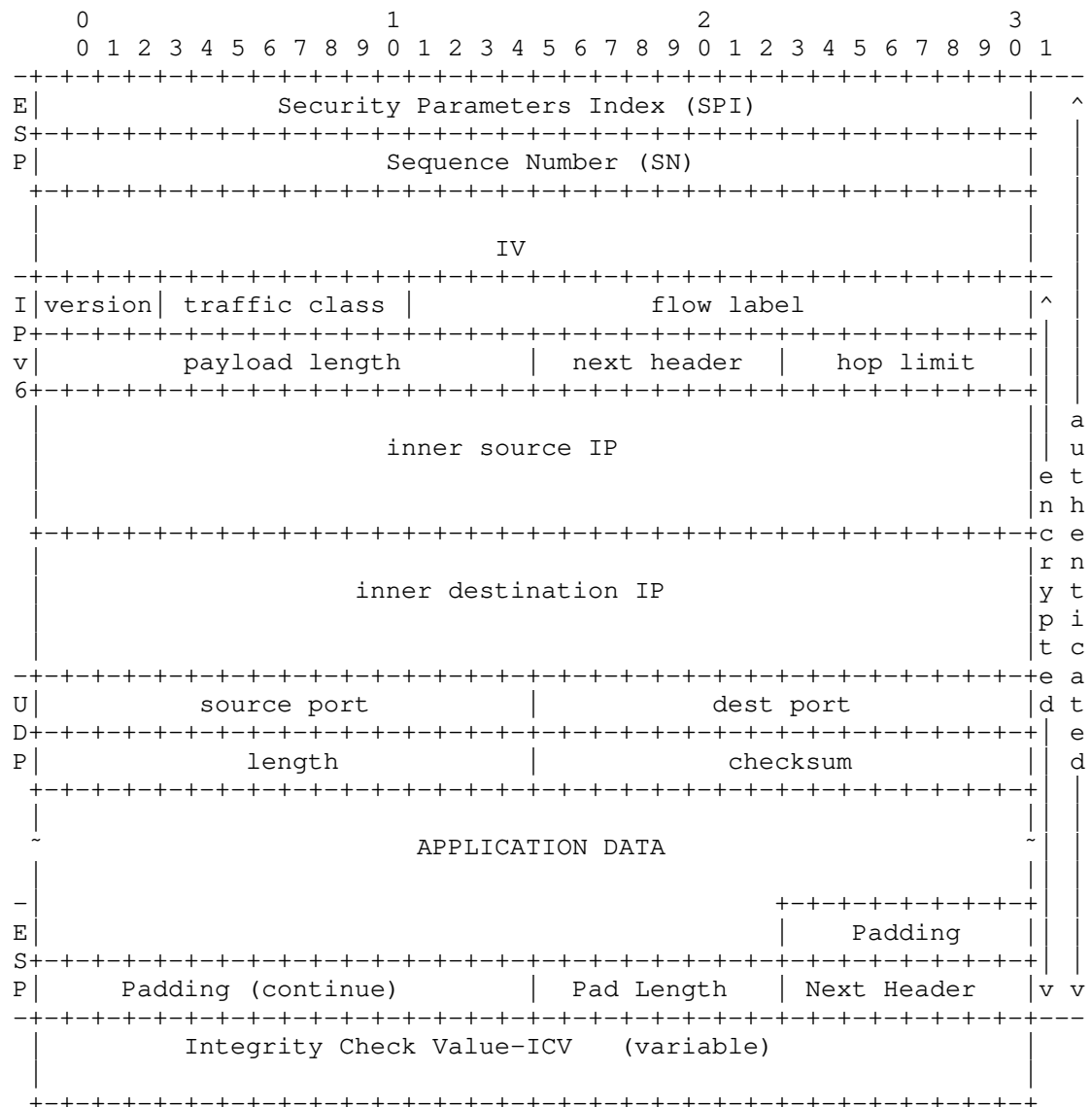


Figure 3: Standard ESP VPN Packet Description

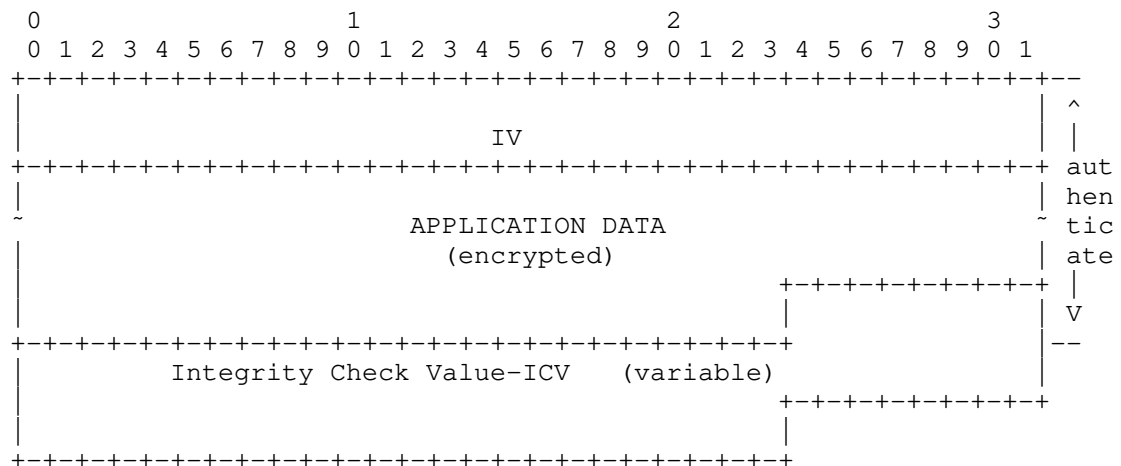


Figure 4: Diet-ESP Single UDP Session IoT VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified".

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	0
ESP_SN	esp_sn_lsb	0
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	"Outer"
	ip6_hl_comp	"Outer"
IP6_LENGTH		
IP6_NH		
IP6_HL_OUTER		
IP6_SRC		
IP6_DST		
UDP_SRC		
UDP_DST		
UDP_LENGTH		
UDP_CHECK		

## A.2. Single TCP session IoT VPN

This section considers the same probe as described in Appendix A.1 but instead of using UDP as a transport layer, the probe uses TCP. In this case TCP is used with no options, no urgent pointers and the SN and ACK Number are compressed to 2 bytes as the throughput is expected to be low.

Figure 5 represents the original TCP packet and Figure 6 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 66 bytes overhead. With IPv4 inner address Diet-ESP results in a 50 byte overhead reduction.

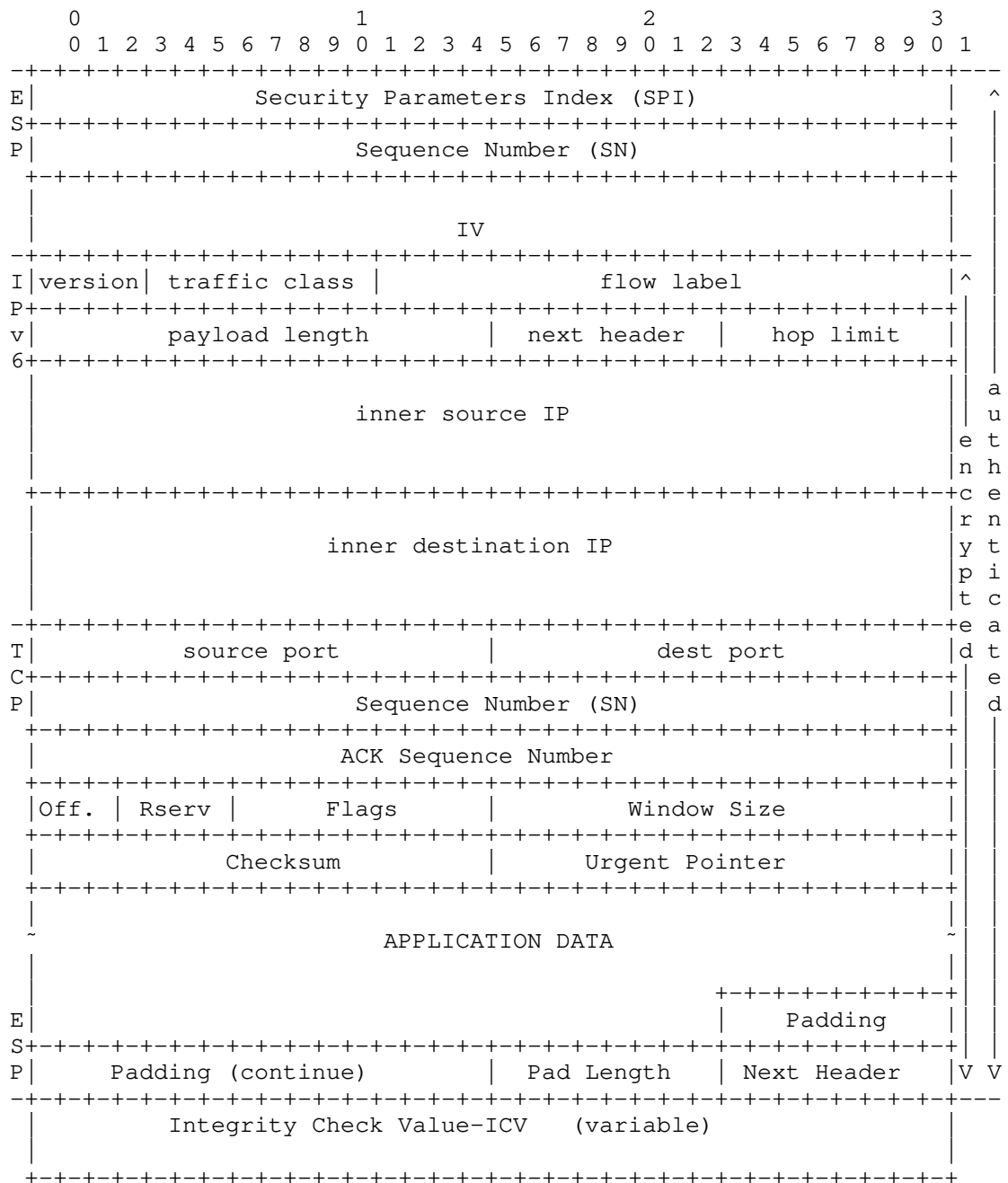


Figure 5: Standard IoT Single TCP Session VPN Packet Description

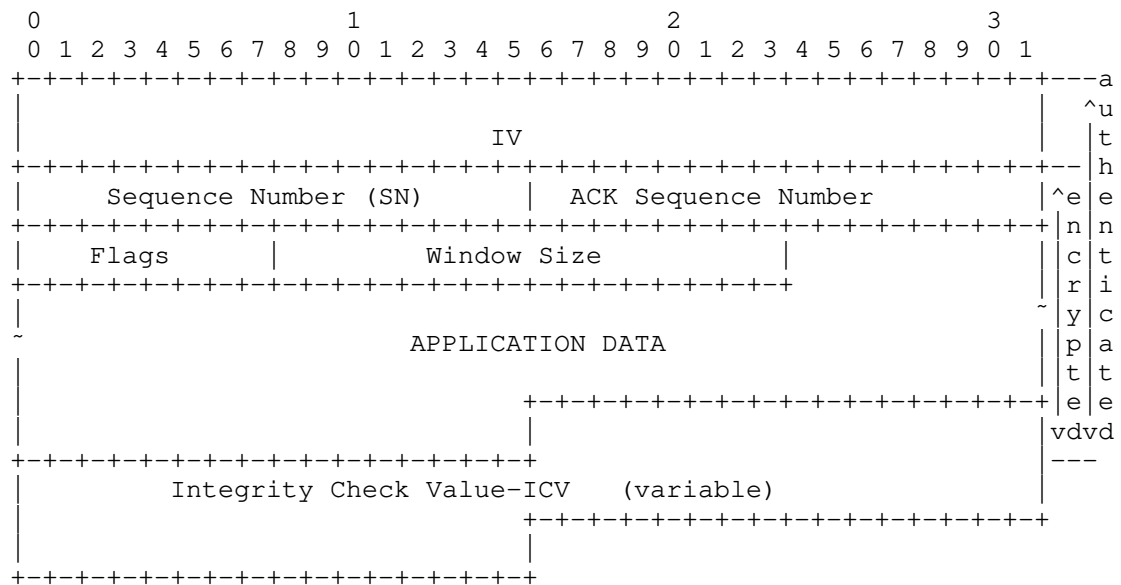


Figure 6: Diet-ESP Single TCP Session IoT VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified". Note for simplicity, tcp\_sn and tcp\_ack are negotiated to start with 0, but it could be any other value as well.

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	0
ESP_SN	esp_sn_lsb	0
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	"Outer"
	ip6_hl_comp	"Outer"
IP6_LENGTH		
IP6_NH		
IP6_HL_OUTER		
IP6_SRC		
IP6_DST		
TCP_SRC		
TCP_DST		
TCP_SN	tcp_lsb	2
	tcp_sn	0
TCP_ACK	tcp_lsb	2
	tcp_ack	0
TCP_OPTIONS	tcp_options	"False"
TCP_CHECK		
TCP_URGENT	tcp_urgent	"False"

### A.3. Traditional VPN

This section illustrates the case of an company VPN. The VPN is typically set by a remote host that forwards all its traffic to the security gateway. As transport protocols are "Unspecified", compression is limited to ESP and the inner IP header. For the inner IP header, the Destination IP address is "Unspecified" so the compression of the inner IP address excludes the Destination IP address. Similarly, the inner IP Next Header cannot be compressed as the transport layer is not specified. For ESP, the security gateway may only have a sufficiently low number of remote users with relatively low throughput in which case SPI and SN can be compressed to 2 bytes. As throughput remains relatively low, the alignment may also set to 8 bits.

#### A.3.1. IPv6 in IPv6

Figure 7 represents the original TCP packet with IPv6 inner IP addresses and Figure 8 represents the corresponding packet compressed



with Diet-ESP. The compression with Diet-ESP results in a reduction of 32 bytes.

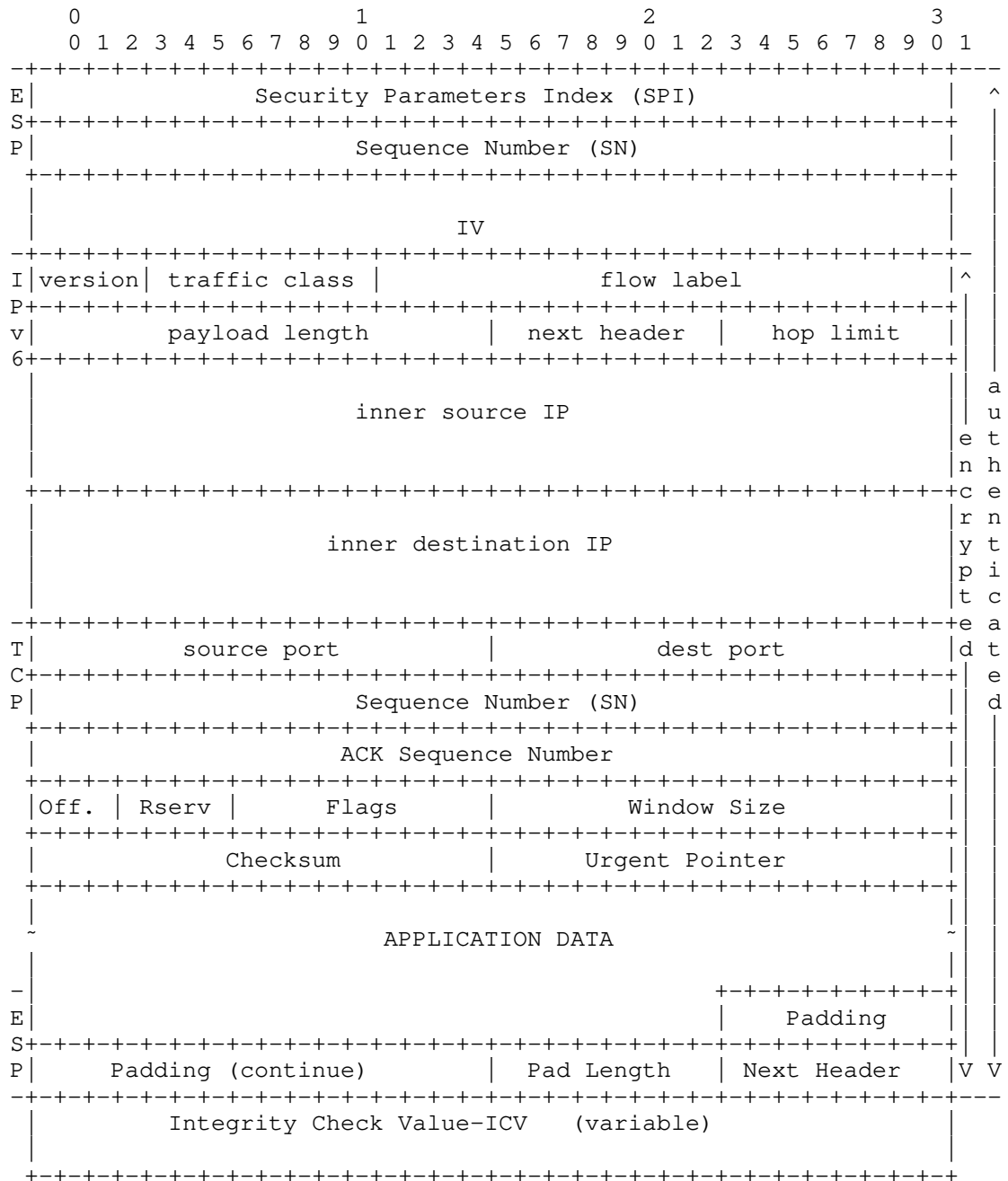


Figure 7: Standard ESP VPN Packet Description

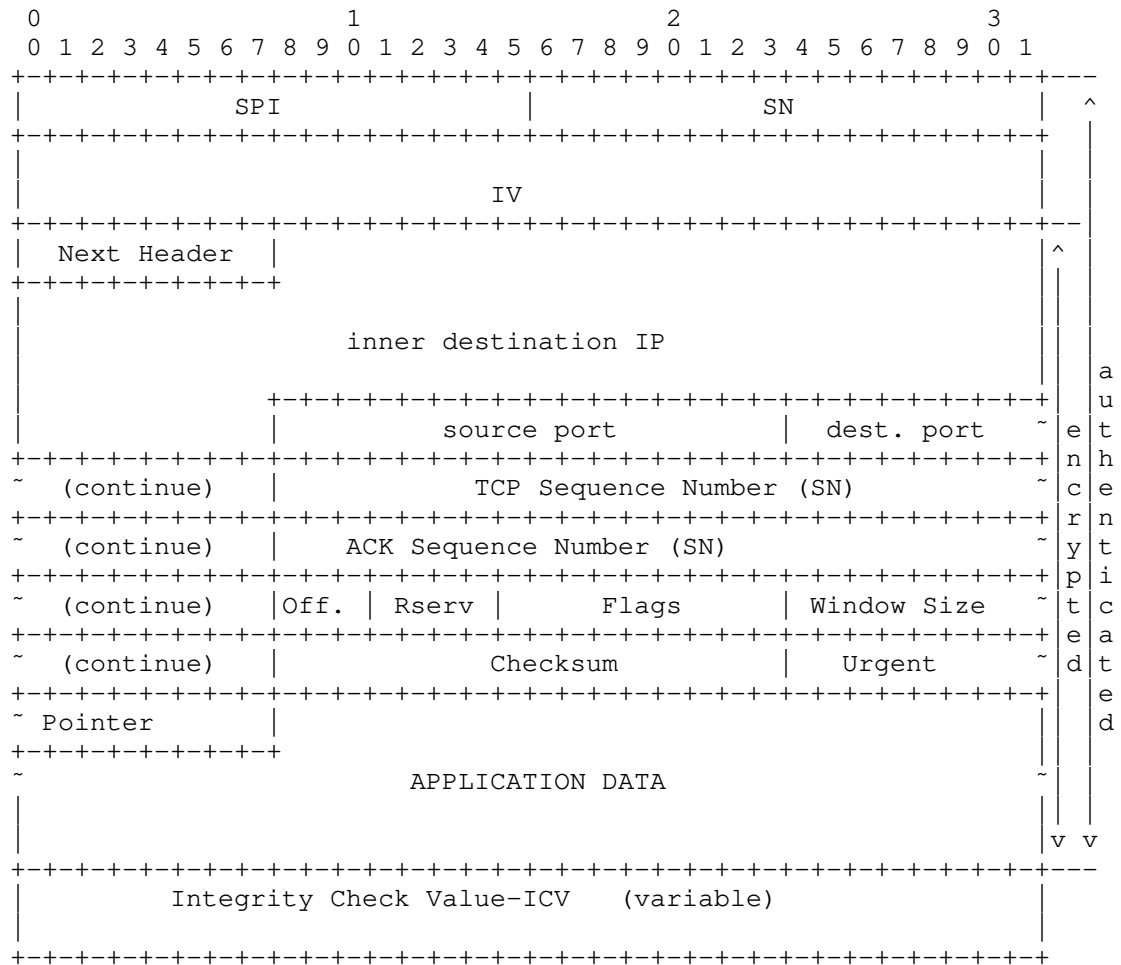


Figure 8: Diet-ESP VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified".

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	"Outer"
IP6_LENGTH		
IP6_HL_OUTER	ip6_hl_comp	"Outer"
IP6_SRC		

#### A.3.2. IPv6 in IPv4

If the compressed inner IP header is an IPv6, but the outer IP header is an IPv4 header, the activated rules differ, as IP6\_OUTER cannot be used. Instead, ip6\_tcfl\_comp and ip6\_hl\_comp are set to "Value". The resulting ESP packet is the same as in Figure 8.

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP6_VALUE	ip6_tcfl_comp	"Value"
	ip_version	6
	ip6_tc	0
	ip6_fl	0
IP6_LENGTH		
IP6_HL_OUTER	ip6_hl_comp	"Value"
	ip6_hl	255
IP6_SRC		

#### A.3.3. IPv4 in IPv4

Figure 9 represents the original TCP packet with IPv6 inner IP addresses and Figure 10 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 24 bytes.

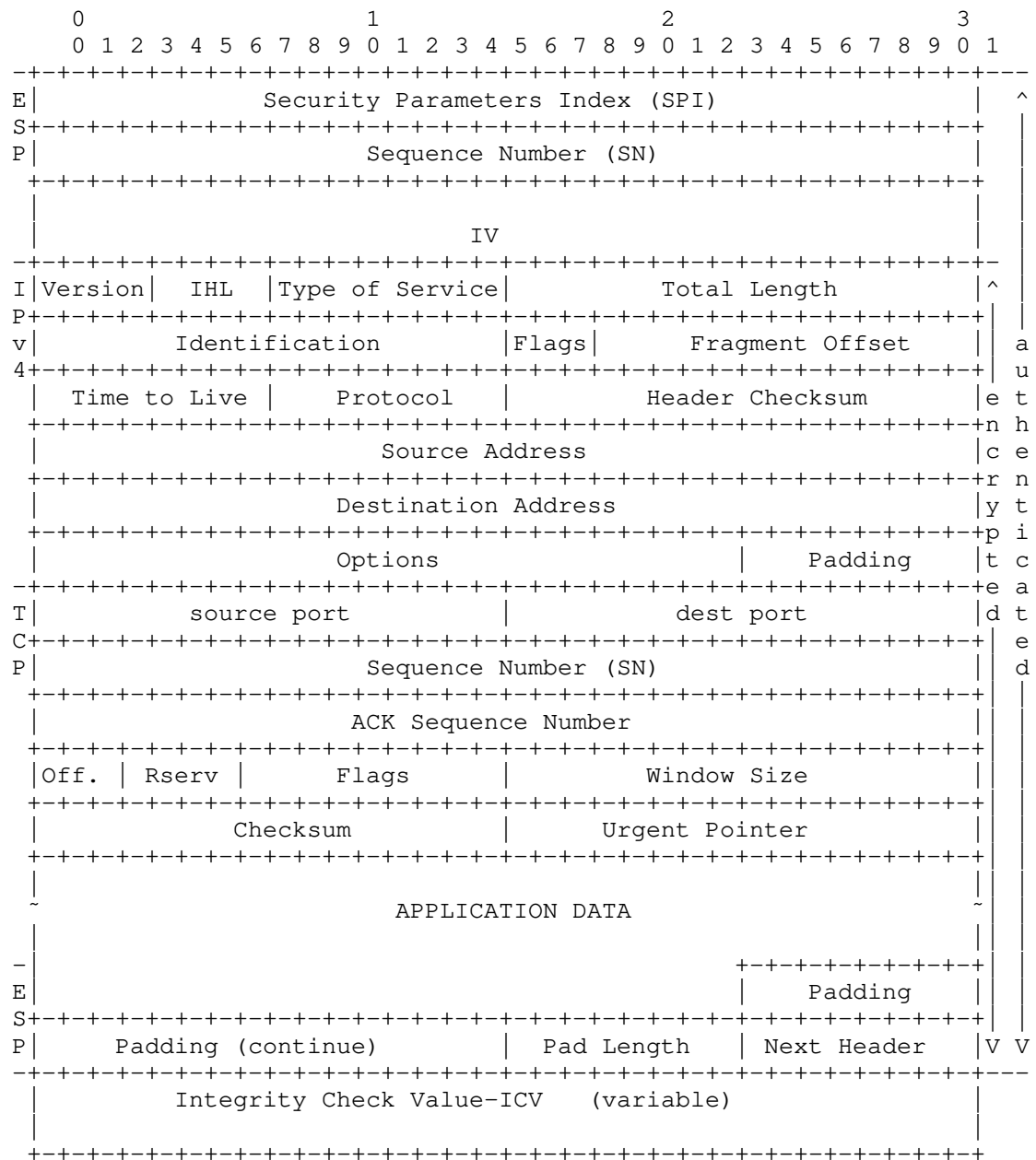


Figure 9: Standard ESP VPN Packet Description

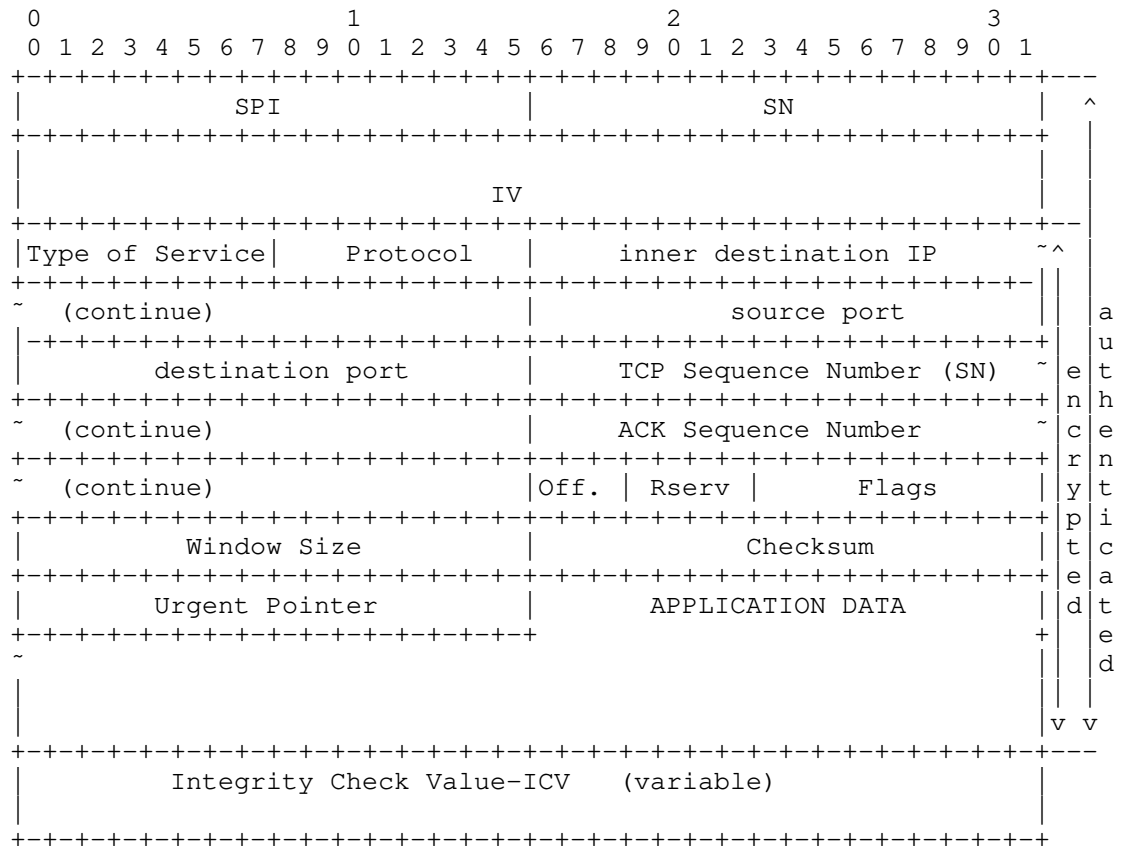


Figure 10: Diet-ESP VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified".

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP4_OPT_DIS		
IP4_LENGTH		
IP4_FRAG_DIS		
IP4_TTL_OUTER		
IP4_CHECK		
IP4_SRC		

#### A.3.4. IPv4 in IPv6

If the compressed inner IP header is an IPv4, but the outer IP header is an IPv6 header, the activated rules differ, as IP4\_TTL\_OUTER cannot be used. Instead, IP4\_TTL\_VALUE is used. The resulting ESP packet is the same as in Figure 10.

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP4_OPT_DIS		
IP4_LENGTH		
IP4_FRAG_DIS		
IP4_TTL_VALUE	ip4-ttl	255
IP4_CHECK		
IP4_SRC		

#### Appendix B. EHC Classification (Informative)

EHC Rules for Header fields depends on the property of the given field. The complete classification for all supported protocols is provided in Appendix B The current classification is based on the classification provided by ROHC (see Appendix A of [RFC5225]). However, as EHC agrees on a context out of band, not all

classifications provided by ROHC are necessary and classification may end up differently.

A key difference between ROHC and ESP Header Compression is that ESP needs an explicit agreement between the peers, whereas ROHC does not proceed to any out-of-band agreement. Instead ROHC learns some values by reading them from an initial packet. This learning phase is not anymore needed with ESP Header Compression as these fields can be agreed. For that reason fields classified as STATIC-DEF by ROHC becomes classified as STATIC-KNOWN for ESP Header Compression. In addition, the EHC classification also differs from the ROHC classification as EHC may be able to segment classes according to the ESP context. In that sense, EHC does not consider a single class - which is the one with the least constraints, as ROHC does. Instead, the EHC classification intends to associate the different classes to the ESP context.

EHC requires these four classes, in order to classify the different header fields:

**STATIC-KNOWN** These fields are expected to remain constant throughout the lifetime of the flow. As a result, they can be negotiated out of band and stored in a context.

**INFERRED** These fields contain values that can be inferred from other values, for example the size of the frame carrying the packet, and thus do not have to be included in compressed packets.

**PATTERN** These are fields that change between each packet, but change in a predictable pattern.

**IRREGULAR** These are the fields for which no useful change pattern can be identified and should be transmitted uncompressed in all compressed packets.

This section details the classification for the currently supported protocol fields. Bbeing ESP encapsulated, a given field may end up with a different classification than without encapsulation. In order to point out these differences, this section also provides for information the classification provided by ROHC. In addition, given the context associated to ESP, a given field may be classified differently. In that case, the multiple classifications are mentioned.

#### B.1. ESP

This section provides a EHC classification for the fields of the ESP protocol.



Field	EHC Class	ROHC class
Security Policy Index	STATIC-KNOWN	STATIC-DEF
Sequence Number	PATTERN	PATTERN
Padding	INFERRED / STATIC-KNOWN	
Pad Length	INFERRED / STATIC-KNOWN	
Next Header	STATIC-KNOWN / IRREGULAR	

Fields Padding, Pad Length, Next Header were not classified by ROHC because, they could not be compressed by either ROHCOVerIPsec or ROHC. ROHCOVerIPsec compresses protocols encapsulated by ESP. These fields are part of ESP and so cannot be compressed by ROHCOVerIPsec. ROHC compresses fields sent on the wire but these fields are encrypted by ESP and so cannot be compressed by ROHC.

In EHC, when present, Padding and Pad Length are INFERRED from the necessary protocol alignment, the cipher alignment and the ESP packet length before the encryption occurs. For packet with fixed size, these values will not changed during the session and as a result, may be classified as STATIC-KNOWN. Similarly, for some specific alignment values, such as 8 bit alignment, these fields may also have fixed values and thus may be classified as STATIC-KNOWN too.

Next Header is STATIC-KNOWN when it is part of the TS, in which case it has been agreed between the peers. Otherwise, NH is IRREGULAR and thus cannot be compressed.

## B.2. IPv6 (Inner)

This section provides a EHC classification for the fields of the IPv6 protocol. The IPv6 address only considers the inner IP header when used in conjunction of the tunnel mode.

Field	EHC Class	ROHC class
Version	STATIC-KNOWN	STATIC-KNOWN
Traffic Class	STATIC-KNOWN / INFERRED / IRREGULAR	RACH
Flow Label	STATIC-KNOWN / INFERRED / IRREGULAR	STATIC-DEF
Payload Length	INFERRED	INFERRED
Next Header	STATIC-KNOWN / IRREGULAR	STATIC-DEF
Hop Limit	STATIC-KNOWN / INFERRED	RACH
Source Address	STATIC-KNOWN	STATIC-DEF
Destination Address	STATIC-KNOWN	STATIC-DEF

Traffic Class, Flow Label, Hop Limit are STATIC-KNOWN when part of the TS, in which case it has been agreed between the peers in the EHC context. Alternatively, the inner IPv6 header is INFERRED from the outer IP header if the outer IP header is an IPv6 header. In any other cases, these fields are IRREGULAR.

Next Header is STATIC-KNOWN when it is part of the TS, in which case it has been agreed between the peers. Otherwise, NH is IRREGULAR and thus cannot be compressed.

### B.3. IPv4 (Inner)

This section provides a EHC classification for the fields of the IPv6 protocol. The IPv6 address only considers the inner IP header when used in conjunction of the tunnel mode.

Field	EHC Class	ROHC class
Version	STATIC-KNOWN	STATIC-KNOWN
Header Length		
. Options enabled	IRREGULAR	RACH
. Options disabled	STATIC-KNOWN	STATIC-KNOWN
Type of Service	STATIC-KNOWN	RACH
Total Length	INFERRED	INFERRED
Identification		
. Sequential	PATTERN	PATTERN
. Seq. swap	PATTERN	PATTERN
. Random	IRREGULAR	IRREGULAR
. Zero	STATIC-KNOWN	STATIC-KNOWN
Flags	STATIC-KNOWN / IRREGULAR	
. Reserved		STATIC-KNOWN
. Don't Fragment		RACH
. More Fragment		STATIC-KNOWN
Fragment offset	STATIC-KNOWN / IRREGULAR	STATIC-KNOWN
Time To Live	STATIC-KNOWN / INFERRED	INFERRED
Protocol	STATIC-KNOWN / IRREGULAR	STATIC-DEF
Header Checksum	INFERRED	INFERRED
Source Address	STATIC-KNOWN	STATIC-DEF
Destination Address	STATIC-KNOWN	STATIC-DEF
Options	IRREGULAR	
Padding	IRREGULAR	

Version, Source and Destination Address and Protocol STATIC-KNOWN when part of the TS, in which case it has been agreed between the peers in the EHC context. Otherwise, they are IRREGULAR and thus cannot be compressed.

Traffic Class, Flow Label, Time To Live and Flags are STATIC-KNOWN if agreed between the two peers. Otherwise the inner IPv4 header may also be inferred from the outer IP header if the outer IP header is an IPv4 header. In any other cases, these fields are IRREGULAR.

Header Length depends on the options, thus when peers agree on disabling options, the Header Length becomes STATIC-KNOWN and IRREGULAR otherwise.

Type of Service (DSCP and ECN) are optional and may be disabled in which case they can be classified as STATIC-KNOWN.

Identification, Flags and Fragment Offset are used to deal with fragmentation. When fragmentation is disabled, these fields may be classified as STATIC-KNOWN. When fragmentation is activated,

Identification may be classified as PATTERN or IRREGULAR. Flags or Fragment offset may be classified as IRREGULAR.

Type of Service, Options and Padding cannot be compressed in a static way without in-band signalling, thus they are classified as IRREGULAR.

#### B.4. UDP

This section provides an EHC classification for the fields of the UDP header.

Field	EHC Class	ROHC class
Source Port	STATIC-KNOWN / IRREGULAR	STATIC-DEF
Destination Port	STATIC-KNOWN / IRREGULAR	STATIC-DEF
Length	INFERRED	INFERRED
Checksum	STATIC-KNOWN / INFERRED	STATIC, IRREGULAR

Source and Destination Port are STATIC-KNOWN when part of the TS, in which case it has been agreed between the peers. Otherwise, they are IRREGULAR and thus cannot be compressed.

When peers have agreed to disable UDP checksum, the checksum is always zero and so its value is STATIC-KNOWN. Otherwise the checksum needs to be computed once the packet has been decompressed. UDP checksum can be computed as ESP provides an integrity check, thus the received packet is believed to be unchanged. Note also that integrity check does not enable error correction which CRC does, but in case of a bit error encryption will fail, thus this case is considered as irrelevant.

#### B.5. UDP-Lite

This section provides an EHC classification for the fields of the UDP-Lite header.

Field	EHC Class	ROHC class
Source Port	STATIC-KNOWN	STATIC-DEF
Destination Port	STATIC-KNOWN	STATIC-DEF
Checksum Coverage	STATIC-KNOWN / INFERRED / IRREGULAR	IRREGULAR
Checksum	INFERRED	IRREGULAR

See Appendix B.4 for classification of Source and Destination Port.

The Checksum Coverage is the part of the UDP-Lite packet, and indicates the number of bytes covered by the checksum. The minimal value is 8, which is the UDP-Lite Header. The maximum value is the size of the UDP-Lite packet, which means that the checksum is the same as in UDP. The Coverage can be agreed to be a fixed value or a value that is function of the total length of the UDP packet. In these cases, Coverage can be classified as STATIC-KNOWN or INFERRED. Otherwise it is classified as IRREGULAR.

In UDP-Lite the checksum cannot be disabled, but its coverage changes. Thus it will never appear as zero, but it can be INFERRED in every case, according to the value of Checksum Coverage.

#### B.6. TCP

This section provides an EHC classification for the fields of the TCP header.

Field	EHC Class	ROHC class RFC4413
Source Port	STATIC-KNOWN	STATIC-DEF
Destination Port	STATIC-KNOWN	STATIC-DEF
Sequence Number	PATTERN	CHANGING
Acknowledgement Number	PATTERN	CHANGING
Data Offset		INFERRED
. Options enabled	IRREGULAR	
. Options disabled	STATIC-KNOWN	
Reserved Bits	IRREGULAR	CHANGING
Flags	IRREGULAR	CHANGING
Window	IRREGULAR	CHANGING
Checksum		
. Disabled	STATIC-KNOWN	STATIC
. Enabled	INFERRED	IRREGULAR
Urgent Pointer	IRREGULAR	CHANGING
Options	IRREGULAR	CHANGING

See Appendix B.4 for classification of Source and Destination Port.

The TCP Sequence and Acknowledgement Number increase in a PATTERN but caused by the different TCP-Flags the increase is not performed in every packet.

Data Offset contains the length of the TCP header including the options. If options are agreed to be disabled it is STATIC-KNOWN.

If Options are enabled it cannot be decompressed without in-band signalling, thus it is classified as IRREGULAR in that case.

See Appendix B.4 for the checksum classification.

Flags, Windows, Urgent Pointer and Options cannot be compressed without in-band signalling, thus classified as IRREGULAR in every case.

#### Appendix C. Document Change Log

[draft-mglt-6lo-diet-esp-00.txt]:  
Changing affiliation

[draft-mglt-6lo-diet-esp-00.txt]:  
Updating references

[draft-mglt-ipsecme-diet-esp-01.txt]:  
Diet ESP described in the ROHC framework  
ESP is not modified.

[draft-mglt-ipsecme-diet-esp-00.txt]:  
NAT consideration added.  
Comparison actualized to new Version of 6LoWPAN ESP.

[draft-mglt-dice-diet-esp-00.txt]: First version published.

#### 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 Munchen, Bavaria  
Germany

Email: guggemos@mnmt-team.org  
URI: <http://www.mnm-team.org/~guggemos>

Carsten Bormann  
Universitaet Bremen TZI  
Postfach 330440  
Bremen D-28359  
Germany

Phone: +49-421-218-63921  
Email: cabo@tzi.org

ipsecme  
Internet-Draft  
Intended status: Standards Track  
Expires: 14 November 2022

D. Migault  
Ericsson  
T. Guggemos  
LMU Munich  
C. Bormann  
Universitaet Bremen TZI  
D. Schinazi  
Google LLC  
13 May 2022

ESP Header Compression and Diet-ESP  
draft-mglt-ipsecme-diet-esp-08

Abstract

With the use of encrypted ESP for secure IP communication, the compression of IP payload is only possible with complex frameworks, such as RObust Header Compression (ROHC). Such frameworks are too complex for numerous use cases and especially for IoT scenarios, which makes IPsec not being used here, although it offers architectural benefits.

ESP Header Compression (EHC) defines a flexible framework to compress communications protected with IPsec/ESP. Compression and decompression is defined by EHC Rules orchestrated by EHC Strategies. The necessary state is hold within the IPsec Security Association and can be negotiated during key agreement, e.g. with IKEv2.

The document specifies the necessary parameters of the EHC Context to allow compression of ESP and the most common included protocols, such as IPv4, IPv6, UDP and TCP and the corresponding EHC Rules. It also defines the Diet-ESP EHC Strategy which compresses up to 32 bytes per packet for traditional IPv6 VPN and up to 66 bytes for IPv6 VPN sent over a single TCP or UDP session.

Status of This Memo

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

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



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

This Internet-Draft will expire on 14 November 2022.

#### Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

#### Table of Contents

1. Requirements notation . . . . .	3
2. Introduction . . . . .	3
3. Terminology . . . . .	4
4. Protocol Overview . . . . .	5
5. IPsec Compression Mode . . . . .	7
6. EHC Context . . . . .	7
6.1. EHC Context Parameters for ESP . . . . .	8
6.2. EHC Context Parameters for Inner IP . . . . .	8
6.3. EHC Context Parameters for Transport Protocol . . . . .	10
7. EHC Rules . . . . .	12
7.1. EHC Rules for ESP . . . . .	14
7.2. EHC Rules for inner IPv4 . . . . .	16
7.3. EHC Rules for inner IPv6 . . . . .	19
7.4. EHC Rules for UDP . . . . .	21
7.5. EHC Rules for UDP-Lite . . . . .	22
7.6. EHC Rules for TCP . . . . .	22
8. Diet-ESP EHC Strategy . . . . .	24
8.1. Outbound Packet Processing . . . . .	28
8.2. Inbound Packet Processing . . . . .	30
9. IANA Considerations . . . . .	32
10. Security Considerations . . . . .	32
11. Privacy Considerations . . . . .	34
12. Acknowledgment . . . . .	34
13. References . . . . .	34
13.1. Normative References . . . . .	34

13.2. Informational References . . . . .	35
Appendix A. Illustrative Examples . . . . .	36
A.1. Single UDP Session IoT VPN . . . . .	36
A.2. Single TCP session IoT VPN . . . . .	39
A.3. Traditional VPN . . . . .	43
Authors' Addresses . . . . .	52

## 1. Requirements notation

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

## 2. Introduction

IPsec/ESP [RFC4303] secures communications either using end-to-end security or by building a VPN, where the traffic is carried to a secure domain via a security gateway.

IPsec/ESP was not designed to minimize its associated networking overhead. In fact, bandwidth optimization often adds computational overhead that may negatively impact large infrastructures in which bandwidth usage is not a constraint. On the other hand, in IoT communications, sending extra bytes can significantly impact the battery life of devices and thus the life time of the device. The document describes a framework that optimizes the networking overhead associated to IPsec/ESP for these devices.

Most compression mechanisms work with dynamic compression contexts. Some mechanisms, such as ROHC [RFC5858], agree and dynamically change the context over a dedicated channel. Others, such as 6LowPAN, send the context together with the actual protocol information in a separate compression header. Those mechanism fail when it comes to compress encrypted payloads as appearing in ESP. This is found to be a major reason, why IPsec and in particular ESP is not widely developed in environments where bandwidth saving is a critical task, such as in IoT scenarios.

ESP Header Compression (EHC) chooses another form of context agreement, which is similar to the one defined by Static Context Header Compression (SCHC). It works with a static compression context agreed for a specific Security Association. The context itself can be negotiated during the key agreement, which allows only minimal the changes to the actual ESP implementation.

EHC itself is defined as a framework that specifically compresses ESP protected communications. EHC is highly flexible to address any use case where compression is necessary. EHC takes advantage of the negotiation between the communication endpoint to agree on the cryptographic parameters, which in some cases already includes parameters that remain constant during the communications (like layer 4 ports, or IP addresses) and can thus be used as part of the compression context. Only additional, EHC specific parameters need to be agreed for the purpose of compression. In addition EHC Rules define how fields may be compressed and decompressed given the provided parameters. Finally, EHC defines EHC Strategy which defines how a set of EHC Rule is coordinated.

This document specifies EHC Context parameters for the most common Layer 3 and 4 protocols and the associated EHC Rules. Additionally, an EHC Strategy called Diet-ESP is defined, which compresses up to 32 bytes per packet for traditional VPN and up to 66 bytes for VPN set over a single TCP or UDP session. Its main purpose is a maximum level of compression with a minimum of additional agreement. This is achieved by defining a default usage of existing IPsec SA parameters wherever possible.

### 3. Terminology

This document uses the following terminology:

- EHC     ESP Header Compression
- IoT     Internet of Things
- IP     If not stated otherwise, IP means IPv6.
- LSB     Least Significant Bytes
- MSB     Most Significant Bytes
- SAD     IPsec Security Association Database
- SA     IPsec Security Association
- SPD     IPsec Security Policy Database
- TS     IPsec Traffic Selector
- SPI     ESP Security Parameter Index
- SN     ESP Sequence Number
- PAD     ESP Padding
- PL     ESP Pad Length
- NH     Next Header
- IV     Initialization Vector
- IIV     Implicit Initialization Vector
- ICV     Integrity Check Value
- VPN     Virtual Private Network

#### 4. Protocol Overview

ESP Header Compression (EHC) compresses IPsec ESP packets, thus reducing the size of the packet sent on the wire, while carrying an equivalent level of information with an equivalent level of security.

EHC is able to compress any protocol encapsulated in ESP and ESP itself. Concerned fields include those of the ESP protocol, as well as other protocols in the ESP payload such as the IP header when the tunnel mode is used, but also upper layer protocols, such as the UDP or the TCP header. Non ESP fields may be compressed by ESP under certain circumstances, but EHC is not intended to provide a generic way outside of ESP to compress these protocols. Compression of the unprotected IP header and the unencrypted ESP header may be performed by mechanism such as 6LoWPAN [RFC4944], SCHC [RFC8724], ROHC [RFC5795] or 6LoWPAN-GHC [RFC7400].

EHC is based on a static compression context, EHC Rules coordinated by an EHC Strategy:

**EHC Context:** Stores the information of a specific header field which can be compressed by EHC. This can be specific header values such as IP addresses or L4 ports do not have to be send on the wire at all, or compression information for fields which can be partially compressed, such as sequence numbers.

**EHC Rules:** Defines how the information of the EHC Context is used to compress a specific field. It defines compression functions, such as "elided", "least significant byte" and others, being applied on the header field.

**EHC Strategy:** Is applied to efficiently coordinate EHC Context and EHC Rules. The EHC Strategy "Diet-ESP" defined in this document utilizes the information in the IPsec SA to pre-define the EHC Context without explicitly exchanging the EHC Context.

As depicted in Figure 1, the EHC Strategy - Diet-ESP in our case - and the EHC Context are agreed upon between the two peers, e.g. during key exchange. The EHC Rules are to be implemented on the peers and do not require further agreement.

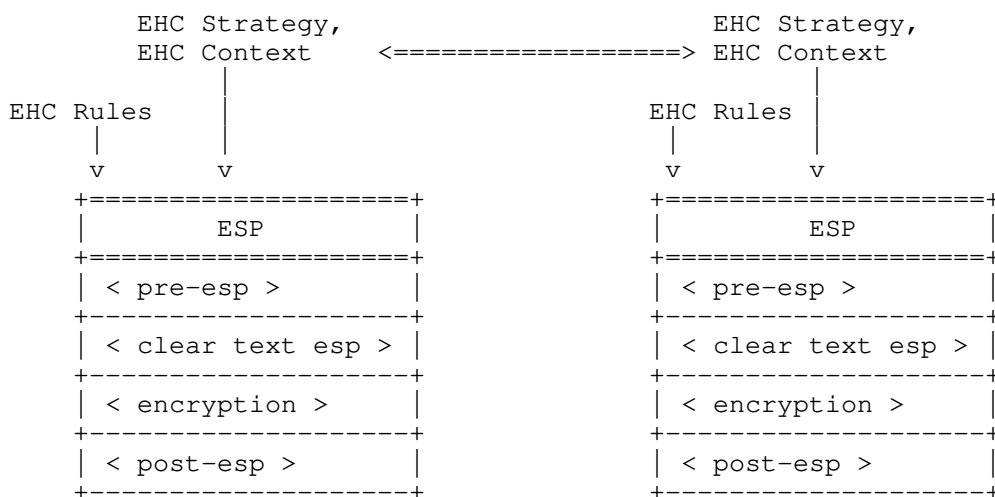


Figure 1: ESP Header Compression Overview

In Figure 1, the ESP stack is represented by various sub layers describing the packet processing inside the ESP:

**pre-esp:** represents treatment performed to a non ESP packet, i.e. before ESP encapsulation or decapsulation is being performed.

Any compression of protocols not specific to but encrypted by ESP, such as L4 and higher protocols, is performed here.

**clear text esp:** designates the ESP encapsulation / decapsulation processing performed on an non encrypted ESP packet. This layer includes compression for fields which are included during the ESP encapsulation. A typical example is the later encrypted Tunnel IP header and the fields of the ESP trailer.

**encryption:** designates the encryption/decryption phase This layer could include compression of encryption information (e.g. Initialization Vector, etc.), but this is currently out of scope of this document.

**post-esp** the processing performed on an ESP encrypted packet. This layer includes compression of the ESP header.

EHC Rules may be processed at any of these layers and thus impact differently the standard ESP. More specifically, EHC Rules performed at the "pre-esp" or "post-esp" layer do not require the current ESP stack to be updated and can simply be appended to the current ESP stack. On the other hand, EHC Rules at the "clear text esp" may require modification of the current ESP stack.

The set of EHC rules described in this document as well as the EHC Strategies may be extended in the future. Nothing prevents such EHC Rules and Strategies to be updated.

## 5. IPsec Compression Mode

Signalling the compression of a certain ESP packet is crucial for correct decompression at the sender. Situation where decompression may fail unforeseen are various, such as IP fragmentation, UDP options [I-D.ietf-tsvwg-udp-options] just to name a few.

With EHC, the agreement of the level or occurrence of compression is left the negotiation protocol (e.g. IKEv2). In order to achieve per-packet signalization of the compression level, this document proposes new IPsec modes "Compressed Transport" and "Compressed Tunnel", which are meant to be agreed during the negotiation of the EHC Context and EHC Strategy. This can lead to multiple SAs, and thus, multiple SPIs for different levels of compression agreed with the EHC Context. The receiver can detect the level of compression of an incoming packet by looking up the used EHC Context and EHC strategy in the corresponding SA.

If the sender detects that the de-compression can not be guaranteed with a given EHC Context and EHC Strategy, it MUST NOT apply compression. If an SA with IPsec Mode "Tunnel"/"Transport" is available, the sender SHOULD send the packet uncompressed, rather than discard the packet. When there is no uncompressed SA available, the packet MUST be dropped.

## 6. EHC Context

The EHC Context provides the necessary information so the two peers can proceed to the appropriated compression and decompression defined by the EHC Strategy.

The EHC Context is defined on a per-SA basis. A context can be defined for any protocol encapsulated with ESP and for ESP itself. For each header field, a context attribute is provided to the EHC Context in order to allow compression and decompression. Most power of EHC lies in the fact, that the attributes for some protocols are already available in the IPsec SA (e.g. IP addresses in the Traffic Selector). Such attributes are designated by "Yes" in the "In SA" column. All others need to be negotiated separately in order to allow EHC to work properly.

As this document is limited to the Diet-ESP strategy, the EHC Context in this section used by the Diet-ESP Strategy to activate specific EHC Rules as well as to execute the EHC Rules by providing the necessary parameters..

#### 6.1. EHC Context Parameters for ESP

Context Attribute	In SA	Possible Values
ipsec_mode	Yes	"Tunnel", "Transport"
outer_version	Yes	"IPv4", "IPv6"
esp_spi	Yes	ESP SPI
esp_spi_lsb	No	0, 1, 2, 3, 4
esp_sn	Yes	ESP Sequence Number
esp_sn_lsb	No	0, 1, 2, 3, 4
esp_sn_gen	No	"Time", "Incremental"
esp_align	No	8, 16, 24, 32
esp_encr	Yes	ESP Encryption Algorithm

Table 1

#### 6.2. EHC Context Parameters for Inner IP

Parameters associated to the Inner IP addresses are only specified when the SA has been configured with the tunnel mode. As a result when ipsec\_mode is set to "Transport" the parameters below MUST NOT be considered and are considered as "Undefined"

Context Attribute	In SA	Possible Values
ip_version	Yes	"IPv4", "IPv6"

Table 2

## 6.2.1. EHC Context Parameters for inner IPv6

Context Attribute	In SA	Possible Values
ip6_tcfl_comp	No	"Outer", "Value", "UnComp"
ip6_tc	No	IPv6 Traffic Class
ip6_fl	No	IPv6 Flow Label
ip6_hl_comp	No	"Outer", "Value", "UnComp"
ip6_hl	No	Hop Limit Value
ip6_src	Yes	IPv6 Source Address
ip6_dst	Yes	IPv6 Destination Address

Table 3

ip6\_tcfl\_comp indicates how Traffic Class and Flow Label fields of the inner IP Header are expected to be compressed. "Outer" indicates Traffic Class and Flow Label are read from the outer IP header, "Value" indicates these values are provided by the Diet-ESP Context, while "Uncompress" indicates that no compression occurs and these values are read in the inner IP inner header.

ip6\_hl\_comp indicates how Hop Limit field of the inner IP Header is expected to be compressed. (see ip6\_tcfl\_comp).

ip6\_dst designates the Destination IPv6 Address of the inner IP header. The IP address is provided by the TS, and can be defined as a range of IP addresses. Compression is only considered when ip6\_dst indicates a single IP Address. When the TS defines more than a single IP address ip6\_dst is considered as "Unspecified" and its value MUST NOT be considered for compression.

## 6.2.2. EHC Context Parameters for inner IPv4

Context Attribute	In SA	Possible Values
ip4_options	No	"Options", "No_Options"
ip4_id	No	IPv4 Identification



ip4_id_lsb	No	0,1,2
ip4_ttl_comp	No	"Outer", "Value", "UnComp"
ip4_ttl	No	IPv4 Time To Live
ip4_src	Yes	IPv4 Source Address
ip4_dst	Yes	IPv4 Destination Address
ip4_frag_enable	No	"True", "False"

Table 4

ip4\_options specifies if the IPv4 header contains any options. If set to "No\_Options", the first 8 bit of the IPv4 header (being the IP version and IP header length) are compressed. If set to "Options" this bits are sent uncompressed.

ip4\_ttl indicates how the Time To Live field of the inner IP Header is expected to be compressed. (see ip6\_hl\_comp).

### 6.3. EHC Context Parameters for Transport Protocol

The following parameters are provided by the SA but the SA may specify single value or a range of values. When the SA specifies a range of values, these parameters MUST NOT be considered and are considered as Unspecified.

Context Attribute	In SA	Possible Values
14_proto	Yes	IPv6/ESP Next Header, IPv4 Protocol
14_src	Yes	UDP/UDP-Lite/TCP Source Port
14_dst	Yes	UDP/UDP-Lite/TCP Destination Port

Table 5

#### 6.3.1. EHC Context Parameters for UDP

For UDP, there are no additional parameters necessary than the ones in Section 6.3

## 6.3.2. EHC Context Parameters for UDP-Lite

Context Attribute	In SA	Possible Values
udplite_coverage	No	8-6535, "Length", "uncompressed"

Table 6

`udplite_coverage`: For UDP-Lite, the checksum can have different coverages, which is defined by the "Checksum Coverage" field which replaces the "Length" field of UDP. This context field defines the coverage in advance by either a specific value (8-16535), the actual length of the UDP-Lite payload ("Length" or 0) or as uncompressed. Note that `udplite_coverage` is indicated on a packet basis and cannot be greater than the UDP length. In this case `udplite_coverage` is negotiated for all packets and the actual coverage for a given UDP packet is derived as the minimum value between `udplite_coverage` and the length of the UDP packet.

## 6.3.3. EHC Context Parameters for TCP

Context Attribute	In SA	Possible Values
<code>tcp_sn</code>	No	TCP Sequence Number
<code>tcp_ack</code>	No	TCP Acknowledgment Number
<code>tcp_lsb</code>	No	0, 1, 2, 3, 4
<code>tcp_options</code>	No	"True", "False"
<code>tcp_urgent</code>	No	"True", "False"

Table 7

`tcp_sn` holds the current Sequence Number of the TCP session.

`tcp_ack` holds the current Acknowledgement Number of the TCP session.

`tcp_lsb` holds the number of lsb of `tcp_sn` and `tcp_ack` sent on the wire.

`tcp_options` says if options are enabled in the current TCP session. If `tcp_options` is set to "False" the Options field in TCP can be elided.

`tcp_urgent` says if the urgent pointer is enabled in the current TCP session. If `tcp_urgent` is set to "False" the Urgent Pointer field in TCP can be elided.

## 7. EHC Rules

This section describes the EHC Rules involved in Diet-ESP. The EHC Rules defined by Diet-ESP may be used in the future by EHC Strategies other than Diet-ESP, so they are described in an independent way.

A EHC Rule defines the compression and decompression of one or more fields and EHC Rules are represented this way:

EHC Rule	Field	Action	Parameters
EHC_RULE_NAME	f1	a1	p1_1, ... p1_n
	~	...	~
	fm	am	pm_1, ... pm_n

Figure 2: EHC Rules

The EHC Rule is designated by a name (`EHC_RULE_NAME`) and the concerned Fields (`f1, ..., fm`). Each field compression and decompression is represented by an Action (`a1, ..., am`). The Parameters indicate the necessary parameters for the action to perform both the compression and the decompression.

The table below provides a high level description of the Actions used by Diet-ESP. As these Action may take different arguments and may operate differently for each field a complete description is provided in the next sections as part of the EHC Rule description.

Function	Compression	Decompression
send-value	No	No
elided	Not send	Get from EHC Context
lsb(_lsb_size)	Sent LSB	Get from EHC Context
lower	Not send	Get from lower layer
checksum	Not send	Compute checksum.
padding(_align)	Compute padding	Get padding

Table 8

- a. send-value designates an action that does not perform any compression or decompression of a field.
- b. elided designates an action where both peers have a local value of the field. The compression of the field consists in removing the field, and the decompression consists in retrieving the field value from a known local value. The local value may be stored in a EHC Context or defined by the EHC Rule (like a zero value for example).
- c. lsb designates an action where both peers have a local value of the field, but the compression consists in sending only the LSB bytes instead of the whole field. The decompression consists in retrieving the field from the LSB sent as well as some other additional local values.
- d. lower designates an action where the compression consists in not sending the field. The decompression consists in retrieving the field from the lower layers of the packet. A typical example is when both IP and UDP carry the length of the payload, then the length of the UDP payload can be inferred from the one of the IP layer.
- e. checksum designates an action where the compression consists in not sending a checksum field. The decompression consists in re-computing the checksum. ESP provides an integrity-check based on signature of the ESP payload (ICV). This makes removing checksum possible, without harming the checksum mechanism.
- f. padding designates an action that computes the padding of the ESP packet. The function is specific to the ESP.

For all actions, the function can be performed only when the appropriated parameters and fields are provided. When a field or a parameters does not have an appropriated value its value is

designated as "Unspecified". Specifically some fields such as inner IP addresses, ports or transport protocols are agreed during the SA negotiation and are specified by the SA. Their value in the SA may take various values that are not appropriated to enable a compression. For example, when these fields are defined as a range of values, or by selectors such as OPAQUE or ANY these fields cannot be retrieved from a local value. Instead, when they are defined as a "Single" value (i.e a single IP address, or a single port number or a single transport protocol number) compression and decompression can be performed. These SA related fields are considered as "Unspecified" when not limited to a "Single" value.

When a field or a parameter is "Unspecified", the EHC Rule MUST NOT be activated. This is the purpose of the EHC Strategy to avoid ending in such case. In any case, when one of these condition is not met, the EHC Rule MUST NOT perform any compression or decompression action and the packet MUST be discarded. When possible, an error SHOULD be raised and logged.

#### 7.1. EHC Rules for ESP

This section describes the EHC Rules for ESP which are summed up in the table below.

EHC Rule	Field	Action	Parameters
ESP_SPI	SPI	lsb	esp_spi_lsb, esp_spi
ESP_SN	Sequence Number	lsb	esp_sn_lsb, esp_sn_gen, esp_sn
ESP_NH	Next Header	elided	l4_proto, ipsec_mode
ESP_PAD	Pad Length, Padding	padding	esp_align, esp_encr

Table 9

ESP\_SPI designates the EHC Rule compressing / decompressing the SPI. ESP\_SPI is performed in the "post-esp" phase. The SPI is compressed using "lsb". The sending peer only places the LSB bytes of the SPI and the receiving peer retrieve the SPI from the LSB bytes carried in the packets as well as from the SPI value stored in the SA. The SPI MUST be retrieved as its full value is included in the signature check. The two peers MUST agree on the number of LSB bytes to be sent: "esp\_spi\_lsb". Upon agreeing on "esp\_spi\_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full SPI.

ESP\_SN designates the EHC Rule compressing / decompressing the ESP Sequence Number. ESP\_SN is performed in the "post-esp" phase. ESP\_SN is only activated if the SN ("esp\_sn"), the LSB significant bytes ("esp\_sn\_lsb") and the method used to generate the SN ("esp\_sn\_gen") are defined. The Sequence Number is compressed using "lsb". Similarly to the SPI, the Sequence Number MUST be retrieved in order to complete the signature check of the ESP packet. Unlike the SPI, the Sequence Number is not agreed by the peers, but is changing for every packet. As a result, in order to retrieve the Sequence Number from the LSB "esp\_sn\_lsb", the peers MUST agree on generating Sequence Number in a similar way. This is negotiated with "esp\_sn\_gen" and the receiver MUST ensure that "esp\_sn\_lsb" is big enough to absorb minor packet losses or time differences between the peers.

ESP\_NH designates the EHC Rule compressing / decompressing the ESP Next Header. ESP\_NH is performed in the "clear text esp" phase. ESP\_NH is only activated if the Next Header is specified. The Next Header can be specified as IP (IPv4 or IPv6) when the IPsec tunnel mode is used ("ipsec\_mode" set to "Tunnel") or when the transport mode ("ipsec\_mode" set to "Transport") is used when the Traffic Selector defines a "Single" Protocol ID ("l4\_proto"). The Next Header, is compressed using "elided". The Next Header indicates the Header in the Payload Data. When the Tunnel mode is chosen, the type of the header is known to be an IP header. Similarly, the TS may also hold transport layer protocol, which specifies the Next Header value for Transport mode. The Next Header value is only there to provide sufficient information for decapsulating ESP. In other words decompressing this fields would occur in the "clear text esp" phase and striped but directly removed again by the ESP stack. For these reasons, implementation may simply omit decompressing this field.

ESP\_PAD designates the EHC Rule compressing / decompressing the Pad Length and Padding fields. ESP\_PAD is performed in the "clear text esp" phase. Pad Length and Padding define the padding. The purpose of padding is to respect a 32 bit alignment for ESP or block sizes of the used cryptographic suite. As the ESP trailer is encrypted,

Padding and Pad Length MUST to be performed by ESP and not by the encryption algorithm. Thus, ESP\_PAD always needs to respect the cipher alignment ("esp\_encr"), if applicable. Compression may be performed especially when device support alignment smaller than 32 bit. Such alignment is designated as "esp\_align" and the padding bytes are the necessary bytes so the ESP packet has a length that is a multiple of "esp\_align".

When "esp\_align" is set to an 8-bit alignment padding bytes are not necessary, and Padding as well as Pad Length are removed. For values that are different from 8-bit alignment, padding bytes needs to be computed according to the ESP packet length why ESP\_PAD MUST be the last action of "clear text esp". The resulting number of padding byte is then expressed in Padding and Pad Length fields with Pad Length set to padding bytes number - 1 and Padding is generated as described in [RFC4303].

Combining the Pad Length and Padding fields could potentially add an overhead on fixed size padding. In fact some applications may only send the same type of fixed size data, in which case the Pad Length would not be necessary to be specified. However, the only corner case Pad Length fields would actually add an overhead is when padding is expected to be of zero size. In this case, specifying an 8-bit alignment solve this issue.

## 7.2. EHC Rules for inner IPv4

All IPv4 EHC Rules MUST be performed during the "clear text esp" phase. The EHC Rules are only defined for compressing the inner IPv4 header and thus can only be used when the SA is using the Tunnel mode.

EHC Rule	Field	Action	Parameters
IP4_OPT_DIS	Version	elided	ip_version
	Header Length	elided	
IP4_LENGTH	Total Length	lower	
IP4_ID	Identification	lsb	ip4_id, ip4_id_lsb
IP4_FRAG_DIS	Flags	elided	
	Fragment Offset	elided	
IP4_TTL_OUTER	Time To Live	elided	ip4_ttl
IP4_TTL_VALUE	Time To Live	elided	ip4_ttl
IP4_PROT	Protocol	elided	l4_proto
IP4_CHECK	Header Checksum	checksum	
IP4_SRC	Source Address	elided	ip4_src
IP4_DST	Dest. Address	elided	ip4_dst

Table 10

IP4\_OPT\_DIS designates that the IPv4 header does not include any options and indicates if the first byte of the IPv4 header - consisting of IP version and IPv4 Header Length, are compressed. The Version "ip\_version" is defined by the SA and is thus compressed using "elided". If the header does not contain any options, it is compressed with "elided" and decompressed to "20", the default length of the IPv4 header. If the header does contains some options, the length is not compressed.

IP4\_LENGTH designates the EHC Rule compressing / decompressing the Total Length Field of the inner IPv4 header. The Total Length is compressed by the sender and not sent. The receiver decompresses it by recomputing the Total Length from the outer IP header. The outer IP header can be IPv4 or IPv6 and IP4\_LENGTH MUST support both versions if both versions are supported by the device. Note that the length of the inner IP payload may also be subject to updates if decompression of the upper layers occurs.



IP4\_ID designates the EHC Rule compressing / decompressing the Identification Field. IP4\_ID is only activated if the ID ("ip4\_id"), the LSB significant bytes ("ip4\_id\_lsb") are defined. Upon agreeing on "ip4\_id\_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full IP Identification. Note also that unlike the ESP SN, the IPv4 Identification is not part of the SA. As a result, when the ID is compressed, its value MUST be stored in the EHC Context. The reserved attribute for that is "ip4\_id"

IP4\_FRAG\_DIS designates that the inner IPv4 header does not support fragmentation. If activated, IP4\_FRAG\_DIS indicates compression of Flags and Fragment Offset field in the IPv4 header which consists of 2 bytes. Both fields are compressed with "elided" and decompressed with their default value according to [RFC0791], which is 0b010 for Flags and 0 for Fragment Offset.

IP4\_TTL\_OUTER designates an EHC Rule compressing / decompressing the Time To Live field of the inner IP header. If the outer IP header is an IPv6 header, the Hop Limit is used for decompression. The Time To Live field is compressed / decompressed using "lower", thus the field is not sent. The receiver decompresses it by reading its value from the outer IP header (TTL in case of IPv4 or HL in case of IPv6).

IP4\_TTL\_VALUE designates an EHC Rule compressing / decompressing the Time To Live field of the inner IP header. IP4\_TTL\_VALUE is only activated when the Hop Limit ("ip4\_ttl") has been agreed. Time To Live is compressed / decompressed using the "elided" method.

IP4\_PROTO designates the EHC Rule compressing / decompressing the Protocol field of the inner IPv4 header. IP4\_PROTO is only activated if the Protocol is specified, that is when the Traffic Selectors defines a "Single" Protocol ID ("l4\_proto"). When the Protocol ID identified by the SA has a "Single" value, the Protocol is compressed and decompressed using the "elided" method.

IP4\_CHECK designates the EHC rule compressing / decompressing the Header Checksum field of the inner IPv4 header. The IPv4 header checksum is not sent by the sender and the receiver computes from the decompressed inner IPv4 header. IP4\_CHECK MUST compute the checksum and not fill the checksum field with zeros. As a result, IP4\_CHECK is the last decompressing EHC Rule to be performed on the decompressed IPv4 header.

IP4\_SRC compresses the source IP address of the inner IPv4 header. IP4\_SRC\_IP is only be activated when the Traffic Selectors agreed by the SA defines a "Single" source IP address ("ip4\_src"). The Source IP address is compressed / decompressed using the "elided" method.

IP4\_DST works in a similar way as IP4\_SRC\_IP but for the destination IP address ("ip4\_dst")

### 7.3. EHC Rules for inner IPv6

All IPv6 EHC Rules MUST be performed during the "clear text esp" phase. The EHC Rules are only defined for compressing the inner IPv6 header and thus can only be used when the SA is using the Tunnel mode.

EHC Rule	Field	Action	Parameters
IP6_OUTER	Version	elided	ip_version
	Traffic Class	lower	
	Flow Label	lower	
IP6_VALUE	Version	elided	ip_version
	Traffic Class	elided	ip6_tc
	Flow Label	elided	ip6_fl
IP6_LENGTH	Payload Length	lower	
IP6_NH	Next Header	elided	l4_proto
IP6_HL_OUTER	Hop Limit	lower	
IP6_HL_VALUE	Hop Limit	elided	ip6_hl
IP6_SRC	Source Address	elided	ip6_src
IP6_DST	Dest. Address	elided	ip6_dst

Table 11

IP6\_OUTER designates an EHC Rule for compressing / decompressing the first 32 bits of the inner IPv6 header formed by the Version, Traffic Class and Flow Label. IP6\_OUTER only proceeds to compression when both the outer and inner IP header are IPv6 header. When the outer IP header is an IPv4, the compression is bypassed. Bypassing enables to proceed to compression of IPv4 and IPv6 traffic in a VPN use case with a single SA. The Version "ip\_version" is defined by the SA and is thus compressed using "elided". The other parameters Traffic

Class and Flow Label are compressed using "lower". More specifically, the fields are not sent. The receiver decompresses them by reading their value from the outer IPv6 header.

IP6\_VALUE designates an EHC Rule for compressing / decompressing the first 32 bits of the inner IPv6 header formed by the Version, Traffic Class and Flow Label. IP6\_VALUE is only activated if the Version of the inner IP header agreed by the SA is set to "Version 6" ("ip\_version" set to "Version 6") and the specific values of the Traffic Class ("ip6\_tc") and the Flow Label ("ip6\_fl") are specified. With IP6\_VALUE all fields are compressed and decompressed using "elided". Version is provided by the SA ("ip\_version") while other fields are explicitly provided (ip6\_tc, ip6\_fl).

IP6\_LENGTH designates the EHC Rule compressing / decompressing the Payload Length Field of the inner IPv6 header. The Payload Length is compressed by the sender and is not sent. The receiver decompress it by recomputing the Payload Length from the outer IP header. The IP header can be IPv4 or IPv6 and IP6\_LENGTH MUST support both versions if both versions are supported by the device. Note that the length of the inner IP payload may also be subject to updates if decompression of the upper layers occurs.

IP6\_NH designates the EHC Rule compressing / decompressing the Next Header field of the inner IPv6 header. IP6\_NH is only activated if the Next Header is specified, that is when the Traffic Selectors defines a "Single" Protocol ID ("l4\_proto"). When the Protocol ID identified by the SA has a "Single" value, the Next Header is compressed and decompressed using the "elided" method.

IP6\_HL\_OUTER designates an EHC Rule compressing / decompressing the Hop Limit field of the inner IP header. If the outer IP header is an IPv4 header, the Time To Live is used for decompression. The Hop Limit field is compressed / decompressed using the "lower". More specifically, the fields are not sent. The receiver decompresses them by reading their value from the outer IPv6 header.

IP6\_HL\_VALUE designates an EHC Rule compressing / decompressing the Hop Limit field of the inner IP header. IP6\_HL\_VALUE is only activated when the Hop Limit ("ip6\_hl") has been agreed. The Hop Limit is compressed / decompressed using the "elided" method.

IP6\_SRC compresses the source IP address of the inner IP header. IP6\_SRC\_IP is only be activated when the Traffic Selectors agreed by the SA defines a "Single" source IP address ("ip6\_src"). The Source IP address is compressed / decompressed using the "elided" method.

IP6\_DST works in a similar way as IP6\_SRC\_IP but for the destination IP address ("ip6\_dst")

#### 7.4. EHC Rules for UDP

All UDP EHC Rules MUST be performed during the "pre-esp" phase. The EHC Rules are only defined when the Traffic Selectors agreed during the SA negotiation results in "Single" Protocol ID ("l4\_proto") which is set to UDP (17).

EHC Rule	Field	Action	Parameters
UDP_SRC	Source Port	elided	l4_source
UDP_DST	Dest. Port	elided	l4_dest
UDP_LENGTH	Length	lower	
UDP_CHECK	UDP Checksum	checksum	

Table 12

UDP\_SRC designates the EHC Rule that compresses / decompresses the UDP Source Port. UDP\_SRC is only activated when the Source Port agreed by the SA negotiation ("l4\_src") is "Single". The Source Port is then compressed / decompressed using the "elided" method.

UDP\_DST works in a similar way as UDP\_SRC but for the Destination Port ("l4\_dst").

UDP\_LENGTH designates the EHC Rule compressing / decompressing the Length Field of the UDP header. The length is compressed by the sender and is not sent. The receiver decompresses it by recomputing the Length from the IP address header. The IP address can be IPv4 or IPv6 and UDP\_LENGTH MUST support both versions if both versions are supported by the device.

UDP\_CHECK designates the EHC Rule compressing / decompressing the UDP Checksum. The UDP Checksum is not sent by the sender and the receiver computes from the decompressed UDP payload. UDP\_CHECK MUST compute the checksum and not fill the checksum field with zeros. As a result, UDP\_CHECK is the last decompressing EHC Rule to be performed on the decompressed UDP Payload.

### 7.5. EHC Rules for UDP-Lite

All UDP-lite EHC Rules MUST be performed during the "pre-esp" phase. The EHC Rules are only defined when the Traffic Selectors agreed during the SA negotiation results in a "Single" Protocol ID ("l4\_proto") which is set to UDPLite (136).

EHC Rule	Field	Action	Parameters
UDP-LITE_SRC	Source Port	elided	l4_source
UDP-LITE_DST	Dest. Port	elided	l4_dest
UDP-LITE_COVERAGE	Checksum Coverage	elided	udplite_coverage
UDP-LITE_CHECK	UDP-Lite Checksum	checksum	

Table 13

UDP-LITE\_SRC works similarly to UDP\_SRC

UDP-LITE\_DST works similarly to UDP\_DST

UDP-LITE\_COVERAGE designates the EHC Rule compressing / decompressing the UDP-Lite Coverage field. UDP-LITE\_COVERAGE is only activated when the Coverage ("udplite\_coverage") has been agreed with a valid value. The Coverage is compressed / decompressed using the "elided" method.

UDP-LITE\_CHECK designates the EHC Rule compressing / decompressing the UDP-Lite checksum. UDP-LITE\_CHECK is only activated if the Coverage is defined either elided or sent. UDP-LITE\_CHECK computes the checksum using "checksum" according to the uncompressed UDP packet and the value of the Coverage.

### 7.6. EHC Rules for TCP

All TCP EHC Rules MUST be performed during the "pre-esp" phase. The EHC Rules are only defined when the Traffic Selectors agreed during the SA negotiation results in a "Single" Protocol ID ("l4\_proto") which is set to TCP (6).

EHC Rule	Field	Action	Parameters
TCP_SRC	Source Port	elided	l4_source
TCP_DST	Dest. Port	elided	l4_dest
TCP_SN	Sequence Number	lsb	tcp_sn, tcp_lsb
TCP_ACK	Acknowledgment Number	lsb	tcp_ack, tcp_lsb
TCP_OPTIONS	Data Offset	elided	tcp_options
	Reserved Bits	elided	
TCP_CHECK	TCP Checksum	checksum	
TCP_URGENT	TCP Urgent Field	elided	tcp_urgent

Table 14

TCP\_SRC works similarly to UDP\_SRC.

TCP\_DST works similarly to UDP\_DST.

TCP\_SN designates the EHC Rule compressing / decompressing the TCP Sequence Number. TCP\_SN is only activated if the SN ("tcp\_sn") and the LSB significant bytes ("tcp\_lsb") are defined. The TCP SN is compressed using "lsb". The sending peer only places the LSB bytes of the TCP SN ("tcp\_sn") and the receiving peer retrieve the TCP SN from the LSB bytes carried in the packets as well as from the TCP SN value stored in EHC Context ("tcp\_sn"). The two peers MUST agree on the number of LSB bytes to be sent: "tcp\_lsb". Upon agreeing on "tcp\_lsb", the receiving peer MUST NOT agree on a value not carrying sufficient information to retrieve the full TCP SN. Note also that unlike the ESP SN, the TCP SN is not part of the SA. As a result, when the SN is compressed, the value of the TCP SN MUST be stored in the EHC Context. The reserved attribute for that is "tcp\_sn"

TCP\_ACK designates the EHC Rule compressing / decompressing the TCP Acknowledgment Number and works similarly to TCP SN. Note that "tcp\_lsb" is agreed for both TCP SN and TCP Acknowledgment. Similarly the value of the complete TCP Acknowledgment Number MUST be stored in the "tcp\_ack" attribute of the EHC Context.

TCP\_OPTIONS designates the EHC Rule compressing / decompressing TCP options related fields such as Data Offset and Reserved Bits. TCP\_OPTION can only be activated when the TCP Option ("tcp\_options") is defined. When "tcp\_options" is set to "False" and indicates there are no TCP Options, the Data Offsets and Reserved Bits are compressed / decompressed using the "elided" method with Data Offset and Reserved Bits set to zero.

TCP\_CHECK designates the EHC Rule compressing / decompressing the TCP Checksum. TCP\_CHECK works similarly as UDP\_CHECK.

TCP\_URGENT designates the EHC Rule compressing / decompressing the urgent related information. When "tcp\_urgent" is set to "False" and indicates there are no TCP Urgent related information, the Urgent Pointer is then "elided" and filled with zeros.

## 8. Diet-ESP EHC Strategy

From the attributes of the EHC Context, Diet-ESP defined as an EHC Strategy, which EHC Rules to apply. The EHC Strategy is defined for outbound packets which compresses the packet as well as for inbound packet where the decompression occurs.

Diet-ESP results from a compromise between compression efficiency, ease to configure Diet-ESP and the various use cases considered. In order to achieve a great simplicity,

- \* Diet-ESP favors compression methods that required fewer configuration: For IPv6, ip6\_tcfl\_comp and ip6\_hl\_com to "Outer" so that ip6\_tc, ip6\_fl and ip6\_hl can be derived from the packet. Similarly, ip4\_ttl\_comp has is set to "Outer" so ip4\_tll can be derived from the packet.
- \* Diet-ESP limits compression method to those foreseen as the most commonly used. As such, esp\_sn\_gen has been set to "Incremental" as this is the most common method used to generate SN. The other method would be "Time".
- \* Diet-ESP limits compression to the most foreseen scenarios. IPv4 compression has been limited in favor of IPv6 as constraint devices have largely adopted IPv6, and the gain versus the complexity to deploy IPv4 inner IP addresses has not been proved. As a result some compressions for IPv4 are not considered by DIet-ESP. This involved compression of the IPv4 options by setting ip4\_options to "No\_Options". Similarly IPv4 ID compression has not been enabled by setting ip4\_id and ip4\_id\_lsb to "Unspecified".
- \* Diet-ESP negotiated values shared by different rules such as tcp\_lsb which is shared for TCP ACK as well as for the TCP SN.

- \* Diet-ESP defines a logic to set the necessary parameters from those agreed by the standard ESP agreement, which limits the setting of parameters.

The following tables shows, which EHC Rules are activated by default for the supported protocols ESP, IPv4, IPv6, UDP, UDP-Lite and TCP when using the Diet-ESP strategy and which ones are activated due to certain circumstances or explicit negotiation

#### ESP:

EHC Rule	Activated if	Parameter	Value
ESP_SPI	Diet-ESP	esp_spi_lsb	Negotiated
		esp_spi	In SA
ESP_SN	Diet-ESP	esp_sn_lsb	Negotiated
		esp_sn_gen	Negotiated
		esp_sn	In SA
ESP_NH	Diet-ESP	ipsec_mode	In SA
		l4_proto	In SA
ESP_PAD	Diet-ESP	esp_align	Negotiated
		esp_encr	In SA

Table 15

#### IPv4:

EHC Rule	Activated if	Parameter	Value
IP4_OPT_DIS	ip_version==4	ip_version	In SA
IP4_LENGTH	ip_version==4	None	
IP4_FRAG_DIS	ip_version==4	None	
IP4_TTL_OUTER	ip_version==4	None	



IP4_TTL_OUTER	ip_version==4	l4_proto	In SA
IP4_CHECK	ip_version==4	None	
IP4_SRC	ip_version==4	ip4_src	In SA
IP4_DST	ip_version==4	ip4_dst	In SA

Table 16

IPv6:

EHC Rule	Activated if	Parameter	Value
IP6_OUTER	ip_version==6	ip_version	In SA
IP6_LENGTH	ip_version==6	None	
IP6_NH	ip_version==6	l4_proto	In SA
IP6_HL_OUTER	ip_version==6	None	
IP6_SRC	ip_version==6	ip6_src	In SA
IP6_DST	ip_version==6	ip6_dst	In SA

Table 17

UDP:

EHC Rule	Activated if	Parameter	Value
UDP_SRC	l4_proto==17	l4_source	In SA
UDP_DST	l4_proto==17	l4_dest	In SA
UDP_LENGTH	l4_proto==17	None	
UDP_CHECK	l4_proto==17	None	

Table 18

UDP-Lite:

EHC Rule	Activated if	Parameter	Value
UDP_LITE_SRC	14_proto==136	14_source	In SA
UDP_LITE_DST	14_proto==136	14_dest	In SA
UDP_LITE_COVERAGE	14_proto==136	udplite_coverage	Negotiated
UDP_LITE_CHECK	14_proto==136	None	

Table 19

TCP:

EHC Rule	Activated if	Parameter	Value
TCP_SRC	14_proto==6	14_source	In SA
TCP_DST	14_proto==6	14_dest	In SA
TCP_SN	14_proto==6	tcp_sn	In SA
		tcp_lsb	Negotiated
TCP_ACK	14_proto==6	tcp_ack	In SA
		tcp_lsb	Negotiated
TCP_OPTIONS	14_proto==6	tcp_options	Negotiated
TCP_CHECK	14_proto==6	None	
TCP_URGENT	14_proto==6	tcp_urgent	Negotiated

Table 20

Thus, the parameters that the two peers needs to agree on are:

- \* esp\_sn\_lsb
- \* esp\_spi\_lsb
- \* esp\_align
- \* udplite\_coverage
- \* tcp\_lsb
- \* tcp\_options

\* tcp\_urgent

Implementation may differ from the description below. However, the outcome MUST remain the same.

### 8.1. Outbound Packet Processing

Diet-ESP compression is defined as follows:

1. In phase "pre-esp": Match the inbound packet with the SA and determine if the Diet-ESP EHC Strategy has been activated. If the Diet-ESP EHC Strategy has been activated proceed to next step, otherwise skip all steps associated to Diet-ESP and proceed to the standard ESP as defined in [RFC4303]
2. In phase "pre-esp": If "l4\_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), proceed to the compression of the specific layer. Otherwise, the transport layer is not compressed.
3. In phase "clear text esp": If "ipsec\_mode" is set to "Tunnel" mode, determine "ip\_version" the IP version of the inner IP addresses and proceed to the appropriated inner IP address compression.
4. In phase "clear text esp" and "post-esp": Proceed to the ESP compression.

UDP compression is defined as below:

1. If "l4\_src" designates a "Single" Source Port, apply UDP\_SRC to compress the Source Port.
2. If "l4\_dst" designates a "Single" Destination Port, apply UDP\_DST to compress the Destination Port.
3. Apply UDP\_CHECK to compress the Checksum.
4. Apply UDP\_LENGTH to compress the Length.

UDP-lite compression is defined as below:

1. If "l4\_src" designates a "Single" Source Port, apply the UDP-LITE\_SRC to compress the Source Port.
2. If "l4\_dst" designates a "Single" Destination Port, apply the UDP-LITE\_DST, to compress the Destination Port.
3. If "udplite\_coverage" is specified, apply the UDP-LITE\_COVERAGE, to compress the Coverage.
4. Apply UDP-LITE\_CHECK to compress the Checksum.

TCP compression is defined as below:

1. If "l4\_src" designates a "Single" Source Port than apply the TCP\_SRC to compress the Source Port.

2. If "l4\_dst" designates a "Single" Destination Port than apply the TCP\_DST to compress the Destination Port.
3. If "tcp\_lsb" is lower than 4, then "tcp\_sn" "tcp\_ack" attributes of the Diet-ESP Context are updated with the value provided from the packet before applying the TCP\_SN and the TCP\_ACK EHC Rules.
4. If "tcp\_options" is set to "False" apply the TCP\_OPTIONS EHC Rule.
5. If "tcp\_urgent" is set to "False" apply the TCP\_URGENT EHC Rule.
6. Apply TCP\_CHECK to compress the Checksum.

Inner IPv6 Header compression is defined as below:

1. If "ip6\_src" designates a "Single" Source IP address, apply the IP6\_SRC to compress the IPv6 Source Address.
2. If "ip6\_dst" designates a "Single" Destination IP address, apply the IP6\_DST to decompress the IPv6 Destination Address.
3. Apply IP6\_HL\_OUTER to compress the Hop Limit.
4. If "l4\_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP6\_NH to compress the Next Header.
5. Apply, IP6\_LENGTH to compress the Length.
6. Apply IP6\_OUTER to compress Version, Traffic Class and Flow Label.

Inner IPv4 Header compression is defined as below:

1. Apply, IP4\_LENGTH to compress the Length.
2. Apply IP4\_TTL\_OUTER to compress Time To Live.
3. Apply, IP4\_CHECK to compress the IPv4 header checksum.
4. If "ip4\_src" designates a "Single" Source IP address, apply the IP4\_SRC to compress the IPv4 Source Address.
5. If "ip4\_dst" designates a "Single" Destination IP address, apply the IP4\_DST to decompress the IPv4 Destination Address.

ESP compression is defined as below:

1. In phase "clear text esp": If "ipsec\_mode" is set to "Tunnel" or "l4\_proto" is set to a "Single value - eventually different from TCP, UDP or UDP-Lite, apply ESP\_NH, to compress the Next Header.
2. In phase "clear text esp": If "esp\_encr" specify an encryption algorithm that does not provide padding, then apply ESP\_PAD to compress the Pad Length and Padding.
3. Proceed to the ESP encryption as defined in [RFC4303].
4. In phase "post-esp": If "esp\_sn\_lsb" is different from 4, then apply ESP\_SN. To compress the ESP SN.
5. In phase "post-esp": If "esp\_spi\_lsb" is different from 4, then apply ESP\_SPI to compress the SPI.

## 8.2. Inbound Packet Processing

Diet-ESP decompression is defined as follows:

1. Match the inbound packet with the SA and determine if the Diet-ESP EHC Strategy has been activated. When Diet-ESP is activated this means that the "esp\_spi\_lsb" are sufficient to index the SA and proceed to next step, otherwise skip all steps associated to Diet-ESP and proceed to the standard ESP as defined in [RFC4303]
2. In phase "clear text esp" and "post-esp": Proceed to the ESP decompression.
3. In phase "clear text esp": If "ipsec\_mode" is set to "Tunnel" mode, determine "ip\_version" the IP version of the inner IP addresses and proceed to the appropriated inner IP address decompression, except for the computation of the checksums and length.
4. In phase "pre-esp": If "l4\_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), proceed to the decompression of the specific layer, except for the computation of the checksums and length replaced by zero fields.
5. In phase "pre-esp": Proceed to the decompression of the checksums and length.

ESP decompression is defined as follows:

1. In phase "post-esp": If "esp\_spi\_lsb" is different from 4, then apply ESP\_SPI to decompress the SPI.
2. In phase "post-esp": If "esp\_sn\_lsb" is different from 4, then apply ESP\_SN. To decompress the ESP SN.
3. Proceed to the ESP signature validation and decryption as defined in [RFC4303].
4. In phase "clear text esp": If "ipsec\_mode" is set to "Tunnel" or "l4\_proto" is set to a "Single value - eventually different from TCP, UDP or UDP-Lite, apply ESP\_NH, to decompress the Next Header.
5. In phase "clear text esp": If "esp\_encr" specify an encryption algorithm that does not provide padding, then apply ESP\_PAD to compress the Pad Length and Padding.
6. Extract the ESP Data Payload and apply decompression EHC Rule to the ESP Data Payload.

UDP decompression is defined as follows:

1. If "l4\_src" designates a "Single" Source Port, apply UDP\_SRC to decompress the Source Port.
2. If "l4\_dst" designates a "Single" Destination Port, apply UDP\_DST to decompress the Destination Port.

3. Apply UDP\_LENGTH to compress the Length. The length value is computed from the length provided by the lower layer, with the additional added bytes during the UDP decompression including the length size.
4. Apply UDP\_CHECK to decompress the Checksum.
5. Update the Length of the lower layers:
  1. If "ipsec\_mode" is set to "Transport" mode, update the Length of the outer IP header (IPv4 or IPv6). The Length is incremented by the number of bytes generated by the decompression of the transport layer.
  2. If "ipsec\_mode" is set to "Tunnel" mode, update the Length of the inner IP address (IPv4 or IPv6) as well as the outer IP header (IPv4 or IPv6). The Length is incremented by the number of bytes generated by the decompression of the transport layer.

UDP-Lite decompression is defined as follows:

1. If "l4\_src" designates a "Single" Source Port, apply the UDP-LITE\_SRC to decompress the Source Port.
2. If "l4\_dst" designates a "Single" Destination Port, apply the UDP-LITE\_DST, to decompress the Destination Port.
3. If "udplite\_coverage" is specified, apply the UDP-LITE\_COVERAGE, to decompress the Coverage.
4. Apply UDP-LITE\_CHECK to compress the Checksum.
5. Update the Length of the lower layers as defined in UDP.

TCP decompression is defined as follows:

1. If "l4\_src" designates a "Single" Source Port than apply the TCP\_SRC to decompress the Source Port.
2. If "l4\_dst" designates a "Single" Destination Port than apply the TCP\_DST to decompress the Destination Port.
3. If "tcp\_lsb" is lower than 4, apply TCP\_SN and the TCP\_ACK to decompress the TCP Sequence Number and the TCP Acknowledgment Number.
4. If "tcp\_options" is set to "False" apply TCP\_OPTIONS to decompress Data Offset and Reserved Bits.
5. If "tcp\_urgent" is set to "False" apply the TCP\_URGENT to decompress the Urgent Pointer.
6. Apply TCP\_CHECK to decompress the Checksum.

Inner IPv6 decompression is defined as follows:

1. Apply IP6\_OUTER to decompress Version, Traffic Class and Flow Label.
2. Set the Length to zero.

3. If "l4\_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP6\_NH to decompress the Next Header.
4. Hop Limit is decompressed with IP6\_HL\_OUTER (with "ip6\_hl\_comp" set to "Outer").
5. If the "ip6\_src" designates a "Single" Source IP address, apply the IP6\_SRC to decompress the IPv6 Source Address.
6. If the "ip6\_dst" designates a "Single" Destination IP address then apply the IP6\_DST to decompress the IPv6 Destination Address.
7. Apply, IP6\_LENGTH to provide the replace the zero length value by its appropriated value. The Length value considers the length provided by the lower layers to which are added the additional bytes due to the decompression, minus the length of the inner IP6 Header.

Inner IPv4 decompression is defined as follows:

1. Apply, IP4\_LENGTH to provide the replace the zero length value by its appropriated value. The Length value considers the length provided by the lower layers to which are added the additional bytes due to the decompression, minus the length of the inner IPv4 Header. The value computed from the lower layer will have to be overwritten in case further decompression occurs.
2. Apply IP4\_TTL\_OUTER to decompress Time To Live.
3. If "l4\_proto" designates a "Single" Protocol ID (UDP, TCP or UDP-Lite), apply IP4\_PROT to decompress the Protocol Field.
4. If "ip4\_src" designates a "Single" Source IP address, apply the IP4\_SRC to decompress the IPv4 Source Address.
5. If "ip4\_dst" designates a "Single" Destination IP address then apply the IP4\_DST to decompress the IPv4 Destination Address.
6. Apply IP4\_CHECK to decompress the checksum of the IPv4 header

## 9. IANA Considerations

There are no IANA consideration for this document.

## 10. Security Considerations

This section lists security considerations related to the Diet-ESP protocol.

### Security Parameter Index (SPI):

The Security Parameter Index (SPI) is used by the receiver to index the Security Association that contains appropriated cryptographic material. If the SPI is not found, the packet is rejected as no further checks can be performed. In EHC, the value of the SPI is not reduced, but compressed why the SPI value may not be fully provided between the compressor and the de-

compressor. On the other hand, its uncompressed value is provided to the ESP-processing and no weakness is introduced to ESP itself. On an implementation perspective, it is strongly recommended that decompression is deterministic. Compression and decompression adds some additional treatment to the ESP packet, which might be used by an attacker. In order to minimize the load associated to decompression, decompression is expected to be deterministic. The incoming compressed SPI with the associated IP addresses should output a single and unique uncompressed SPI value. If an uncompressed SPI values have to be considered, then the receiver could end in n signature checks which may be used by an attacker for a DoS attack.

#### Sequence Number (SN):

The Sequence Number (SN) is used as an anti-replay attack mechanism. Compression and decompression of the SN is already part of the standard ESP namely the Extended Sequence Number (ESN). The SN in a standard ESP packet is 32 bit long, whether EHC enables to reduce it to 0 bytes and the main limitation to the compression a deterministic decompression. SN compression consists in indicating the least significant bits of the uncompressed SN on the wire. The size of the compressed SN must consider the maximum reordering index such that the probability that a later sent packet arrives before an earlier one. In addition the size of SN should also consider maximum consecutive packets lost during transmission. In the case of ESP, this number is set to  $2^{32}$  which is, in most real world case, largely over-provisioned. When the compression of the SN is not appropriately provisioned, the most significant bit value may be de-synchronized between the sending and receiving parties. Although IKEv2 provides some re-synchronization mechanisms, in case of IoT the de-synchronization will most likely result in a renegotiation and thus DoS possibilities. Note that IoT communication may also use some external parameters, i.e. other than the compressed SN, to define whether a packet be considered or not and eventually derive the SN. One such scenario may be the use of time windows. Suppose a device is expected to send some information every hour or every week. In this case, for example, the SN may be compressed to zero bytes. Instead the SN may be derived by incrementing the SN every hour after the last received valid packet. Considering the time the packet is received make it possible to consider the time derivation of the sensor clock. If TIME is used as the method to generate the SN, the receiver MUST ensure that the `esp_sn_lsb` is big enough to resist time differences between the nodes. Note also that the anti-replay mechanism needs to define the size of the anti-replay window. [RFC4303] provides guidance to set the window size and are similar to those used to define the size of the compressed SN.



## 11. Privacy Considerations

### Security Parameter Index (SPI):

Until Diet-ESP is deployed outside the scope of IoT and small devices, the use of a compressed SPI may provide an indication that one of the endpoint is a sensor. Such information may be used, for example, to evaluate the number of appliances deployed, or - in addition with other information, such as the time interval, the geographic location - be used to derive the type of data transmitted.

### Sequence Number (SN):

If incremented for each ESP packet, the SN may leak some information like the amount of transmitted data or the age of the sensor. The age of the sensor may be correlated with the software used and the potential bugs. On the other hand, re-keying will re-initialize the SN, but the cost of a re-keying may not be negligible and thus, frequent re-keying can be considered. In addition to the re-key operation, the SN may be generated in order to reduce the accuracy of the information leaked. In fact, the SN does not have to be incremented by one for each packet it just has to be an increasing function. Using a function such as a TIME may prevent characterizing the age or the use of the sensor. Note that the use of such function may also impact the compression efficiency and result in larger compressed SN. Another possibility may also consists in compressing the SN to the low order bytes to reduce the information related to the age or the number of packets being exchanged.

## 12. Acknowledgment

We would like to thank Orange and Universitee Pierre et Marie Curie for initiating the work on Diet-ESP. We Would like to thank Sylvain Killian for implementing an open source Diet-ESP on Contiki and testing it on the FIT IoT-LAB [fit-iot-lab] funded by the French Ministry of Higher Education and Research. We thank the IoT-Lab Team and the INRIA for maintaining the FIT IoT-LAB platform and for providing feed backs in an efficient way.

We would like to thank Bob Moskowitz for not copyrighting Diet HIP. The "Diet" terminology is from him.

We would like to thank those we received many useful feed backs among others: Dominique Bartel, Anna Minaburo, Suresh Krishnan, Samita Chakrabarti, Michael Richarson, Tero Kivinen.

## 13. References

### 13.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://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, <<https://www.rfc-editor.org/info/rfc4309>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC5795] Sandlund, K., Pelletier, G., and L-E. Jonsson, "The RObust Header Compression (ROHC) Framework", RFC 5795, DOI 10.17487/RFC5795, March 2010, <<https://www.rfc-editor.org/info/rfc5795>>.
- [RFC5858] Ertekin, E., Christou, C., and C. Bormann, "IPsec Extensions to Support Robust Header Compression over IPsec", RFC 5858, DOI 10.17487/RFC5858, May 2010, <<https://www.rfc-editor.org/info/rfc5858>>.
- [RFC7400] Bormann, C., "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 7400, DOI 10.17487/RFC7400, November 2014, <<https://www.rfc-editor.org/info/rfc7400>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 13.2. Informational References

- [RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zuniga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <<https://www.rfc-editor.org/info/rfc8724>>.
- [RFC8750] Migault, D., Guggemos, T., and Y. Nir, "Implicit Initialization Vector (IV) for Counter-Based Ciphers in Encapsulating Security Payload (ESP)", RFC 8750, DOI 10.17487/RFC8750, March 2020, <<https://www.rfc-editor.org/info/rfc8750>>.
- [I-D.ietf-tsvwg-udp-options]  
Touch, J., "Transport Options for UDP", Work in Progress, Internet-Draft, draft-ietf-tsvwg-udp-options-18, 26 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-tsvwg-udp-options-18.txt>>.
- [fit-iot-lab]  
"Future Internet of Things (FIT) IoT-LAB", <<https://www.ietf-lab.info>>.

## Appendix A. Illustrative Examples

### A.1. Single UDP Session IoT VPN

This section considers a IoT IPv6 probe hosting a UDP application. The probe is dedicated to a single application and establishes a single UDP session. As a result, inner IP addresses and UDP Ports have a "Single" value and can be easily compressed. The probes sets an IPsec VPN using IPv6 addresses in order to connect its secure domain - typically a Home Gateway. The use of IPv6 for inner and outer IP addresses, enables to infer inner IP fields from the outer IP address. The probes encrypts with AES-CCM\_8 [RFC4309]. AES-CCM does not have padding, so the padding is performed by ESP. The probes uses an 8 bit alignment which enables to fully compress the ESP Trailer. In addition, as the probe SA is indexed using the outer IP addresses (or eventually the radio identifiers) which enables to fully compress the SPI. As the probe provides information every hour, the Sequence Number using time can be derived from the received time, which enables to fully compress the SN.

Figure 3 represents the original UDP packet and Figure 4 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 61 bytes overhead. With IPv4 inner IP addressed Diet-ESP results in an 45 byte overhead reduction.

Further compression may be done for example by using an implicit IV [RFC8750] and by compressing the outer IP addresses (not represented on the figure). In addition, application data may also be compressed with mechanisms outside of the scope of Diet-ESP.

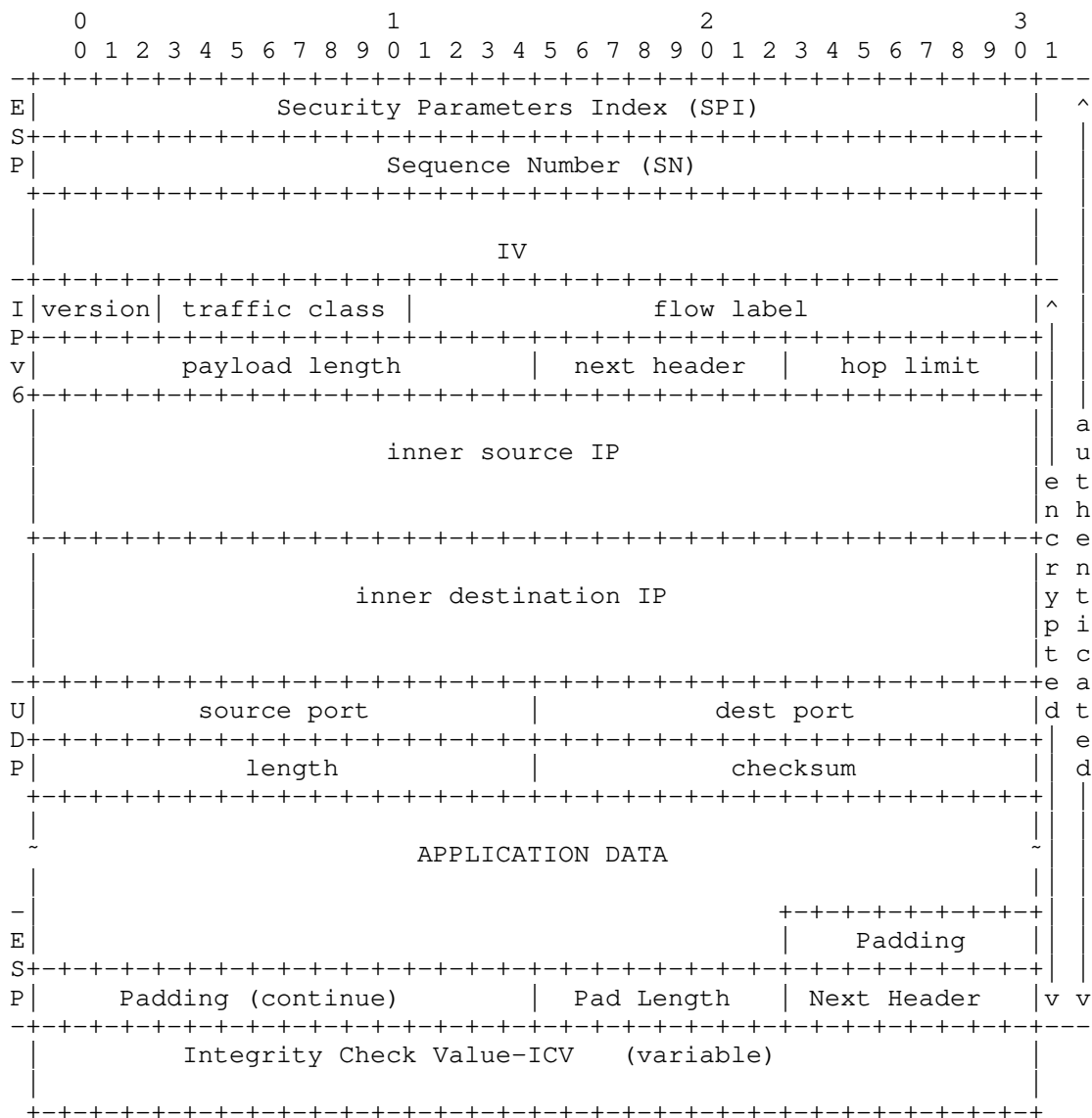


Figure 3: Standard ESP VPN Packet Description

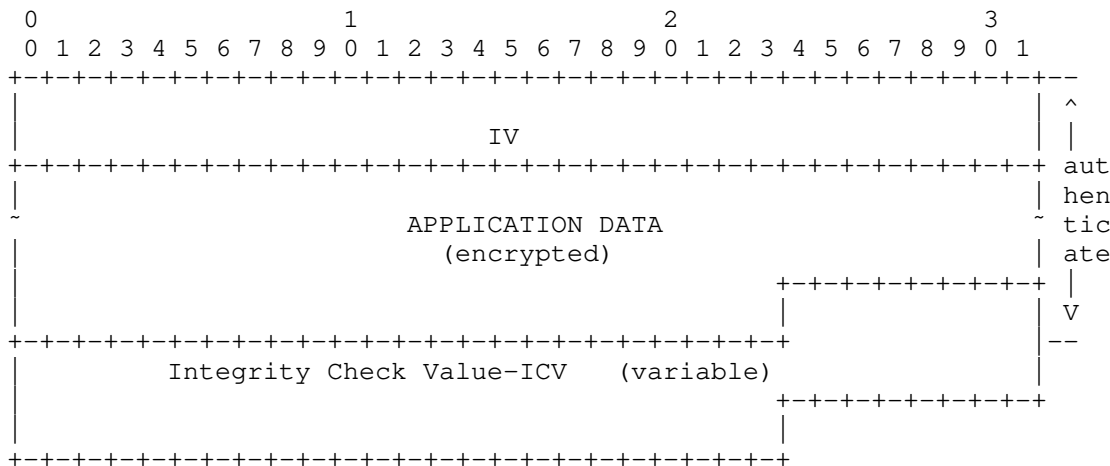


Figure 4: Diet-ESP Single UDP Session IoT VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified".

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	0
ESP_SN	esp_sn_lsb	0
	esp_sn_gen	
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	
	ip6_hl_comp	
IP6_LENGTH		
IP6_NH		
IP6_HL_OUTER		
IP6_SRC		
IP6_DST		
UDP_SRC		
UDP_DST		
UDP_LENGTH		
UDP_CHECK		

Table 21

#### A.2. Single TCP session IoT VPN

This section considers the same probe as described in Appendix A.1 but instead of using UDP as a transport layer, the probe uses TCP. In this case TCP is used with no options, no urgent pointers and the SN and ACK Number are compressed to 2 bytes as the throughput is expected to be low.

Figure 5 represents the original TCP packet and Figure 6 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 66 bytes overhead. With IPv4 inner address Diet-ESP results in a 50 byte overhead reduction.

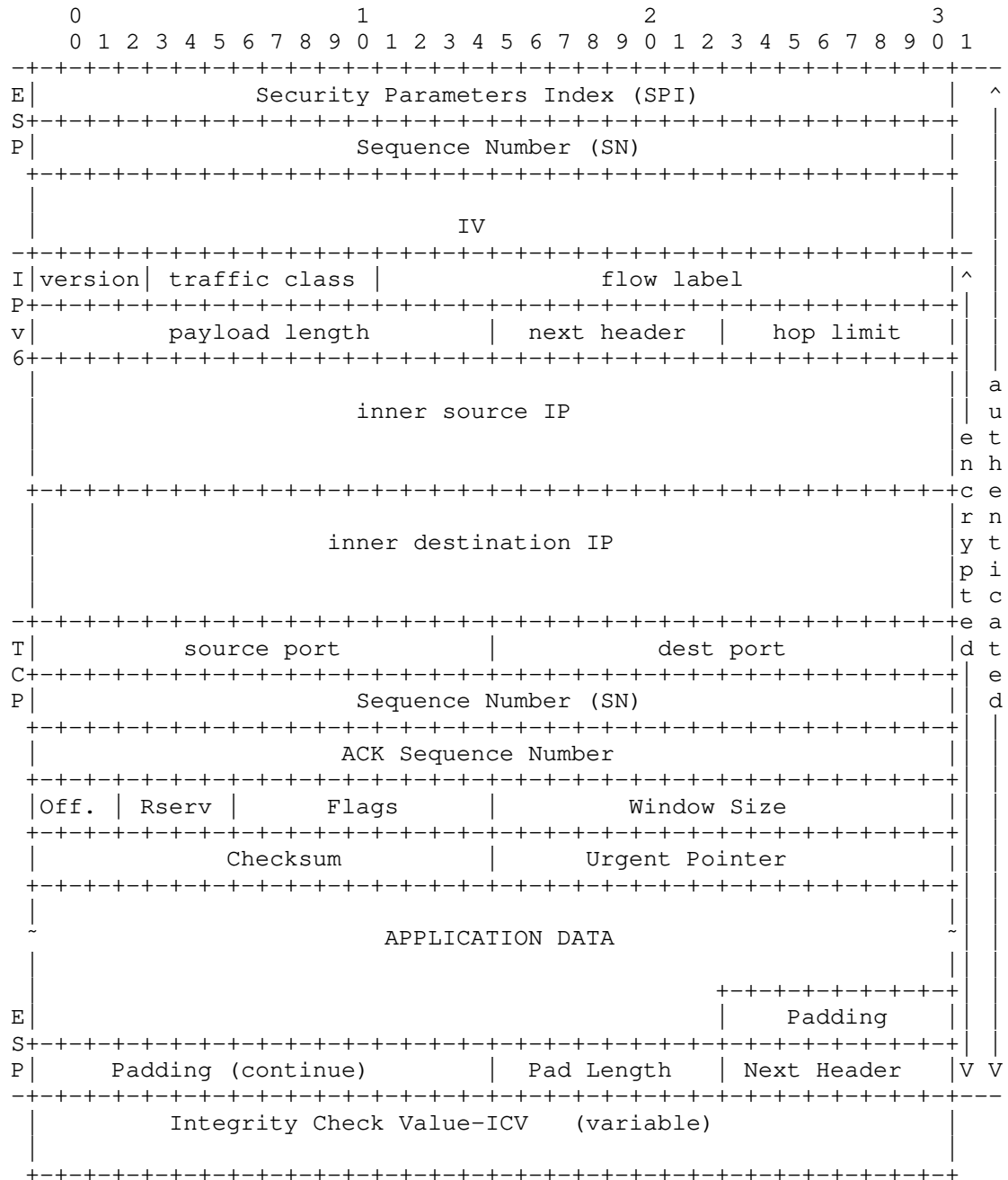


Figure 5: Standard IoT Single TCP Session VPN Packet Description



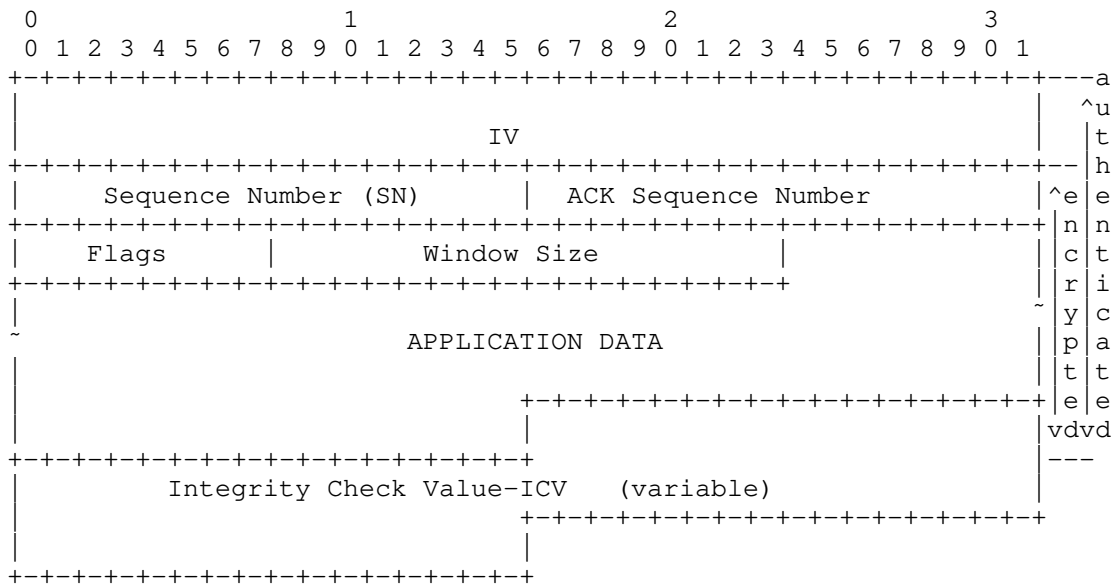


Figure 6: Diet-ESP Single TCP Session IoT VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified". Note for simplicity, tcp\_sn and tcp\_ack are negotiated to start with 0, but it could be any other value as well.

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	0
ESP_SN	esp_sn_lsb	0
	esp_sn_gen	
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	
	ip6_hl_comp	

IP6_LENGTH		
IP6_NH		
IP6_HL_OUTER		
IP6_SRC		
IP6_DST		
TCP_SRC		
TCP_DST		
TCP_SN	tcp_lsb	2
	tcp_sn	0
TCP_ACK	tcp_lsb	2
	tcp_ack	0
TCP_OPTIONS	tcp_options	"False"
TCP_CHECK		
TCP_URGENT	tcp_urgent	"False"

Table 22

### A.3. Traditional VPN

This section illustrates the case of an company VPN. The VPN is typically set by a remote host that forwards all its traffic to the security gateway. As transport protocols are "Unspecified", compression is limited to ESP and the inner IP header. For the inner IP header, the Destination IP address is "Unspecified" so the compression of the inner IP address excludes the Destination IP address. Similarly, the inner IP Next Header cannot be compressed as the transport layer is not specified. For ESP, the security gateway may only have a sufficiently low number of remote users with relatively low throughput in which case SPI and SN can be compressed to 2 bytes. As throughput remains relatively low, the alignment may also set to 8 bits.

#### A.3.1. IPv6 in IPv6

Figure 7 represents the original TCP packet with IPv6 inner IP addresses and Figure 8 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 32 bytes.

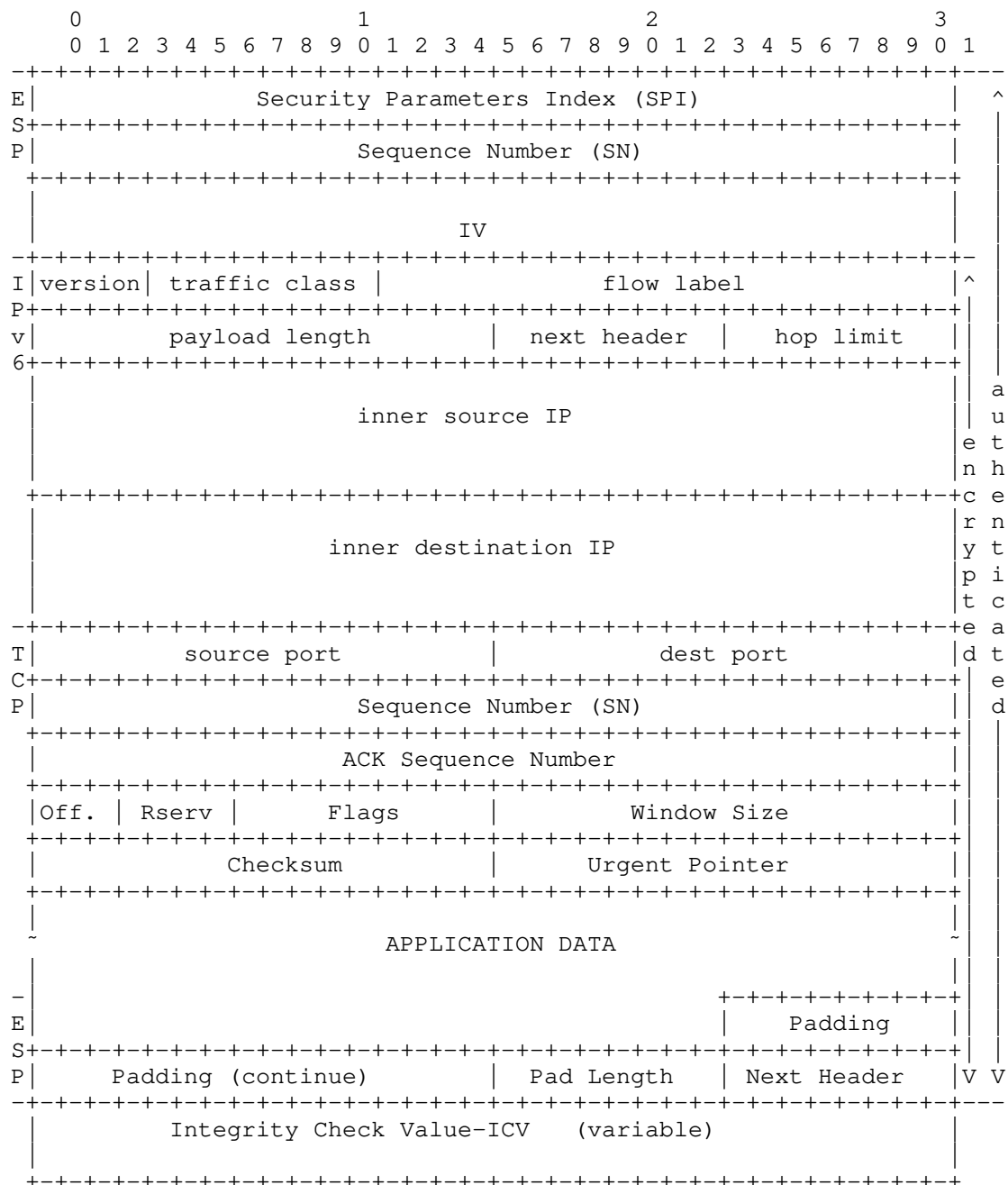


Figure 7: Standard ESP VPN Packet Description

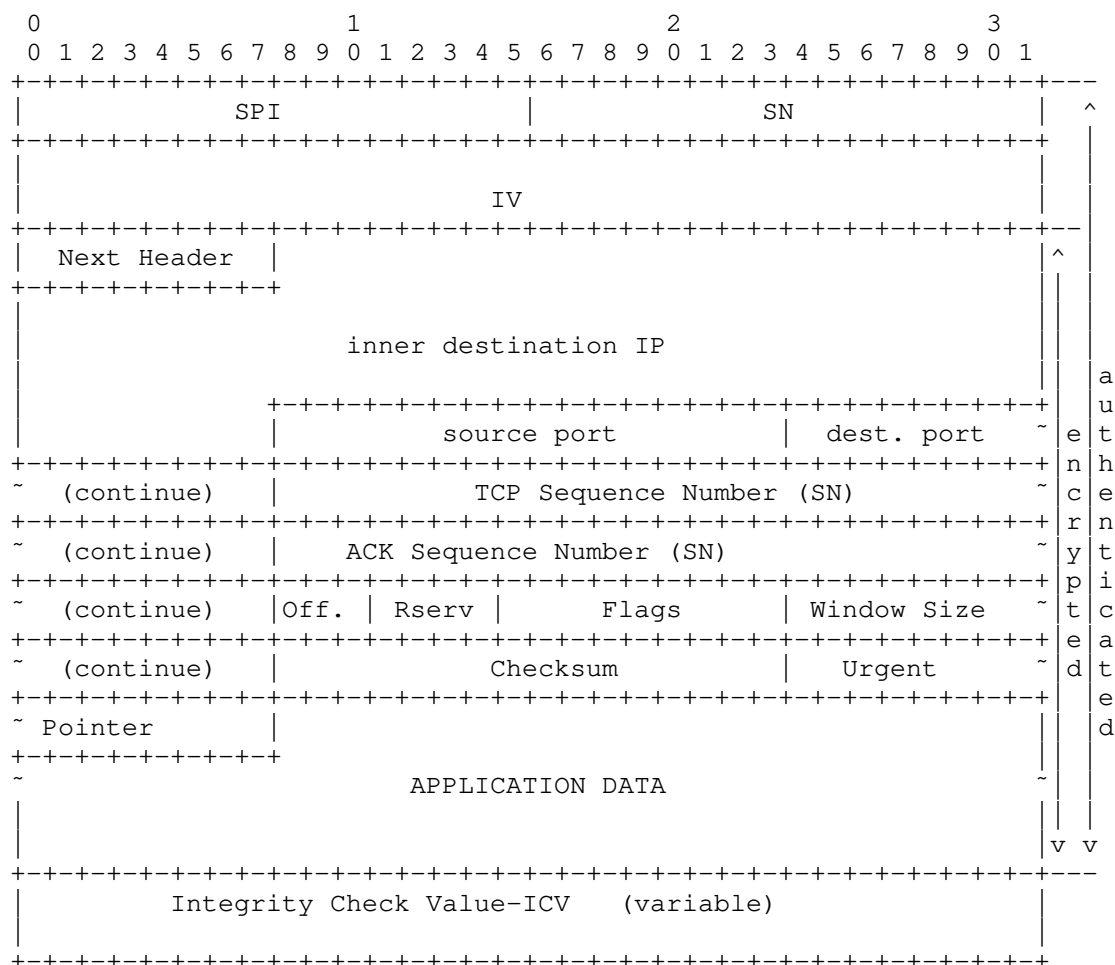


Figure 8: Diet-ESP VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified".

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	
IP6_LENGTH		
IP6_HL_OUTER	ip6_hl_comp	
IP6_SRC		

Table 23

## A.3.2. IPv6 in IPv4

If the compressed inner IP header is an IPv6, but the outer IP header is an IPv4 header, the activated rules differ, as IP6\_OUTER cannot be used. Instead, ip6\_tcfl\_comp and ip6\_hl\_comp are set to "Value". The resulting ESP packet is the same as in Figure 8.

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	
ESP_NH		
ESP_PAD	esp_align	8
IP6_OUTER	ip6_tcfl_comp	
IP6_LENGTH		
IP6_HL_OUTER	ip6_hl_comp	
IP6_SRC		

Table 24

#### A.3.3. IPv4 in IPv4

Figure 9 represents the original TCP packet with IPv4 inner IP addresses and Figure 10 represents the corresponding packet compressed with Diet-ESP. The compression with Diet-ESP results in a reduction of 24 bytes.

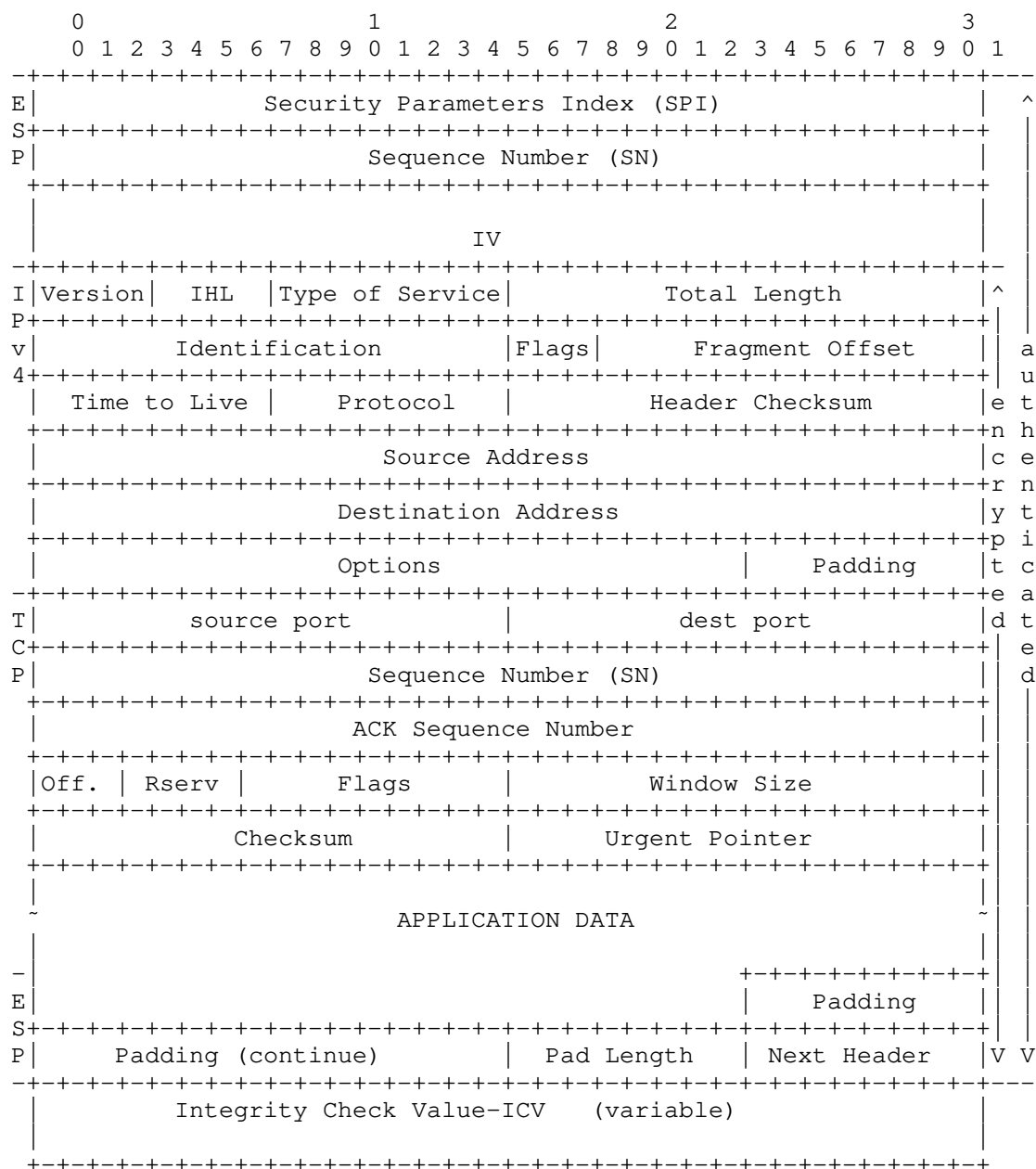


Figure 9: Standard ESP VPN Packet Description



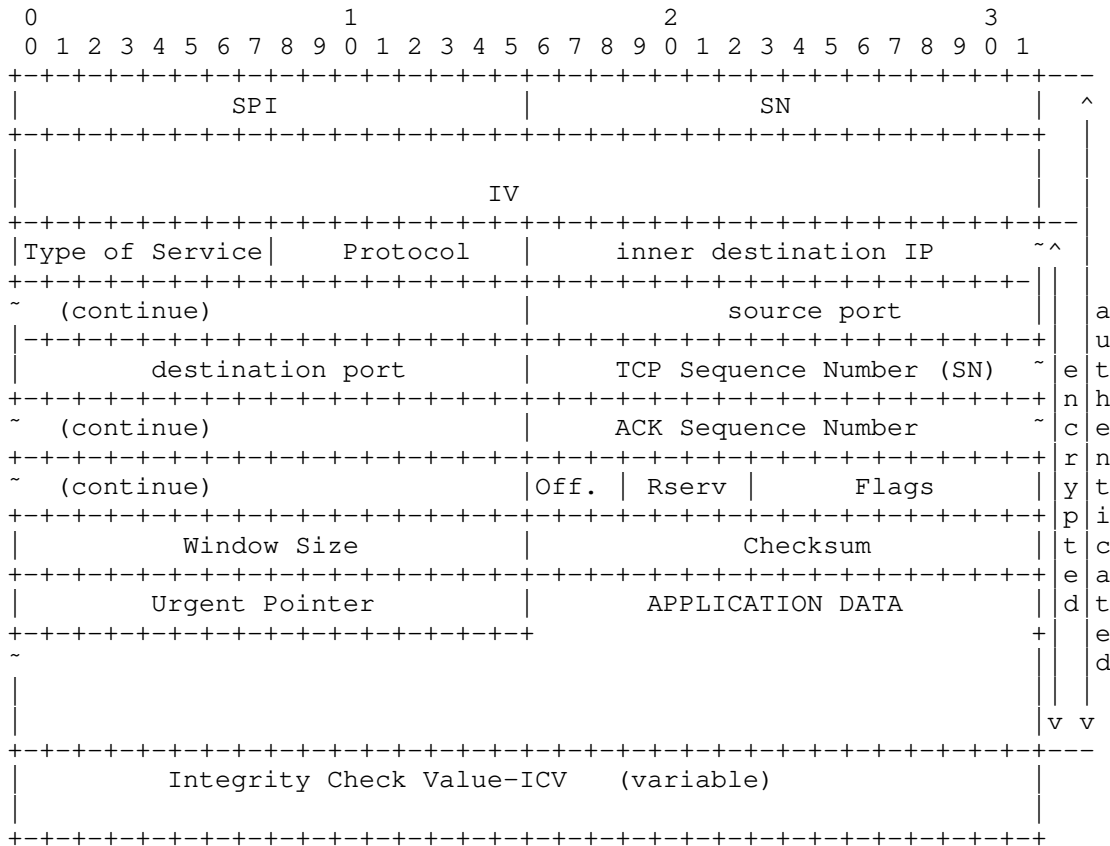


Figure 10: Diet-ESP VPN Packet Description

The following table illustrates the activated rules and the attributes of the Diet-ESP Context that needs an explicit agreement to achieve the compression. All other attributes used by the rules are part of the SA agreement. Parameters of not activated rules are left "Unspecified".

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP4_OPT_DIS		
IP4_LENGTH		
IP4_FRAG_DIS		
IP4_TTL_OUTER		
IP4_CHECK		
IP4_SRC		

Table 25

#### A.3.4. IPv4 in IPv6

If the compressed inner IP header is an IPv4, but the outer IP header is an IPv6 header, the activated rules differ, as IP4\_TTL\_OUTER cannot be used. Instead, IP4\_TTL\_VALUE is used. The resulting ESP packet is the same as in Figure 10.

EHC Rule	Context Attribute	Value
ESP_SPI	esp_spi_lsb	2
ESP_SN	esp_sn_lsb	2
	esp_sn_gen	"Incremental"
ESP_NH		
ESP_PAD	esp_align	8
IP4_OPT_DIS		
IP4_LENGTH		
IP4_FRAG_DIS		
IP4_CHECK		
IP4_SRC		

Table 26

## Authors' Addresses

Daniel Migault  
Ericsson  
8400 boulevard Decarie  
Montreal, QC H4P 2N2  
Canada  
Email: daniel.migault@ericsson.com

Tobias Guggemos  
LMU Munich  
Oettingenstr. 67  
80538 Munchen  
Germany  
Email: guggemos@nm.ifi.lmu.de  
URI: <http://www.nm.ifi.lmu.de/~guggemos>

Carsten Bormann  
Universitaet Bremen TZI  
Postfach 330440  
Bremen D-28359  
Germany  
Phone: +49-421-218-63921  
Email: cabo@tzi.org

David Schinazi  
Google LLC  
1600 Amphitheatre Parkway  
Mountain View, California 94043  
United States of America  
Email: dschinazi.ietf@gmail.com

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

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Requirements notation . . . . .	2
2. Introduction . . . . .	2
3. Terminology . . . . .	3
4. Implicit IV . . . . .	3
5. Initiator Behavior . . . . .	4
6. Responder Behavior . . . . .	4
7. Security Consideration . . . . .	4
8. IANA Considerations . . . . .	5
9. References . . . . .	5
9.1. Normative References . . . . .	5
9.2. Informational References . . . . .	6
Authors' Addresses . . . . .	7

### 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 a 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

- o IoT: Internet of Things.
- o IV: Initialization Vector.
- o 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.

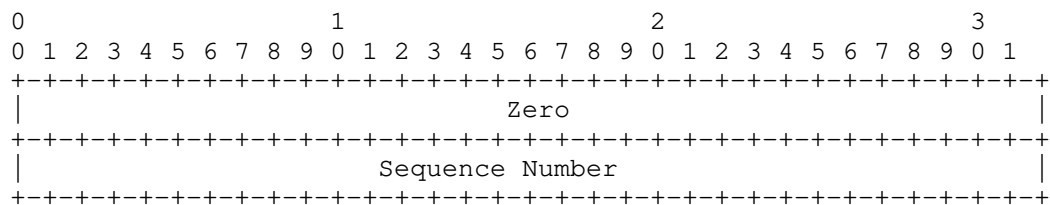


Figure 1: Implicit IV with a 4 byte Sequence Number

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

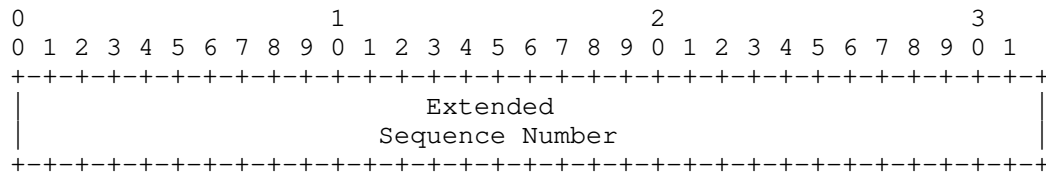


Figure 2: Implicit IV with an 8 byte Extended Sequence Number

- o 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](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](mailto:daniel.migault@ericsson.com)

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

Email: [guggemos@mnmt-team.org](mailto:guggemos@mnmt-team.org)  
URI: <http://mnmt-team.org/~guggemos>

Yoav Nir  
Dell EMC  
9 Andrei Sakharov St  
Haifa 3190500  
Israel

Email: [ynir.ietf@gmail.com](mailto:ynir.ietf@gmail.com)

Network Working Group  
Internet-Draft  
Updates: 4555, 6311 (if approved)  
Intended status: Standards Track  
Expires: December 1, 2017

V. Smyslov  
ELVIS-PLUS  
May 30, 2017

Responder Initiated IP Addresses Update in MOBIKE  
draft-smyslov-ipsecme-ikev2-r-mobike-00

Abstract

IKEv2 Mobility and Multihoming Protocol (MOBIKE) allows peers to update their IP addresses without re-establishing IKE and IPsec Security Associations (SAs). In the MOBIKE protocol it is the Initiator of the IKE SA, who is responsible for selecting new SA addresses and for initiating the IP addresses update procedure. This document presents an extension to the MOBIKE protocol that allows the Responder to initiate the update.

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 1, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology and Notation . . . . .	3
3. Protocol Overview . . . . .	3
4. Protocol Description . . . . .	4
4.1. Capability Advertising . . . . .	4
4.2. Responder Initiated IP Address Update . . . . .	5
4.2.1. High Availability Cluster Scenario . . . . .	7
5. Payload Formats . . . . .	8
5.1. MOBIKE_SUPPORTED Notification . . . . .	8
5.2. SWITCH_TO_IP_ADDRESS Notification . . . . .	9
6. Security Considerations . . . . .	9
7. IANA Considerations . . . . .	9
8. References . . . . .	9
8.1. Normative References . . . . .	9
8.2. Informative References . . . . .	10
Author's Address . . . . .	10

## 1. Introduction

The Internet Key Exchange protocol version 2 (IKEv2), specified in [RFC7296], is a key part of the IP Security (IPsec) architecture. It allows peers to perform authenticated key exchange, which results in establishing IKE Security Association (IKE SA) and to create a data protection channels called IPsec Security Associations (IPsec SAs). In original IKEv2 the IKE and IPsec SAs are established between the IP addresses used in IKEv2 negotiation. The IKEv2 Mobility and Multihoming Protocol (MOBIKE), specified in [RFC4555], extends the IKEv2 functionality by allowing peers to dynamically change IP addresses of the established SAs without the need to re-establish these SAs.

The main use case for the MOBIKE protocol is a remote access user that travels and moves from one from one IP address to another without re-establishing existing SAs with the VPN gateway. However, the MOBIKE also supports more complex scenarios when VPN gateway is multihomed and its addresses may change over time.

In the MOBIKE it is the Initiator (e.g. the remote access client) who is responsible for detecting the working IP addresses pairs and for deciding which pair to use. In other words, the Responder (e.g. the VPN gateway) plays a passive role and could neither initiate the IP address update process nor tell the Initiator which IP address is

preferred to use. This limitation makes use of complex scenarios less efficient and decreases the value of MOBIKE protocol.

For example, if the VPN gateway is a load sharing cluster where each node has its own IP address, then the cluster must be able to move SA between nodes depending on their current load. Currently Redirect Mechanism for IKEv2 [RFC5685] can accomplish this task, however it requires IKE SA to be re-established, that is very inefficient. Another possible solution is to use IKE SA Cloning along with the MOBIKE (see [RFC7791] for scenario description), but the limitation of the MOBIKE protocol makes this problematic. Obviously, the client has insufficient information to select when and to which of cluster IP addresses to move an SA to and the VPN gateway has no means to provide the client with this information.

This specification extends the MOBIKE protocol by adding ability for the Responder to ask the Initiator for IP address update and to provide it with the new IP address to use.

## 2. Terminology and 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].

In this document the term "Initiator" means the party who originally initiated the first IKE SA (in a series of possibly several rekeyed IKE SAs), and "Responder" means the other party. This is consistent with a way these terms are used in [RFC4555]. Note, that in [RFC7296] the terms "original initiator" and "original responder" mean the party, who initiated (or responded to) the latest IKE SA in a series of possibly several rekeyed IKE SAs.

## 3. Protocol Overview

The MOBIKE protocol is designed in such a way, that it is the IKE SA Initiator, who is responsible for performing the actions concerned with the selecting of a working IP addresses pair and for initiating an IP addresses update exchange. Usually the Initiator selects an IP addresses pair by continuously probing different pairs and choosing the working one. If several pairs work then the choice between them is arbitrary. The Responder cannot influence the process of selecting and cannot ask the client to immediately switch to a particular gateway's address. As a result the process of selection a new pair takes substantial time and may ends up with a suboptimal path. Moreover, in case the Responder isn't multihomed (and thus doesn't provide the Initiator with a list of additional IP

addresses), the change of its IP address cannot be handled by the MOBIKE.

Obviously, this limitation comes from the fact that there might be middleboxes on the path (like Network Address Translators (NAT) or firewalls) that might disallow IP packets to come from VPN gateway to the client unless the client first contacts the VPN gateway. For example, the client might reside behind a dynamic NAT that creates a mapping when IP packet first come from the client to the gateway. If the gateway tries to send an IP packet to the client from different IP address, the packet would be dropped since the NAT box has no corresponding mapping.

This specification provides the following solution to the described problem. When the Responder decides that its end of existing SA should be switched from its original IP address IP\_R1 to a new address IP\_R2, it initiates an INFORMATIONAL exchange containing a new notification SWITCH\_TO\_IP\_ADDRESS, that contains IP\_R2. The request message of this exchange is sent from IP\_R1 address, so that an existing middlebox mappings are used and the message can reach the Initiator. However, the response message is sent to a newly presented IP\_R2 address, so that a new middlebox mappings are created. Once the Initiator completes exchange containing SWITCH\_TO\_IP\_ADDRESS notification, it immediately initiates standard MOBIKE procedure for updating SA addresses by starting the INFORMATIONAL exchange containing UPDATE\_SA\_ADDRESSES notification.

#### 4. Protocol Description

##### 4.1. Capability Advertising

According to [RFC4555], the peers must exchange MOBIKE\_SUPPORTED notifications in the IKE\_AUTH exchange before they can use the MOBIKE protocol. If the Initiator supports this specification and is willing to use it, then it MUST include a single octet 0x52 ('R') in the notification data of the MOBIKE\_SUPPORTED notification sent to the Responder. There is no need for the Initiator to know whether the Responder supports this specification or not, so the MOBIKE\_SUPPORTED notification sent by the Responder has an empty notification data.

Note, that [RFC4555] specifies that MOBIKE\_SUPPORTED notification must contains no data when sending and the content of the notification data must be ignored while parsing. So, So, if the Responder doesn't support this specification, it will just ignore the content of the MOBIKE\_SUPPORTED notification and will use MOBIKE without this extension.

```

(IP_I1:500 -> IP_R1:500)
HDR, SAi1, KEi, Ni,
    N(NAT_DETECTION_SOURCE_IP),
    N(NAT_DETECTION_DESTINATION_IP)  -->

<-- (IP_R1:500 -> IP_I1:500)
HDR, SAr1, KEr, Nr,
    N(NAT_DETECTION_SOURCE_IP),
    N(NAT_DETECTION_DESTINATION_IP)

(IP_I1:4500 -> IP_R1:4500)
HDR, SK { IDi, CERT, AUTH,
    SAi2, TSi, TSr,
    N(MOBIKE_SUPPORTED('R')) }  -->

<-- (IP_R1:4500 -> IP_I1:4500)
HDR, SK { IDr, CERT, AUTH,
    SAr2, TSi, TSr,
    N(MOBIKE_SUPPORTED),
    N(ADDITIONAL_IP4_ADDRESS) }

```

#### 4.2. Responder Initiated IP Address Update

If the Initiator advertised its support for this specification during the initial exchange as described in Section 4.1, then the Responder is free to initiate IP Address Update request at any time. If the Initiator doesn't indicate its support for this extension, then the Responder MUST NOT initiate IP Address Update request. The IP Address Update request MUST NOT be initiated by the Initiator, the Responder MUST take no action if it receives such a request (apart from sending an empty response message to complete the exchange).

It is up to the Responder to decide when to initiate an IP Address request and what new address to include into it. Some of the possible reasons are:

- o Responder's IP address is changed due to Network Interface Card (NIC) reconfiguration
- o Responder is multihomed and wishes to switch SA to a different IP address
- o Responder is a cluster and wishes to move SA to a different node having its own IP address

The Responder requests the Initiator to update SA Address by initiating the INFORMATIONAL exchange containing a new status type notification SWITCH\_TO\_IP\_ADDRESS. The notification data of this



notification contains a new IP address the Responder requests the Initiator to use for the IKE SA and its Child SAs. Note, that the exchange request message MUST be sent using old SA addresses. In the example below the SA was established using IP\_I1 and IP\_R1 addresses for the Initiator and Responder respectively, and the Responder wishes to change the address of its end of the SA to IP\_R2. So, it initiates the INFORMATIONAL exchange from IP\_R1 address containing the SWITCH\_TO\_IP\_ADDRESS notification with IP\_R2 address. However, since the response message should come on a new address (IP\_R2), at this point the Responder MUST be able to receive packets on the IP address it included in the SWITCH\_TO\_IP\_ADDRESS notification.

```
<-- (IP_R1:4500 -> IP_I1:4500)
HDR, SK { N(SWITCH_TO_IP_ADDRESS(IP_R2)) }
```

Since the request is sent using old SA addresses, it is expected to pass through the middleboxes and reach the Initiator because it must use existing mappings.

Upon receiving the SWITCH\_TO\_IP\_ADDRESS notification the Initiator extracts its content and makes a decision whether the received IP address is appropriate for the SA. If the received IP address is among the addresses previously received from the Responder in ADDITIONAL\_IP4\_ADDRESS or ADDITIONAL\_IP6\_ADDRESS notifications, then it is definitely appropriate for the SA. Otherwise local policy must be consulted to decide whether the received IP is appropriate. If the address is considered inappropriate, then the Initiator MUST complete the exchange by sending an empty message to an old address (IP\_R1) and continue to use this address. It is RECOMMENDED that the Initiator immediately initiates Liveness Check exchange to ensure that the Responder is able to operate using old address.

```
(IP_I1:4500 -> IP_R1:4500)
HDR, SK {} -->
```

If the Initiator decides that the received address is appropriate, it completes the exchange by sending an empty response message to the newly received address (IP\_R2). Since the response message to the new Responder's address flows in the original direction (from the Initiator to the Responder), it should create new mappings in middleboxes, thus allowing further communication between them. After the response message is sent the Initiator MUST immediately initiate an IP address update procedure according to the MOBIKE specification by sending the INFORMATIONAL exchange request message containing the UPDATE\_SA\_ADDRESSES notification. See [RFC4555] for details. As a result, the remote IP address of the SA is changed from IP\_R1 to IP\_R2. Note that only the IP address is changed, the port remains the same.

```

(IP_I1:4500 -> IP_R2:4500)
HDR, SK {} -->

(IP_I1:4500 -> IP_R2:4500)
HDR, SK { N(UPDATE_SA_ADDRESSES),
          N(NAT_DETECTION_SOURCE_IP),
          N(NAT_DETECTION_DESTINATION_IP),
          N(COOKIE2) } -->

<-- (IP_R2:4500 -> IP_I1:4500)
HDR, SK { N(NAT_DETECTION_SOURCE_IP),
          N(NAT_DETECTION_DESTINATION_IP),
          N(COOKIE2) }

```

The Responder MUST NOT change IP address of the SA until it receives the UPDATE\_SA\_ADDRESSES notification from the Initiator. Note, that there is no need for the Responder to perform Return Routability check once the addresses are updated since it itself requested to change IP address of the SA and it successfully received a response from the Initiator sent to the new address. However, depending on the Responder's policy, the Return Routability check MAY be performed.

If the Responder doesn't receive a response message on a request containing the SWITCH\_TO\_IP\_ADDRESS notification after several retransmissions, then it means that either request or response message cannot use the new path and pass through the middleboxes. In this case the Responder's behavior depends on whether it advertised additional IP addresses before and whether old SA address is still available.

If old SA address is unavailable and no alternative addresses were advertised before, then the IKE SA and all associated Child SAs MUST be torn down. Otherwise the SA MAY be kept in an anticipation that the Initiator after some time detects the old IP address failure itself and performs IP addresses update.

#### 4.2.1. High Availability Cluster Scenario

In case the VPN gateway is a cluster consisting of several nodes each having its own IP address, both Load Sharing (LS) and High Availability (HA) goals may be achieved. For the purposes of HA the nodes share an IKE SA state while only one of them communicate with the IKE SA peer at any given time. Of the active node fails, the other nodes detect this fact and select a new active node for the SAs the failed node served. The selected node must then instruct the failed node peers to switch their SAs to a new IP address using this specification.

Since some exchanges might be in progress when the active node fails, some special measures must be taken to ensure that the IKE SA state is synchronised between the new active cluster node and the client. Protocol Support for High Availability of IKEv2/IPsec [RFC6311] describes the necessary measures. In particular, the new active node initiates the INFORMATIONAL exchange containing the IKEV2\_MESSAGE\_ID\_SYNC notification and optionally the IPSEC\_REPLAY\_COUNTER\_SYNC notification. [RFC6311] states that no other payload must be included in this exchange. However, in case the IP address of the new active node differs from the IP address of the failed active node it is necessary to combine the IKEV2\_MESSAGE\_ID\_SYNC and the SWITCH\_TO\_IP\_ADDRESS notifications in one exchange. So, this specification updates [RFC6311] in this regard: if HA cluster nodes have different IP addresses then in case of failover the request to synchronize Message IDs and the request to change IP address MUST be sent together in the same INFORMATIONAL exchange.

```
<-- (IP_R1:4500 -> IP_I1:4500)
      HDR, SK { N(SWITCH_TO_IP_ADDRESS(IP_R2))
                N(IKEV2_MESSAGE_ID_SYNC),
                [N(IPSEC_REPLAY_COUNTER_SYNC)] }

(IP_I1:4500 -> IP_R2:4500)
HDR, SK { N(IKEV2_MESSAGE_ID_SYNC) } -->
```

Once this exchange is completed the client MUST immediately perform an IP address update procedure according to the MOBIKE specification as described in Section 4.2.

## 5. Payload Formats

### 5.1. MOBIKE\_SUPPORTED Notification

The MOBIKE\_SUPPORTED Notification is defined in [RFC4555], Section 4.2.1 with the Notify Message Type 16396. This definition requires the notification data to be empty while sending and to be ignored when notification is received.

This document updates the definition from [RFC4555]. Exchange Initiator sets the notification data of the MOBIKE\_SUPPORTED Notification to a single octet 0x52 ('R') to indicate that this specification is supported.

## 5.2. SWITCH\_TO\_IP\_ADDRESS Notification

The Notify Message Type for this notification is <TBA by IANA>. The notification data contains new Responder's IP address.

For IPv4, the notification data is 4 octets long and is defined as follows:

```

          1             2             3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-----+-----+-----+-----+-----+-----+-----+-----+
    |                                     New Responder's IPv4 Address                                     |
    +-----+-----+-----+-----+-----+-----+-----+-----+

```

For IPv6, the notification data is 16 octets long and is defined as follows:

```

          1             2             3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-----+-----+-----+-----+-----+-----+-----+-----+
    |                                     New Responder's IPv6 Address                                     |
    +-----+-----+-----+-----+-----+-----+-----+-----+

```

The Protocol ID and SPI Size fields are set to zero.

## 6. Security Considerations

This specification is an extension of the MOBIKE protocol, so the Security Considerations described in [RFC4555] are applied.

## 7. IANA Considerations

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

<TBA> SWITCH\_TO\_IP\_ADDRESS

## 8. References

### 8.1. Normative References

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

- [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006, <<http://www.rfc-editor.org/info/rfc4555>>.
- [RFC6311] Singh, R., Ed., Kalyani, G., Nir, Y., Sheffer, Y., and D. Zhang, "Protocol Support for High Availability of IKEv2/IPsec", RFC 6311, DOI 10.17487/RFC6311, July 2011, <<http://www.rfc-editor.org/info/rfc6311>>.
- [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

- [RFC5685] Devarapalli, V. and K. Weniger, "Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5685, DOI 10.17487/RFC5685, November 2009, <<http://www.rfc-editor.org/info/rfc5685>>.
- [RFC7791] Migault, D., Ed. and V. Smyslov, "Cloning the IKE Security Association in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7791, DOI 10.17487/RFC7791, March 2016, <<http://www.rfc-editor.org/info/rfc7791>>.

## Author's Address

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

Phone: +7 495 276 0211  
Email: [svan@elvis.ru](mailto:svan@elvis.ru)

Network Working Group  
Internet-Draft  
Updates: 4555 (if approved)  
Intended status: Standards Track  
Expires: May 19, 2022

V. Smyslov  
ELVIS-PLUS  
November 15, 2021

Responder Initiated IP Addresses Update in MOBIKE  
draft-smyslov-ipsecme-ikev2-r-mobike-09

Abstract

IKEv2 Mobility and Multihoming Protocol (MOBIKE), defined in [RFC4555] allows peers to update their IP addresses without re-establishing IKE and IPsec Security Associations (SAs). In the MOBIKE protocol it is the initiator of the IKE SA, who is responsible for selecting new SA addresses and for initiating the IP addresses update procedure. This document presents an extension to the MOBIKE protocol that allows the responder to initiate IP address update. The document updates [RFC4555].

Status of This Memo

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

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

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

This Internet-Draft will expire on May 19, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology and Notation . . . . .	3
3. Protocol Overview . . . . .	3
4. Protocol Description . . . . .	4
4.1. Capability Advertising . . . . .	4
4.2. Responder Initiated IP Address Update . . . . .	5
5. Payload Formats . . . . .	7
5.1. MOBIKE_SUPPORTED Notification . . . . .	7
5.2. SWITCH_TO_IP_ADDRESS Notification . . . . .	7
6. Security Considerations . . . . .	7
7. IANA Considerations . . . . .	8
8. References . . . . .	8
8.1. Normative References . . . . .	8
8.2. Informative References . . . . .	8
Author's Address . . . . .	8

## 1. Introduction

The Internet Key Exchange protocol version 2 (IKEv2), specified in [RFC7296], is a key part of the IP Security (IPsec) architecture. It allows peers to perform authenticated key exchange, which results in establishing IKE Security Association (IKE SA) and to create a data protection channels called IPsec Security Associations (IPsec SAs). In original IKEv2 the IKE and IPsec SAs are established between the IP addresses used in IKEv2 negotiation. The IKEv2 Mobility and Multihoming Protocol (MOBIKE), specified in [RFC4555], extends the IKEv2 functionality by allowing peers to dynamically change IP addresses of the established SAs without the need to re-establish these SAs.

The main use case for the MOBIKE protocol is a remote access user that travels and moves from one IP address to another without re-establishing existing SAs with the VPN gateway. However, the MOBIKE also supports more complex scenarios when VPN gateway is multihomed and its addresses may change over time.

In the MOBIKE it is the original initiator of the IKE SA (e.g. the remote access client) who is responsible for detecting the working IP addresses pairs and for deciding which pair to use. In other words, the responder (e.g. the VPN gateway) plays a passive role and could neither initiate the IP address update process nor tell the initiator

which IP address is preferred to use. This limitation makes use of complex scenarios less efficient and decreases the value of MOBIKE protocol.

For example, if the VPN gateway is a load sharing cluster where each node has its own IP address, then the cluster must be able to move SA between nodes depending on their current load. Currently Redirect Mechanism for IKEv2 [RFC5685] can accomplish this task, however it requires new IKE SA to be established, that is very inefficient. Another possible solution is to use IKE SA Cloning along with the MOBIKE (see [RFC7791] for scenario description), but the limitation of the MOBIKE protocol makes this problematic. Obviously, the client has insufficient information to choose when and to which of cluster IP addresses to move an SA to and the VPN gateway has no means to provide the client with this information.

This specification extends the MOBIKE protocol by adding ability for the responder to ask the initiator for IP address update and to provide it with the new IP address to use.

## 2. Terminology and Notation

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

In this document the term "initiator" means the party who originally initiated the first IKE SA (in a series of possibly several rekeyed IKE SAs), and "responder" means the other party. This is consistent with a way these terms are used in [RFC4555]. Note, that in [RFC7296] the terms "original initiator" and "original responder" mean the party, who initiated (or responded to) the latest IKE SA in a series of possibly several rekeyed IKE SAs.

## 3. Protocol Overview

The MOBIKE protocol is designed in such a way, that it is the IKE SA initiator, who is responsible for performing the actions concerned with the selecting of a working IP addresses pair and for initiating an IP addresses update exchange. Usually the initiator selects an IP addresses pair by periodically probing different pairs and choosing the working one. If several pairs work then the choice between them is arbitrary. The responder cannot influence the process of selecting and cannot ask the client to immediately switch to a particular gateway's address. As a result the process of selection a



new pair takes substantial time and may end up with a suboptimal path.

Obviously, this limitation comes from the fact that there might be middleboxes on the path like Network Address Translators (NAT) or firewalls, that might disallow IP packets to come from VPN gateway to the client unless the client first contacts the VPN gateway. For example, the client might reside behind a dynamic NAT that creates a mapping when IP packet first comes from the client to the gateway. If the gateway tries to send an IP packet to the client from different IP address, the packet would be dropped since the NAT box has no corresponding mapping.

This specification provides the following solution to the described problem. When the responder decides that its end of existing SA should be switched from its original IP address IP\_R1 to a new IP address IP\_R2, it initiates an INFORMATIONAL exchange containing a new notification SWITCH\_TO\_IP\_ADDRESS, that contains IP\_R2. Once the initiator completes an exchange containing SWITCH\_TO\_IP\_ADDRESS notification, it immediately initiates standard MOBIKE procedure for updating SA addresses by starting the INFORMATIONAL exchange containing UPDATE\_SA\_ADDRESSES notification.

#### 4. Protocol Description

##### 4.1. Capability Advertising

According to [RFC4555], the peers must exchange MOBIKE\_SUPPORTED notifications in the IKE\_AUTH exchange before they can use the MOBIKE protocol. If the initiator supports this specification and is willing to use it, then it MUST include a single octet 0x52 ('R') in the notification data of the MOBIKE\_SUPPORTED notification sent to the responder. There is no need for the initiator to know whether the responder supports this specification or not, so the MOBIKE\_SUPPORTED notification sent by the responder has an empty notification data.

Note, that [RFC4555] specifies that MOBIKE\_SUPPORTED notification must contain no data when sending and the content of the notification data must be ignored while parsing. So, if the responder doesn't support this specification, it will just ignore the content of the MOBIKE\_SUPPORTED notification and will use MOBIKE without this extension.

```

(IP_I1:500 -> IP_R1:500) -->
HDR, SAi1, KEi, Ni,
  N(NAT_DETECTION_SOURCE_IP),
  N(NAT_DETECTION_DESTINATION_IP)

      <-- (IP_R1:500 -> IP_I1:500)
      HDR, SAR1, KEr, Nr,
        N(NAT_DETECTION_SOURCE_IP),
        N(NAT_DETECTION_DESTINATION_IP)

(IP_I1:4500 -> IP_R1:4500) -->
HDR, SK { IDi, CERT, AUTH,
  SAi2, TSi, TSr,
  N(MOBIKE_SUPPORTED('R')) }

      <-- (IP_R1:4500 -> IP_I1:4500)
      HDR, SK { IDr, CERT, AUTH,
        SAR2, TSi, TSr,
        N(MOBIKE_SUPPORTED),
        N(ADDITIONAL_IP4_ADDRESS) }

```

#### 4.2. Responder Initiated IP Address Update

If the initiator advertised its support for this specification during the initial exchange as described in Section 4.1, then the responder is free to initiate IP Address Update request at any time. If the initiator doesn't indicate its support for this extension, then the responder MUST NOT initiate IP Address Update request. The IP Address Update request MUST NOT be initiated by the initiator, the responder MUST NOT take any action if it receives such a request (apart from sending an empty response message to complete the exchange).

It is up to the responder to decide when to initiate an IP Address Update request and what new address to include into it. Some of the possible reasons are:

- o the responder is multihomed and wishes to switch an SA to a different IP address
- o the responder is a cluster and wishes to move an SA to a different node having its own IP address

The responder requests the initiator to update SA address by initiating the INFORMATIONAL exchange containing a new status type notification SWITCH\_TO\_IP\_ADDRESS. Its notification data contains a new IP address the responder requests the initiator to use for the IKE SA and its Child SAs. In the example below the SA was

established using IP\_I1 and IP\_R1 addresses for the initiator and responder respectively, and the responder wishes to change the address of its end of the SA to IP\_R2. So, it initiates the INFORMATIONAL exchange from IP\_R1 address containing the SWITCH\_TO\_IP\_ADDRESS notification with IP\_R2 address.

```

                <-- (IP_R1:4500 -> IP_I1:4500)
                    HDR, SK { N(SWITCH_TO_IP_ADDRESS(IP_R2)) }
(IP_I1:4500 -> IP_R1:4500) -->
HDR, SK {}

```

Upon receiving the SWITCH\_TO\_IP\_ADDRESS notification the initiator extracts its content and makes a decision whether the received IP address is appropriate for the SA. If the received IP address is among the addresses previously received from the responder in ADDITIONAL\_IP4\_ADDRESS or ADDITIONAL\_IP6\_ADDRESS notifications, then it is appropriate for the SA. Otherwise local policy must be consulted to decide whether the received IP is appropriate. If the address is considered inappropriate, then the initiator MUST current address. It is RECOMMENDED that the initiator immediately initiates Liveness Check exchange to ensure that the responder is able to operate using its current address.

If the initiator makes a decision that the received address is appropriate the initiator initiates an IP address update procedure according to the MOBIKE specification by sending an INFORMATIONAL exchange request message containing the UPDATE\_SA\_ADDRESSES notification. See [RFC4555] for details. As a result, the remote IP address of the SA is changed from IP\_R1 to IP\_R2. Note that only the IP address is changed, the port remains the same.

```

(IP_I1:4500 -> IP_R2:4500) -->
HDR, SK { N(UPDATE_SA_ADDRESSES),
          N(NAT_DETECTION_SOURCE_IP),
          N(NAT_DETECTION_DESTINATION_IP),
          N(COOKIE2) }

                <-- (IP_R2:4500 -> IP_I1:4500)
                    HDR, SK { N(NAT_DETECTION_SOURCE_IP),
                              N(NAT_DETECTION_DESTINATION_IP),
                              N(COOKIE2) }

```

The responder MUST NOT change IP addresses of the SA until it receives the UPDATE\_SA\_ADDRESSES notification from the initiator.

## 5. Payload Formats

### 5.1. MOBIKE\_SUPPORTED Notification

The MOBIKE\_SUPPORTED Notification is defined in [RFC4555], Section 4.2.1 with the Notify Message Type 16396. This definition requires the notification data to be empty while sending and to be ignored when notification is received.

This document updates the definition from [RFC4555]. Exchange initiator sets the notification data of the MOBIKE\_SUPPORTED Notification to a single octet 0x52 ('R') to indicate that this specification is supported.

### 5.2. SWITCH\_TO\_IP\_ADDRESS Notification

The Notify Message Type for this notification is <TBA by IANA>. The notification data contains new responder's IP address.

For IPv4, the notification data is 4 octets long and is defined as follows:

```

      1             2             3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     New Responder's IPv4 Address                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

For IPv6, the notification data is 16 octets long and is defined as follows:

```

      1             2             3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     New Responder's IPv6 Address                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The Protocol ID and SPI Size fields are set to zero.

## 6. Security Considerations

This specification is an extension of the MOBIKE protocol, so the Security Considerations described in [RFC4555] are applied.

## 7. IANA Considerations

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

<TBA> SWITCH\_TO\_IP\_ADDRESS

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, DOI 10.17487/RFC4555, June 2006, <<https://www.rfc-editor.org/info/rfc4555>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.

### 8.2. Informative References

- [RFC5685] Devarapalli, V. and K. Weniger, "Redirect Mechanism for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5685, DOI 10.17487/RFC5685, November 2009, <<https://www.rfc-editor.org/info/rfc5685>>.
- [RFC7791] Migault, D., Ed. and V. Smyslov, "Cloning the IKE Security Association in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7791, DOI 10.17487/RFC7791, March 2016, <<https://www.rfc-editor.org/info/rfc7791>>.

Author's Address

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

Phone: +7 495 276 0211  
Email: [svan@elvis.ru](mailto:svan@elvis.ru)

Internet Engineering Task Force  
Internet-Draft  
Updates: 7296 (if approved)  
Intended status: Standards Track  
Expires: January 10, 2020

C. Tjhai  
M. Tomlinson  
Post-Quantum  
G. Bartlett  
S. Fluhrer  
Cisco Systems  
D. Van Geest  
ISARA Corporation  
O. Garcia-Morchon  
Philips  
V. Smyslov  
ELVIS-PLUS  
July 9, 2019

Framework to Integrate Post-quantum Key Exchanges into Internet Key  
Exchange Protocol Version 2 (IKEv2)  
draft-tjhai-ipsecme-hybrid-qske-ikev2-04

Abstract

This document describes how to extend Internet Key Exchange Protocol Version 2 (IKEv2) so that the shared secret exchanged between peers has resistance against quantum computer attacks. The basic idea is to exchange one or more post-quantum key exchange payloads in conjunction with the existing (Elliptic Curve) Diffie-Hellman payload.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 10, 2020.

## Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Problem Description . . . . .	2
1.2. Proposed Extension . . . . .	3
1.3. Changes . . . . .	4
1.4. Document Organization . . . . .	5
2. Design Criteria . . . . .	5
3. The Framework of Hybrid Post-Quantum Key Exchange . . . . .	7
3.1. Overall design . . . . .	7
3.2. Overall Protocol . . . . .	8
3.2.1. IKE_SA_INIT Round: Negotiation . . . . .	9
3.2.2. IKE_INTERMEDIATE Round: Additional Key Exchanges . . . . .	10
3.2.3. IKE_AUTH Exchange . . . . .	11
3.2.4. CREATE_CHILD_SA Exchange . . . . .	11
4. IANA Considerations . . . . .	14
5. Security Considerations . . . . .	14
6. Acknowledgements . . . . .	16
7. References . . . . .	16
7.1. Normative References . . . . .	16
7.2. Informative References . . . . .	16
Appendix A. Alternative Design . . . . .	17
Authors' Addresses . . . . .	21

## 1. Introduction

## 1.1. Problem Description

Internet Key Exchange Protocol (IKEv2) as specified in RFC 7296 [RFC7296] uses the Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH) algorithm to establish a shared secret between an initiator and a responder. The security of the DH and ECDH algorithms relies on the difficulty to solve a discrete logarithm



problem in multiplicative and elliptic curve groups respectively when the order of the group parameter is large enough. While solving such a problem remains difficult with current computing power, it is believed that general purpose quantum computers will be able to solve this problem, implying that the security of IKEv2 is compromised. There are, however, a number of cryptosystems that are conjectured to be resistant against quantum computer attack. This family of cryptosystems are known as post-quantum cryptography (PQC). It is sometimes also referred to as quantum-safe cryptography (QSC) or quantum-resistant cryptography (QRC).

## 1.2. Proposed Extension

This document describes a framework to integrate QSC for IKEv2, while maintaining backwards compatibility, to derive a set of IKE keys that have resistance to quantum computer attacks. Our framework allows the negotiation of one or more QSC algorithm to exchange data, in addition to the existing DH or ECDH key exchange data. We believe that the feature of using more than one post-quantum algorithm is important as many of these algorithms are relatively new and there may be a need to hedge the security risk with multiple key exchange data from several distinct QSC algorithms.

The secrets established from each key exchange are combined in a way such that should the post-quantum secrets not be present, the derived shared secret is equivalent to that of the standard IKEv2; on the other hand, a post-quantum shared secret is obtained if both classical and post-quantum key exchange data are present. This framework also applies to key exchanges in IKE Security Associations (SAs) for Encapsulating Security Payload (ESP) [RFC4303] or Authentication Header (AH) [RFC4302], i.e. Child SAs, in order to provide a stronger guarantee of forward security.

Some post-quantum key exchange payloads may have size larger than the standard maximum transmission unit (MTU) size, and therefore there could be issues with fragmentation at IP layer. IKE does allow transmission over TCP where fragmentation is not an issue [RFC8229]; however, we believe that a UDP-based solution will be required too. IKE does have a mechanism to handle fragmentation within UDP [RFC7383], however that is only applicable to messages exchanged after the IKE\_SA\_INIT. To use this mechanism, we use the IKE\_INTERMEDIATE exchange as outlined in [I-D.ietf-ipsecme-ikev2-intermediate]. With this mechanism, we do an initial key exchange, using a smaller, possibly non-quantum resistant primitive, such as ECDH. Then, before we do the IKE\_AUTH exchange, we perform one or more IKE\_INTERMEDIATE exchanges, each of which includes a secondary key exchange. As the IKE\_INTERMEDIATE exchange is encrypted, the IKE fragmentation protocol RFC7383 can be used.

The IKE SK\_\* values are updated after each exchange, and so the final IKE SK\_\* values depend on all the key exchanges, hence they are secure if any of the key exchanges are secure.

Note that readers should consider the approach in this document as providing a long term solution in upgrading the IKEv2 protocol to support post-quantum algorithms. A short term solution to make IKEv2 key exchange quantum secure is to use post-quantum pre-shared keys as discussed in [I-D.ietf-ipsecme-qr-ikev2].

Note also, that the proposed approach of performing multiple successive key exchanges in such a way that resulting session keys depend on all of them is not limited to achieving quantum resistance only. It can also be used when all the performed key exchanges are classical (EC)DH ones, but for some reasons (e.g. policy requirements) it is essential to perform multiple of them.

### 1.3. Changes

RFC EDITOR PLEASE DELETE THIS SECTION.

Changes in this draft in each version iterations.

draft-tjhai-ipsecme-hybrid-qske-ikev2-04

- o Clarification about key derivation in case of multiple key exchanges in CREATE\_CHILD\_SA is added.
- o Resolving rekey collisions in case of multiple key exchanges is clarified.

draft-tjhai-ipsecme-hybrid-qske-ikev2-03

- o Using multiple key exchanges CREATE\_CHILD\_SA is defined.

draft-tjhai-ipsecme-hybrid-qske-ikev2-02

- o Use new transform types to negotiate additional key exchanges, rather than using the KE payloads of IKE SA.

draft-tjhai-ipsecme-hybrid-qske-ikev2-01

- o Use IKE\_INTERMEDIATE to perform multiple key exchanges in succession.
- o Handle fragmentation by keeping the first key exchange (a standard IKE\_SA\_INIT with a few extra notifies) small, and encrypting the rest of the key exchanges.

- o Simplify the negotiation of the 'extra' key exchanges.

draft-tjhai-ipsecme-hybrid-qske-ikev2-00

- o We added a feature to allow more than one post-quantum key exchange algorithms to be negotiated and used to exchange a post-quantum shared secret.
- o Instead of relying on TCP encapsulation to deal with IP level fragmentation, we introduced a new key exchange payload that can be sent as multiple fragments within IKE\_SA\_INIT message.

#### 1.4. Document Organization

The remainder of this document is organized as follows. Section 2 summarizes design criteria. Section 3 describes how post-quantum key exchange is performed between two IKE peers and how keying materials are derived for both SAs and child SAs. A summary of alternative approaches that have been considered, but later discarded, are described in Appendix A. Section 4 discusses IANA considerations for the namespaces introduced in this document, and lastly Section 5 discusses security considerations.

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

## 2. Design Criteria

The design of the proposed post-quantum IKEv2 is driven by the following criteria:

- 1) Need for post-quantum cryptography in IPsec. Quantum computers might become feasible in the next 5-10 years. If current Internet communications are monitored and recorded today (D), the communications could be decrypted as soon as a quantum-computer is available (e.g., year Q) if key negotiation only relies on non post-quantum primitives. This is a high threat for any information that must remain confidential for a long period of time  $T > Q - D$ . The need is obvious if we assume that Q is 2040, D is 2020, and T is 30 years. Such a value of T is typical in classified or healthcare data.
- 2) Hybrid. Currently, there does not exist a post-quantum key exchange that is trusted at the level that ECDH is trusted against conventional (non-quantum) adversaries. A hybrid

approach allows introducing promising post-quantum candidates next to well-established primitives, since the overall security is at least as strong as each individual primitive.

- 3) Focus on quantum-resistant confidentiality. A passive attacker can eavesdrop on IPsec communication today and decrypt it once a quantum computer is available in the future. This is a very serious attack for which we do not have a solution. An attacker can only perform active attacks such as impersonation of the communicating peers once a quantum computer is available, sometime in the future. Thus, our design focuses on quantum-resistant confidentiality due to the urgency of this problem. This document does not address quantum-resistant authentication since it is less urgent at this stage.
- 4) Limit amount of exchanged data. The protocol design should be such that the amount of exchanged data, such as public-keys, is kept as small as possible even if initiator and responder need to agree on a hybrid group or multiple public-keys need to be exchanged.
- 5) Future proof. Any cryptographic algorithm could be potentially broken in the future by currently unknown or impractical attacks: quantum computers are merely the most concrete example of this. The design does not categorize algorithms as "post-quantum" or "non post-quantum" and does not create assumptions about the properties of the algorithms, meaning that if algorithms with different properties become necessary in the future, this framework can be used unchanged to facilitate migration to those algorithms.
- 6) Limited amount of changes. A key goal is to limit the number of changes required when enabling a post-quantum handshake. This ensures easier and quicker adoption in existing implementations.
- 7) Localized changes. Another key requirement is that changes to the protocol are limited in scope, in particular, limiting changes in the exchanged messages and in the state machine, so that they can be easily implemented.
- 8) Deterministic operation. This requirement means that the hybrid post-quantum exchange, and thus, the computed key, will be based on algorithms that both client and server wish to support.
- 9) Fragmentation support. Some PQC algorithms could be relatively bulky and they might require fragmentation. Thus, a design goal is the adaptation and adoption of an existing fragmentation

method or the design of a new method that allows for the fragmentation of the key shares.

- 10) Backwards compatibility and interoperability. This is a fundamental requirement to ensure that hybrid post-quantum IKEv2 and a non-post-quantum IKEv2 implementations are interoperable.
- 11) FIPS compliance. IPsec is widely used in Federal Information Systems and FIPS certification is an important requirement. However, algorithms that are believed to be post-quantum are not FIPS compliant yet. Still, the goal is that the overall hybrid post-quantum IKEv2 design can be FIPS compliant.

### 3. The Framework of Hybrid Post-Quantum Key Exchange

#### 3.1. Overall design

This design assigns new Transform Type 4 identifiers to the various post-quantum key exchanges (which will be defined later). We specifically do not make a distinction between classical (DH and ECDH) and post-quantum key exchanges, nor post-quantum algorithms which are true key exchanges versus post-quantum algorithms that act as key transport mechanisms; all are treated equivalently by the protocol. To make this more clear for implementers this document renames Transform Type 4 from "Diffie-Hellman Group Transform IDs" to "Key Exchange Method Transform IDs".

In order to support both hybrid key exchanges (that is, relying on distinct key exchanges) and fragmentation, the proposed hybrid post-quantum IKEv2 protocol extends IKE [RFC7296] by adding additional key exchange messages between the IKE\_SA\_INIT and the IKE\_AUTH exchanges by utilizing IKE\_INTERMEDIATE exchange described in [I-D.ietf-ipsecme-ikev2-intermediate].

In order to minimize communication overhead, only the key shares that are agreed to be used are actually exchanged. In order to achieve this several new Transform Types are defined, each sharing possible Transform IDs with Transform Type 4. The IKE\_SA\_INIT message includes one or more newly defined SA transforms that lists the extra key exchange policy required by the initiator; the responder selects single transform of each type, and returns them back in the response IKE\_SA\_INIT message. Then, provided that additional key exchanges are negotiated the initiator and the responder perform one or more IKE\_INTERMEDIATE exchanges; each such exchange includes a KE payload for one of the negotiated key exchanges.

Here is an overview of the initial exchanges:

Initiator

Responder

```
----->
<-- IKE_SA_INIT (additional key exchanges negotiation) -->
```

```
<-- {IKE_INTERMEDIATE (additional key exchange)} -->
```

```
...
```

```
<-- {IKE_INTERMEDIATE (additional key exchange)} -->
```

```
<-- {IKE_AUTH} -->
```

The extra post-quantum key exchanges can use algorithms that are currently considered to be resistant to quantum computer attacks. These algorithms are collectively referred to as post-quantum algorithms in this document.

Most post-quantum key agreement algorithms are relatively new, and thus are not fully trusted. There are also many proposed algorithms, with different trade-offs and relying on different hard problems. The concern is that some of these hard problems may turn out to be easier to solve than anticipated (and thus the key agreement algorithm not be as secure as expected). A hybrid solution allows us to deal with this uncertainty by combining a classical key exchanges with a post-quantum one, as well as leaving open the possibility of multiple post-quantum key exchanges.

The method that we use to perform hybrid key exchange also addresses the fragmentation issue. The initial IKE\_INIT messages do not have any inherent fragmentation support within IKE; however that can include a relatively short KE payload (e.g. one for group 14, 19 or 31). The rest of the KE payloads are encrypted within IKE\_INTERMEDIATE messages; because they are encrypted, the standard IKE fragmentation solution [RFC7383] is available.

### 3.2. Overall Protocol

In the simplest case, the initiator is happy with a single key exchange (and has no interest in supporting multiple), and he is not concerned with possible fragmentation of the IKE\_SA\_INIT messages (either because the key exchange he selects is small enough not to fragment, or he is confident that fragmentation will be handled either by IP fragmentation, or transport via TCP). In the following we overview the two protocol rounds involved in the hybrid post-quantum protocol.

In this case, the initiator performs the IKE\_SA\_INIT as standard, inserting a preferred key exchange (which is possibly a post-quantum

algorithm) as the listed Transform Type 4, and including the initiator KE payload. If the responder accepts the policy, he responds with an IKE\_SA\_INIT response, and IKE continues as usual.

If the initiator desires to negotiate multiple key exchanges, or he needs IKE to handle any possible fragmentation, then he uses the protocol listed below.

### 3.2.1. IKE\_SA\_INIT Round: Negotiation

Multiple key exchanges are negotiated using the standard IKEv2 mechanism, via SA payload. For this purpose several new transform types, namely Additional Key Exchange 1, Additional Key Exchange 2, Additional Key Exchange 3, etc., are defined. They are collectively called Additional Key Exchanges and have slightly different semantics than existing IKEv2 transform types. They are interpreted as additional key exchanges that peers agreed to perform in a series of IKE\_INTERMEDIATE exchanges. The possible transform IDs for these transform types are the same as IDs for the Transform Type 4, so they all share a single IANA registry for transform IDs.

Key exchange method negotiated via Transform Type 4 MUST always take place in the IKE\_SA\_INIT exchange. Additional key exchanges negotiated via newly defined transforms MUST take place in a series of IKE\_INTERMEDIATE exchanges, in an order of the values of their transform types, so that key exchange negotiated using Transform Type N always precedes that of Transform Type N + 1. Each IKE\_INTERMEDIATE exchange MUST bear exactly one key exchange method. Note that with this semantics, Additional Key Exchanges transforms are not associated with any particular type of key exchange and don't have any specific per transform type transform IDs IANA registry. Instead they all share a single registry for transform IDs - "Key Exchange Method Transform IDs", as well as Transform Type 4. All new key exchange algorithms (both classical or quantum safe) should be added to this registry. This approach gives peers flexibility in defining the ways they want to combine different key exchange methods.

When forming a proposal the initiator adds transforms for the IKE\_SA\_INIT exchange using Transform Type 4. In most cases they will contain classical key exchange methods, however it is not a requirement. Additional key exchange methods are proposed using Additional Key Exchanges transform types. All these transform types are optional, the initiator is free to select any of them for proposing additional key exchange methods. Consequently, if none of Additional Key Exchange transforms are included in the proposal, then this proposal indicates performing standard IKEv2, as defined in [RFC7296]. If the initiator includes any transform of type N (where

N is among Additional Key Exchanges) in the proposal, the responder MUST select one of the algorithms proposed using this type. A transform ID NONE may be added to those transform types which contain key exchange methods that the initiator believes are optional.

The responder performs negotiation using standard IKEv2 procedure described in Section 3.3 of [RFC7296]. However, for the Additional Key Exchange types the responder's choice MUST NOT contain equal transform IDs (apart from NONE), and the ID selected for Transform Type 4 MUST NOT appear in any of Additional Key Exchange transforms. In other words, all selected key exchange methods must be different.

### 3.2.2. IKE\_INTERMEDIATE Round: Additional Key Exchanges

For each extra key exchange agreed to in the IKE\_SA\_INIT exchange, the initiator and the responder perform one or more IKE\_INTERMEDIATE exchanges, as described in [I-D.ietf-ipsecme-ikev2-intermediate].

These exchanges are as follows:

Initiator	Responder
-----	
HDR, SK {Ni(n), KEi(n)}	-->
	<-- HDR, SK {Nr(n), KER(n)}

The initiator sends a nonce in the Ni(n) payload, and the key exchange payload in the KEi(n). This packet is encrypted with the current IKE SK\_\* keys.

On receiving this, the responder sends a nonce in the Nr(n) payload, and the key exchange payload KER(n); again, this packet is encrypted with the current IKE SA keys.

The Diffie-Hellman Group Num field in the KEi(n) and KER(n) payloads MUST match the n-th negotiated extra key exchange. Note that the negotiated transform types (the encryption type, hash type, prf type) are not modified.

Once this exchange is done, then both sides compute an updated keying material:

$$\text{SKEYSEED}(n) = \text{prf}(\text{SK}_d(n-1), \text{KE}(n) \mid \text{Ni}(n) \mid \text{Nr}(n))$$

where KE(n) is the resulting shared secret of this key exchange and SK\_d(n-1) is the last generated SK\_d, (derived from the previous IKE\_INTERMEDIATE exchange, or the IKE\_SA\_INIT if there haven't already been any IKE\_INTERMEDIATE exchanges). Then, SK\_d, SK\_ai, SK\_ar, SK\_ei, SK\_er, SK\_pi, SK\_pr are updated as:



$$\{SK\_d(n) \mid SK\_ai(n) \mid SK\_ar(n) \mid SK\_ei(n) \mid SK\_er(n) \mid SK\_pi(n) \mid SK\_pr(n)\} = \text{prf+} (SKEYSEED(n), Ni(n) \mid Nr(n) \mid SPIi \mid SPIr)$$

Both the initiator and the responder use this updated key values in the next exchange.

### 3.2.3. IKE\_AUTH Exchange

After all IKE\_INTERMEDIATE exchanges have completed, the initiator and the responder perform an IKE\_AUTH exchange. This exchange is the standard IKE exchange, except that the initiator and responder signed octets are modified as described in [I-D.ietf-ipsecme-ikev2-intermediate].

Note, that despite the fact, that a fresh pair of nonces is exchanged in each IKE\_INTERMEDIATE exchange, only nonces from the IKE\_SA\_INIT are included in calculation of AUTH payload (see Section 2.15 of [RFC7296]).

### 3.2.4. CREATE\_CHILD\_SA Exchange

The CREATE\_CHILD\_SA exchange is used in IKEv2 for the purpose of creating additional Child SAs, rekeying them and rekeying IKE SA itself. When creating or rekeying Child SAs, the peers may optionally perform a Diffie-Hellmann key exchange to add a fresh entropy into the session keys. In case of IKE SA rekey, the key exchange is mandatory.

If the IKE SA was created using multiple key exchange methods, the peers may want continue using multiple key exchanges in the CREATE\_CHILD\_SA exchange too. If the initiator includes any Additional Key Exchanges transform in the SA payload (along with Transform Type 4) and the responder agrees to perform additional key exchanges, then the additional key exchanges are performed in a series of the INFORMATIONAL exchanges that follows the CREATE\_CHILD\_SA exchange. These key exchanges are performed in an order of the values of their transform types, so that key exchange negotiated using Transform Type N always precedes key exchange negotiated using Transform Type N + 1. Each INFORMATIONAL exchange MUST bear exactly one key exchange method. Key exchange negotiated via Transform Type 4 always takes place in the CREATE\_CHILD\_SA exchange, as per IKEv2 specification.

Since after IKE SA is created the window size may be greater than one and multiple concurrent exchanges may be active, it is essential to link the INFORMATIONAL exchanges together and with the corresponding CREATE\_CHILD\_SA exchange. A new status type notification ADDITIONAL\_KEY\_EXCHANGE is used for this purpose. Its Notify Message

Type is <TBA by IANA>, Protocol ID and SPI Size are both set to 0. The data associated with this notification is a blob meaningful only to the responder, so that the responder can correctly link successive exchanges. For the initiator the content of this notification is an opaque blob.

The responder MUST include this notification in a CREATE\_CHILD\_SA or INFORMATIONAL response message in case next exchange is expected, filling it with some data that would allow linking this exchange to the next one. The initiator MUST copy the received notification with its content intact into the request message of the next exchange.

Below is an example of three additional key exchanges.

Initiator	Responder
-----	
HDR(CREATE_CHILD_SA), SK {SA, Ni, KEi} -->	<-- HDR(CREATE_CHILD_SA), SK {SA, Nr, KEr, N(ADDITIONAL_KEY_EXCHANGE) (link1)}
HDR(INFORMATIONAL), SK {Ni2, KEi2, N(ADDITIONAL_KEY_EXCHANGE) (link1)} -->	<-- HDR(INFORMATIONAL), SK {Nr2, KEr2, N(ADDITIONAL_KEY_EXCHANGE) (link2)}
HDR(INFORMATIONAL), SK {Ni3, KEi3, N(ADDITIONAL_KEY_EXCHANGE) (link2)} -->	<-- HDR(INFORMATIONAL), SK {Nr3, KEr3, N(ADDITIONAL_KEY_EXCHANGE) (link3)}
HDR(INFORMATIONAL), SK {Ni4, KEi4, N(ADDITIONAL_KEY_EXCHANGE) (link3)} -->	<-- HDR(INFORMATIONAL), SK {Nr4, KEr4}

It is possible that due to some unexpected events (e.g. reboot) the Initiator could forget that he/she is in the process of performing additional key exchanges and never starts next INFORMATIONAL exchanges. The Responder MUST handle this situation gracefully and delete the associated state if he/she doesn't receive the next expected INFORMATIONAL request after some reasonable period of time.

If Responder receives INFORMATIONAL request containing ADDITIONAL\_KEY\_EXCHANGE notification and the content of this notify doesn't correspond to any active key exchange state the Responder has, he/she MUST send back a new error type notification STATE\_NOT\_FOUND. This is a non-fatal notification, its Notify Message Type is <TBA by IANA>, Protocol ID and SPI Size are both set to 0 and the data is empty. If Initiator receives this notification

in response to INFORMATIONAL exchange performing additional key exchange, he/she MUST cancel this exchange and MUST treat the whole series of exchanges started from the CREATE\_CHILD\_SA exchange as failed. In most cases, the receipt of this notification is caused by premature deletion of the corresponding state on the Responder (the time period between INFORMATIONAL exchanges appeared too long from Responder's point of view, e.g. due to a temporary network failure). After receiving this notification the Initiator MAY start a new CREATE\_CHILD\_SA exchange (eventually followed by the INFORMATIONAL exchanges) to retry the failed attempt. If the Initiator continues to receive STATE\_NOT\_FOUND notifications after several retries, he/she MUST treat it as fatal error and delete IKE SA (sending DELETE payload).

When rekeying IKE SA or Child SA it is possible that the peers start doing this at the same time, which is called simultaneous rekeying. Sections 2.8.1 and 2.8.2 of [RFC7296] describes how IKEv2 handles this situation. In a nutshell IKEv2 follows the rule that if in case of simultaneous rekeying two identical new IKE SAs (or two pairs of Child SAs) are created, then one of them should be deleted. Which one is to be deleted is determined by comparing the values of four nonces, that were used in the colliding CREATE\_CHILD\_SA exchanges - the IKE SA (or pair of Child SAs) that was created by the exchange in which the smallest nonce was used should be deleted by the initiator of this exchange.

With multiple key exchanges the SAs are not yet created once the CREATE\_CHILD\_SA is completed, they would be created only after the series of INFORMATIONAL exchanges is finished. For this reason if additional key exchanges were negotiated in the CREATE\_CHILD\_SA initiated by the losing side, there is nothing to delete and this side just stops the rekeying process - he/she MUST not initiate INFORMATIONAL exchange with next key exchange.

In most cases rekey collisions are resolved in the CREATE\_CHILD\_SA exchange. However, a situation may occur when due to packet loss one of the peers receives CREATE\_CHILD\_SA message requesting rekeying SA that is already being rekeyed by this peer (i.e. the CREATE\_CHILD\_SA exchange initiated by this peer has been already completed and the series of INFORMATIONAL exchanges is in progress). In this case TEMPORARY\_FAILURE notification MUST be sent in response to such request.

If multiple key exchanges were negotiated in the CREATE\_CHILD\_SA exchange, then the resulting keys are computed as follows. In case of IKE SA rekey:

$$\text{SKEYSEED} = \text{prf}(\text{SK\_d}, \text{KE} \mid \text{Ni} \mid \text{Nr} \mid \begin{array}{c} \text{KE}(1) \\ \text{KE}(n) \end{array} \mid \begin{array}{c} \text{Ni}(1) \\ \text{Ni}(n) \end{array} \mid \begin{array}{c} \text{Nr}(1) \\ \text{Nr}(n) \end{array} \dots)$$

In case of Child SA creating or rekey:

$$\text{KEYMAT} = \text{prf+}(\text{SK\_d}, \text{KE} \mid \text{Ni} \mid \text{Nr} \mid \begin{array}{c} \text{KE}(1) \\ \text{KE}(n) \end{array} \mid \begin{array}{c} \text{Ni}(1) \\ \text{Ni}(n) \end{array} \mid \begin{array}{c} \text{Nr}(1) \\ \text{Nr}(n) \end{array} \dots)$$

In both cases SK\_d is from existing IKE SA; KE, Ni, Nr - shared key and nonces from the CREATE\_CHILD\_SA; KE(1)..KE(n), Ni(1)..Ni(n), Nr(1)..Nr(n) - shared keys and nonces from additional key exchanges.

#### 4. IANA Considerations

This document renames "Transform Type 4 - Diffie-Hellman Group Transform IDs" to "Transform Type 4 - Key Exchange Method Transform IDs"

This document also adds the following Transform Types to the "Transform Type Values" registry:

Type	Description	Used In	Reference
6	Additional Key Exchange 1	(optional in IKE, AH and ESP)	[RFCXXXX]
7	Additional Key Exchange 2	(optional in IKE, AH and ESP)	[RFCXXXX]
8	Additional Key Exchange 3	(optional in IKE, AH and ESP)	[RFCXXXX]
9	Additional Key Exchange 4	(optional in IKE, AH and ESP)	[RFCXXXX]
10	Additional Key Exchange 5	(optional in IKE, AH and ESP)	[RFCXXXX]
11	Additional Key Exchange 6	(optional in IKE, AH and ESP)	[RFCXXXX]
12	Additional Key Exchange 7	(optional in IKE, AH and ESP)	[RFCXXXX]

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

<TBA>            ADDITIONAL\_KEY\_EXCHANGE

and a new Notify Message Type in the "Notify Message Types - Error Types" registry:

<TBA>            STATE\_NOT\_FOUND

#### 5. Security Considerations

The key length of the Encryption Algorithm (Transform Type 1), the Pseudorandom Function (Transform Type 2) and the Integrity Algorithm (Transform Type 3), all have to be of sufficient length to prevent attacks using Grover's algorithm [GROVER]. In order to use the extension proposed in this document, the key lengths of these

transforms SHALL be at least 256 bits long in order to provide sufficient resistance to quantum attacks. Accordingly the post-quantum security level achieved is at least 128 bits.

SKEYSEED is calculated from shared, KEx, using an algorithm defined in Transform Type 2. While a quantum attacker may learn the value of KEx', if this value is obtained by means of a classical key exchange, other KEx values generated by means of a quantum-resistant algorithm ensure that the final SKEYSEED is not compromised. This assumes that the algorithm defined in the Transform Type 2 is post-quantum.

The main focus of this document is to prevent a passive attacker performing a "harvest and decrypt" attack. In other words, an attacker that records messages exchanges today and proceeds to decrypt them once he owns a quantum computer. This attack is prevented due to the hybrid nature of the key exchange. Other attacks involving an active attacker using a quantum-computer are not completely solved by this document. This is for two reasons.

The first reason is because the authentication step remains classical. In particular, the authenticity of the SAs established under IKEv2 is protected using a pre-shared key, RSA, DSA, or ECDSA algorithms. Whilst the pre-shared key option, provided the key is long enough, is post-quantum, the other algorithms are not. Moreover, in implementations where scalability is a requirement, the pre-shared key method may not be suitable. Quantum-safe authenticity may be provided by using a quantum-safe digital signature and several quantum-safe digital signature methods are being explored by IETF. For example, if the implementation is able to reliably track state, the hash based method, XMSS has the status of an RFC, see [RFC8391]. Currently, quantum-safe authentication methods are not specified in this document, but are planned to be incorporated in due course.

It should be noted that the purpose of post-quantum algorithms is to provide resistance to attacks mounted in the future. The current threat is that encrypted sessions are subject to eavesdropping and archived with decryption by quantum computers taking place at some point in the future. Until quantum computers become available there is no point in attacking the authenticity of a connection because there are no possibilities for exploitation. These only occur at the time of the connection, for example by mounting a MitM attack. Consequently there is not such a pressing need for quantum-safe authenticity.

This draft does not attempt to address key exchanges with KE payloads longer than 64k; the current IKE payload format does not allow that as a possibility. If such huge KE payloads are required, a work around (such as making the KE payload a URL and a hash of the real

payload) would be needed. At the current time, it appears likely that there will be plenty of key exchanges available that would not require such a workaround.

## 6. Acknowledgements

The authors would like to thanks Frederic Detienne and Olivier Pelerin for their comments and suggestions, including the idea to negotiate the post-quantum algorithms using the existing KE payload. The authors are also grateful to Tobias Heider and Tobias Guggemos for valuable comments.

## 7. References

### 7.1. Normative References

- [I-D.ietf-ipsecme-ikev2-intermediate]  
Smyslov, V., "Intermediate Exchange in the IKEv2 Protocol", draft-ietf-ipsecme-ikev2-intermediate-00 (work in progress), June 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 7.2. Informative References

- [GROVER] Grover, L., "A Fast Quantum Mechanical Algorithm for Database Search", Proc. of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996), 1996.
- [I-D.ietf-ipsecme-qr-ikev2]  
Fluhrer, S., McGrew, D., Kampanakis, P., and V. Smyslov, "Postquantum Preshared Keys for IKEv2", draft-ietf-ipsecme-qr-ikev2-08 (work in progress), March 2019.

- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <<https://www.rfc-editor.org/info/rfc7383>>.
- [RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <<https://www.rfc-editor.org/info/rfc8229>>.
- [RFC8391] Huelsing, A., Butin, D., Gazdag, S., Rijneveld, J., and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, DOI 10.17487/RFC8391, May 2018, <<https://www.rfc-editor.org/info/rfc8391>>.

#### Appendix A. Alternative Design

This section gives an overview on a number of alternative approaches that we have considered, but later discarded. These approaches are:

- o Sending the classical and post-quantum key exchanges as a single transform

We considered combining the various key exchanges into a single large KE payload; this effort is documented in a previous version of this draft (draft-tjhai-ipsecme-hybrid-qske-ikev2-01). This does allow us to cleanly apply hybrid key exchanges during the child SA; however it does add considerable complexity, and requires an independent fragmentation solution.

- o Sending post-quantum proposals and policies in KE payload only

With the objective of not introducing unnecessary notify payloads, we considered communicating the hybrid post-quantum proposal in the KE payload during the first pass of the protocol exchange. Unfortunately, this design is susceptible to the following downgrade attack. Consider the scenario where there is an MitM attacker sitting between an initiator and a responder. The initiator proposes, through SAI payload, to use a hybrid post-quantum group and as a backup a Diffie-Hellman group, and through KEi payload, the initiator proposes a list of hybrid post-quantum

proposals and policies. The MitM attacker intercepts this traffic and replies with N(INVALID\_KEY\_PAYLOAD) suggesting to downgrade to the backup Diffie-Hellman group instead. The initiator then resends the same SAI payload and the KEI payload containing the public value of the backup Diffie-Hellman group. Note that the attacker may forward the second IKE\_SA\_INIT message only to the responder, and therefore at this point in time, the responder will not have the information that the initiator prefers the hybrid group. Of course, it is possible for the responder to have a policy to reject an IKE\_SA\_INIT message that (a) offers a hybrid group but not offering the corresponding public value in the KEI payload; and (b) the responder has not specifically acknowledged that it does not supported the requested hybrid group. However, the checking of this policy introduces unnecessary protocol complexity. Therefore, in order to fully prevent any downgrade attacks, using KE payload alone is not sufficient and that the initiator MUST always indicate its preferred post-quantum proposals and policies in a notify payload in the subsequent IKE\_SA\_INIT messages following a N(INVALID\_KEY\_PAYLOAD) response.

- o New payload types to negotiate hybrid proposal and to carry post-quantum public values

Semantically, it makes sense to use a new payload type, which mimics the SA payload, to carry a hybrid proposal. Likewise, another new payload type that mimics the KE payload, could be used to transport hybrid public value. Although, in theory a new payload type could be made backwards compatible by not setting its critical flag as per Section 2.5 of RFC7296, we believe that it may not be that simple in practice. Since the original release of IKEv2 in RFC4306, no new payload type has ever been proposed and therefore, this creates a potential risk of having a backward compatibility issue from non-conforming RFC IKEv2 implementations. Since we could not see any other compelling advantages apart from a semantic one, we use the existing transform type and notify payloads instead. In fact, as described above, we use the KE payload in the first IKE\_SA\_INIT request round and the notify payload to carry the post-quantum proposals and policies. We use one or more of the existing KE payloads to carry the hybrid public values.

- o Hybrid public value payload

One way to transport the negotiated hybrid public payload, which contains one classical Diffie-Hellman public value and one or more post-quantum public values, is to bundle these into a single KE payload. Alternatively, these could also be transported in a single new hybrid public value payload, but following the same



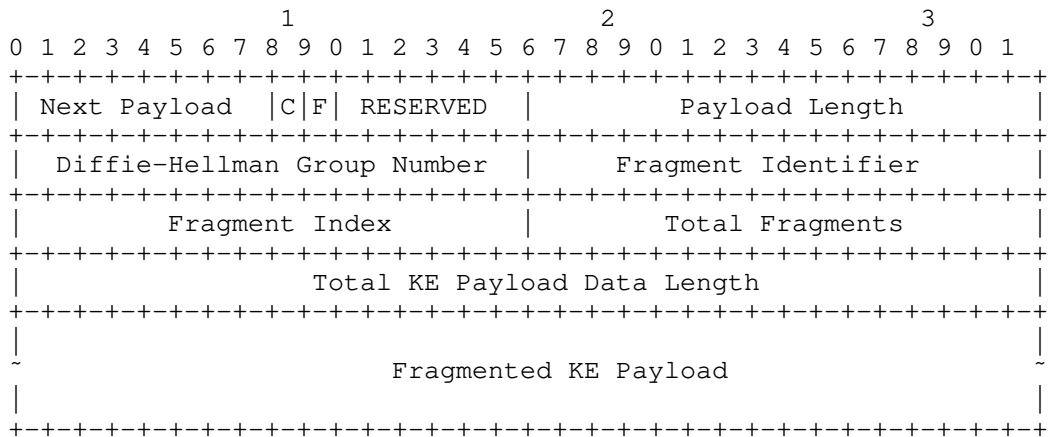
reasoning as above, this may not be a good idea from a backward compatibility perspective. Using a single KE payload would require an encoding or formatting to be defined so that both peers are able to compose and extract the individual public values. However, we believe that it is cleaner to send the hybrid public values in multiple KE payloads--one for each group or algorithm. Furthermore, at this point in the protocol exchange, both peers should have indicated support of handling multiple KE payloads.

- o Fragmentation

Handling of large IKE\_SA\_INIT messages has been one of the most challenging tasks. A number of approaches have been considered and the two prominent ones that we have discarded are outlined as follows.

The first approach was to treat the entire IKE\_SA\_INIT message as a stream of bytes, which we then split it into a number of fragments, each of which is wrapped onto a payload that would fit into the size of the network MTU. The payload that wraps each fragment is a new payload type and it was envisaged that this new payload type will not cause a backward compatibility issue because at this stage of the protocol, both peers should have indicated support of fragmentation in the first pass of the IKE\_SA\_INIT exchange. The negotiation of fragmentation is performed using a notify payload, which also defines supporting parameters such as the size of fragment in octets and the fragment identifier. The new payload that wraps each fragment of the messages in this exchange is assigned the same fragment identifier. Furthermore, it also has other parameters such as a fragment index and total number of fragments. We decided to discard this approach due to its blanket approach to fragmentation. In cases where only a few payloads need to be fragmented, we felt that this approach is overly complicated.

Another idea that was discarded was fragmenting an individual payload without introducing a new payload type. The idea was to use the 9-th bit (the bit after the critical flag in the RESERVED field) in the generic payload header as a flag to mark that this payload is fragmented. As an example, if a KE payload is to be fragmented, it may look as follows.



When the flag F is set, this means the current KE payload is a fragment of a larger KE payload. The Payload Length field denotes the size of this payload fragment in octets--including the size of the generic payload header. The two-octet RESERVED field following Diffie-Hellman Group Number was to be used as a fragment identifier to help assembly and disassembly of fragments. The Fragment Index and Total Fragments fields are self-explanatory. The Total KE Payload Data Length indicates the size of the assembled KE payload data in octets. Finally, the actual fragment is carried in Fragment KE Payload field.

We discarded this approach because we believe that the working group may not be happy using the RESERVED field to change the format of a packet and that implementers may not like the complexity added from checking the fragmentation flag in each received payload. More importantly, fragmenting the messages in this way may leave the system to be more prone to denial of service (DoS) attacks. By using IKE\_INTERMEDIATE to transport the large post-quantum key exchange payloads, there is no longer any issue with fragmentation.

- o Group sub-identifier

As discussed before, each group identifier is used to distinguish a post-quantum algorithm. Further classification could be made on a particular post-quantum algorithm by assigning additional value alongside the group identifier. This sub- identifier value may be used to assign different security parameter sets to a given post-quantum algorithm. However, this level of details does not fit the principles of the document where it should deal with generic hybrid key exchange protocol, not a specific ciphersuite.

Furthermore, there are enough Diffie- Hellman group identifiers should this be required in the future.

#### Authors' Addresses

C. Tjhai  
Post-Quantum

Email: [cjt@post-quantum.com](mailto:cjt@post-quantum.com)

M. Tomlinson  
Post-Quantum

Email: [mt@post-quantum.com](mailto:mt@post-quantum.com)

G. Bartlett  
Cisco Systems

Email: [grbartle@cisco.com](mailto:grbartle@cisco.com)

S. Fluhrer  
Cisco Systems

Email: [sfluhrer@cisco.com](mailto:sfluhrer@cisco.com)

D. Van Geest  
ISARA Corporation

Email: [daniel.vangeest@isara.com](mailto:daniel.vangeest@isara.com)

O. Garcia-Morchon  
Philips

Email: [oscar.garcia-morchon@philips.com](mailto:oscar.garcia-morchon@philips.com)

Valery Smyslov  
ELVIS-PLUS

Email: [svan@elvis.ru](mailto:svan@elvis.ru)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2017

B. Weis  
Cisco Systems  
Y. Nir  
Check Point Software Technologies Ltd.  
V. Smyslov  
ELVIS-PLUS  
March 9, 2017

Group Key Management using IKEv2  
draft-yeung-g-ikev2-11

Abstract

This document presents a new group key distribution protocol. The protocol is in conformance with MSEC key management architecture it contains two components: member registration and group rekeying, both downloading group security associations from the Group Controller/Key Server to a member of the group. The new protocol is similar to IKEv2 in message and payload formats as well as message semantics.

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 September 10, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction and Overview . . . . .	3
1.1. Requirements Language . . . . .	4
1.2. Relationship to GDOI . . . . .	4
1.3. G-IKEv2 Payloads . . . . .	4
2. G-IKEv2 integration into IKEv2 protocol . . . . .	5
2.1. UDP port . . . . .	5
3. G-IKEv2 Protocol . . . . .	5
3.1. G-IKEv2 member registration and secure channel establishment . . . . .	5
3.1.1. GSA_AUTH exchange . . . . .	6
3.1.2. GSA_REGISTRATION Exchange . . . . .	7
3.1.3. IKEv2 Header Initialization . . . . .	8
3.1.4. GM Registration Operations . . . . .	8
3.1.5. GCKS Registration Operations . . . . .	9
3.2. Counter-based modes of operation . . . . .	10
3.3. G-IKEv2 group maintenance channel . . . . .	12
3.3.1. G-IKEv2 GSA_REKEY exchange . . . . .	12
3.3.2. Forward and Backward Access Control . . . . .	14
3.3.3. Forward Access Control Requirements . . . . .	14
3.3.4. Deletion of SAs . . . . .	15
3.3.5. GSA_REKEY GCKS Operations . . . . .	15
3.3.6. GSA_REKEY GM Operations . . . . .	16
4. Header and Payload Formats . . . . .	17
4.1. The G-IKEv2 Header . . . . .	17
4.2. Group Identification (IDg) Payload . . . . .	17
4.3. Security Association - GM Supported Transforms (SAg) . . . . .	17
4.4. Group Security Association Payload . . . . .	18
4.4.1. GSA Policy . . . . .	18
4.5. KEK Policy . . . . .	19
4.5.1. KEK Attributes . . . . .	20
4.5.2. KEK_MANAGEMENT_ALGORITHM . . . . .	21
4.5.3. KEK_ENCR_ALGORITHM . . . . .	21
4.5.4. KEK_KEY_LENGTH . . . . .	22
4.5.5. KEK_KEY_LIFETIME . . . . .	22
4.5.6. KEK_INTEGRITY_ALGORITHM . . . . .	22
4.5.7. KEK_AUTH_METHOD . . . . .	22
4.5.8. KEK_AUTH_HASH . . . . .	22
4.5.9. KEK_MESSAGE_ID . . . . .	23
4.6. GSA TEK Policy . . . . .	23
4.6.1. TEK ESP and AH Protocol-Specific Policy . . . . .	24
4.7. GSA Group Associated Policy . . . . .	25

4.7.1.	ACTIVATION_TIME_DELAY/DEACTIVATION_TIME_DELAY . . . .	26
4.8.	Key Download Payload . . . . .	27
4.8.1.	TEK Download Type . . . . .	28
4.8.2.	KEK Download Type . . . . .	29
4.8.3.	LKH Download Type . . . . .	30
4.8.4.	SID Download Type . . . . .	34
4.9.	Delete Payload . . . . .	35
4.10.	Notify Payload . . . . .	35
4.11.	Authentication Payload . . . . .	36
5.	Security Considerations . . . . .	36
5.1.	GSA registration and secure channel . . . . .	36
5.2.	GSA maintenance channel . . . . .	36
5.2.1.	Authentication/Authorization . . . . .	36
5.2.2.	Confidentiality . . . . .	37
5.2.3.	Man-in-the-Middle Attack Protection . . . . .	37
5.2.4.	Replay/Reflection Attack Protection . . . . .	37
6.	IANA Considerations . . . . .	37
6.1.	New registries . . . . .	37
6.2.	New payload and exchange types to existing IKEv2 registry	38
7.	Acknowledgements . . . . .	38
8.	Contributors . . . . .	38
9.	References . . . . .	39
9.1.	Normative References . . . . .	39
9.2.	Informative References . . . . .	39
Appendix A.	Differences between G-IKEv2 and RFC 6407 . . . . .	41
Authors' Addresses	. . . . .	41

## 1. Introduction and Overview

This document presents a group key management protocol protected by IKEv2. The data communications within the group are protected by a key pushed to the group members (GMs) by the Group Controller/Key Server (GCKS) using IKEv2 [RFC7296]. The GCKS pushes policy and keys for the group to the GM after authenticating it using new payloads included in a new exchange called GSA\_AUTH (similar to the IKE\_AUTH exchange). This document references IKEv2 [RFC7296] but it intended to be a separate document. GDOI update document [RFC6407] presented GDOI using IKEv1 syntax. This document uses IKEv2 syntax. The message semantics of IKEv2 are preserved, in that all communications consists of message request-response pairs. The exception to this rule are the rekeying messages, which are sent in multicast without a response. A number of payloads were deemed unnecessary since [RFC6407] are described in Appendix A

### 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].

### 1.2. Relationship to GDOI

GDOI protocol specified in [RFC6407] is protected by IKEv1 phase1 security association defined in [RFC2407], [RFC2408] and [RFC2409]; these documents are obsoleted and replaced by a new version of the IKE protocol defined in RFC 7296. G-IKEv2 provides group key management between the Group Member and GCKS using the new IKEv2 protocol and inherits the following key advantages over GDOI:

1. Provide a simple mechanism for the responder to keep minimal state and avoid DoS attack from forged IP address using cookie challenge exchange.
2. Improve performance and network latency by the reduced number of initial messages to complete the G-IKEv2 protocol from (10 messages in Main mode and Quick mode, 7 messages in Aggressive mode and Quick) to 4 messages.
3. Fix cryptographic weakness with authentication HASH (IKEv1 authentication HASH specified in RFC 2409 does not include all ISAKMP payloads and does not include ISAKMP header). This issue is documented at [IKE-HASH].
4. Improve protocol reliability where all unicast messages are acknowledged and sequenced.
5. Well defined behavior for error conditions to improve interoperability.

### 1.3. G-IKEv2 Payloads

1. IDg (group ID) - The GM requests the GCKS for membership into the group by sending its IDg payload.
2. GSA (Group Security Association) - The GCKS sends the group policy to the GM using this payload.
3. KD (Key Download) - The GCKS sends the control and data keys to the GM using the KD payload.

## 2. G-IKEv2 integration into IKEv2 protocol

The G-IKEv2 protocol provides the security mechanisms of IKEv2 (peer authentication, confidentiality, message integrity) to protect the group negotiations required for G-IKEv2. The G-IKEv2 exchange further provides group authorization, and secure policy and key download from the GCKS to its group members.

It is assumed that readers are familiar with the IKEv2 protocol, so this document skips many details that are described in [RFC7296].

### 2.1. UDP port

G-IKEv2 SHOULD use port 848, the same as GDOI [RFC6407], because they serve a similar function, and can use the same ports, just as IKEv1 and IKEv2 can share port 500. The version number in the IKEv2 header distinguishes the G-IKEv2 protocol from GDOI protocol [RFC6407].

## 3. G-IKEv2 Protocol

### 3.1. G-IKEv2 member registration and secure channel establishment

The registration protocol consists of minimum two exchanges IKE\_SA\_INIT and GSA\_AUTH; member registration may have a few more messages exchanged if the EAP method, cookie challenge (for DoS protection) or negotiation of Diffie-Hellman group is included. Each exchange consists of request/response pairs. The first exchange IKE\_SA\_INIT is defined in IKEv2 [RFC7296]. This exchange negotiates cryptographic algorithms, exchanges nonces and does a Diffie-Hellman exchange between the group member (GM) and the Group Controller/Key Server (GCKS).

The second exchange GSA\_AUTH authenticates the previous messages, exchange identities and certificates. These messages are encrypted and integrity protected with keys established through the IKE\_SA\_INIT exchange, so the identities are hidden from eavesdroppers and all fields in all the messages are authenticated. The GCKS SHOULD authorize group members to be allowed into the group as part of the GSA\_AUTH exchange. Once the GCKS accepted a group member to join a group it will download the data security keys (TEKs) and/or group key encrypting key (KEK) or KEK array as part of GSA\_AUTH response message. In the following descriptions, the payloads contained in the message are indicated by names as listed below.



Notation	Payload
AUTH	Authentication
CERT	Certificate
CERTREQ	Certificate Request
GSA	Group Security Association
HDR	IKEv2 Header
IDg	Identification - Group
Idi	Identification - Initiator
IDr	Identification - Responder
KD	Key Download
KE	Key Exchange
Ni, Nr	Nonce
SA	Security Association
SAg	Security Association - GM Supported Transforms

The details of the contents of each payload are described in Section 4. Payloads that may optionally appear will be shown in brackets, such as [ CERTREQ ], to indicate that optionally a certificate request payload can be included.

### 3.1.1. GSA\_AUTH exchange

After the group member and GCKS uses IKE\_SA\_INIT exchange to negotiate cryptographic algorithms, exchange nonces, and perform a Diffie-Hellman exchange as defined in IKEv2 [RFC7296], the GSA\_AUTH MUST complete before any other exchanges can be done. The security properties of the GSA\_AUTH exchange are the same as the properties of the IKE\_AUTH exchange. It is used to authenticate the IKE\_SA\_INIT messages, exchange identities and certificates. G-IKEv2 also uses this exchange for group member registration and authorization. Although IKE\_AUTH contains SA2, TSi, and TSr payload the GSA\_AUTH does not contain them. They are not needed because policy is not negotiated between group member and GCKS, but instead downloaded from the GCKS to the group member.

Initiator (Member)	Responder (GCKS)
HDR, SK { Idi, [CERT,] [CERTREQ, ] [IDr, ] AUTH, IDg, [SAg, ] [N ] }	-->

After an unauthenticated secure channel is established by IKE\_SA\_INIT exchange, the member initiates a registration request to join a group indicated by the IDg payload. The GM MAY include an SAg payload declaring which Transforms that it is willing to accept, and also MAY include the Notify payload status type SENDER\_ID\_REQUEST to request SIDs for Counter-based cipher from the GCKS.

```
<-- HDR, SK { IDr, [CERT, ] AUTH, [ GSA, KD, ] [D, ] }
```

The GCKS responds with IDr, optional CERT, and AUTH material as if it were an IKE\_AUTH. It also informs the member the cryptographic policies of the group in the GSA payload and key material in the KD payload. The GCKS can also include Delete (D) payload instructing the group member to delete existing SAs it might have as the result of a previous group member registration.

In addition to the IKEv2 error handling, GCKS can reject the registration request when IDg is invalid or authorization fail, etc. In these cases, see Section 4.10, the GSA\_AUTH response will not include the GSA and KD, but will include a Notify payload indicating errors. If the group member included an SAg payload, and the GCKS chooses to evaluate it, and it detects that group member cannot support the security policy defined for the group, then the GCKS SHOULD return a NO\_PROPOSAL\_CHOSEN. When the GCKS indicates errors, and the group member cannot resolve the errors, the group member MUST delete the registration IKE SA.

Initiator (Member)	Responder (GCKS)
-----	-----
	<-- HDR, SK { N }

When the group member found the policy sent by the GCKS is unacceptable, the member SHOULD notify the GCKS by sending IDg and the Notify type NO\_PROPOSAL\_CHOSEN as shown below.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK {IDg [N,]}	-->
	<-- HDR, SK {}

### 3.1.2. GSA\_REGISTRATION Exchange

When a secure channel is already established between GM and GCKS, the GM registration for a group can reuse the established secure channel. In this scenario the GM will use the GSA\_REGISTRATION exchange by including the desired group ID (IDg) to request data security keys (TEKs) and/or group key encrypting keys (KEKs) from the GCKS. If the group member includes an SAg payload, and the GCKS chooses to evaluate it, and it detects that group member cannot support the security policy defined for the group, then the GCKS SHOULD return a NO\_PROPOSAL\_CHOSEN. The GM MAY also include the Notify payload status type SENDER\_ID\_REQUEST to request SIDs for Counter-based cipher from the GCKS. The GCKS response payloads are created and processed as in the GSA\_AUTH reply.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK {IDg, [SAg, ][N ] } -->	
	<-- HDR, SK { GSA, KD, [D ] }

This exchange can also be used when the group member found the policy sent by the GCKS is unacceptable. The group member SHOULD notify the GCKS by sending IDg and the Notify type NO\_PROPOSAL\_CHOSEN, as shown below. The GCKS MUST unregister the group member.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK {IDg [N,]} -->	
	<-- HDR, SK {}

### 3.1.3. IKEv2 Header Initialization

The Major Version is (2) and Minor Version number is (0) according to IKEv2 [RFC7296], and maintained in this document. The G-IKEv2 IKE\_SA\_INIT, GSA\_AUTH and GSA\_REGISTRATION use the IKE SPI according to IKEv2 [RFC7296], section 2.6.

### 3.1.4. GM Registration Operations

A G-IKEv2 Initiator (GM) requesting registration contacts the GCKS using the IKE\_SA\_INIT exchange and receives the response from the GCKS. This exchange is unchanged from the IKE\_SA\_INIT in IKEv2 protocol.

Upon completion of parsing and verifying the IKE\_SA\_INIT response, the GM sends the GSA\_AUTH message with the IKEv2 payloads from IKE\_AUTH (without the SAi2, TSi and TSr) along with the Group ID informing the GCKS of the group the initiator wishes to join. The initiator MAY specify how many Sender-ID values it would like to receive in the Notify payload status type SENDER\_ID\_REQUEST in case the Data Security SA supports a counter mode cipher (see Section 3.2).

An initiator may be limited in the types of Transforms that it is able or willing to use, and may find it useful to inform the GCKS of which Transforms that it is willing to accept. IT OPTIONALLY includes an SAg payload, which can include ESP and/or AH Proposals. Each Proposal contains a list of Transforms that it is willing to support for that protocol. A Proposal of type ESP can include ENCR, INTEG, and ESN Transforms. A Proposal of type AH can include INTEG, and ESN Transforms. The SPI length of each Proposal in an SAg MUST

be zero, and the SPI field is null. Generally, a single Proposal of each type will suffice, because the group member is not negotiating Transform sets, simply alerting the GCKS to restrictions it may have.

Upon receiving the GSA\_AUTH response, the initiator then parses the response from the GCKS authenticating the exchange using the IKEv2 method, then processing the GSA, and KD.

The GSA payload contains the security policy and cryptographic protocols used by the group. This policy describes the Rekey SA (KEK), if present, Data-security SAs (TEK), and other group policy (GAP). If the policy in the GSA payload is not acceptable to the GM, it SHOULD notify the GCKS with a NO\_PROPOSAL\_CHOSEN Notify (see Section 3.1.1 and Section 3.1.2). Finally the KD is parsed providing the keying material for the TEK and/or KEK. The GM interprets the KD key packets, where each key packet includes the keying material for SAs distributed in the GSA payload. Keying material is matched by comparing the SPIs in the key packets to SPIs previously included in the GSA payloads. Once TEK keys and policy are matched, the GM provides them to the data security subsystem, and it is ready to send or receive packets matching the TEK policy.

The GSA KEK policy MUST include KEK attribute KEK\_MESSAGE\_ID with a Message ID. The Message ID in the KEK\_MESSAGE\_ID attribute MUST be checked against any previously received Message ID for this group. If it is less than the previously received number, it should be considered stale and ignored. This could happen if two GSA\_AUTH exchanges happened in parallel, and the Message ID changed. This KEK\_MESSAGE\_ID is used by the GM to prevent GSA\_REKEY message replay attacks. The first GSA\_REKEY message that the GM receives from the GCKS needs to have a Message ID greater or equal to the Message ID received in the KEK\_MESSAGE\_ID attribute.

#### 3.1.5. GCKS Registration Operations

A G-IKEv2 GCKS passively listens for incoming requests from group members. The GCKS receives the IKE\_SA\_INIT request, select the IKE proposal, generates nonce and DH to include them in the IKE\_SA\_INIT response.

Upon receiving the GSA\_AUTH request, the GCKS authenticates the group member using the same procedures as in the IKEv2 IKE\_AUTH. The GCKS then authorizes the group member according to group policy before preparing to send GSA\_AUTH response. If the GCKS fails to authorize the GM, it will respond with the AUTHORIZATION\_FAILED notify message.

The GSA\_AUTH response will include group policy in GSA payload and keys in the KD payload. If the GCKS policy includes a group rekey

option, this policy is constructed in the GSA KEK and the key is constructed in the KD KEK. The GSA KEK MUST include attribute KEK\_MESSAGE\_ID specifying the starting Message ID the GCKS will be using when sending the GSA\_REKEY message to the group member. This Message ID is used to prevent replay attacks of the GSA\_REKEY message and will be increasing each time a GSA\_REKEY message is sent to the group. The GCKS data traffic policy is included in the GSA TEK and keys are included in KD TEK. GSA GAP MAY also be included to provide the ATD and/or DTD (Section 4.7.1) specifying activation and deactivation delays for SAs generated from the TEKs. If one or more Data Security SAs distributed in the GSA payload included a counter mode of operation, the GCKS includes at least one SID value in the KD payload, and possibly more depending on the request received in the Notify payload status type SENDER\_ID\_REQUEST requesting the number of SIDs from the group member.

If the GCKS receives a GSA\_REGISTRATION exchange with a request to register a GM to a group, the GCKS will need to authorize the GM with the new group (IDg) and respond with corresponding group policy and keys. If the GCKS fails to authorize the GM, it will respond with the AUTHORIZATION\_FAILED notification.

If a group member includes an SAg in its GSA\_AUTH or GSA\_REGISTRATION request, the GCKS MAY evaluate it according to an implementation specific policy.

- o The GCKS could evaluate the list of Transforms and compare it to its current policy for the group. If the group member did not include all of the ESP or AH Transforms in its current policy, then it could return a NO\_PROPOSAL\_CHOSEN Notify.
- o The GCKS could store the list of Transforms, with the goal of migrating the group policy to a different Transform when all of the group members indicate that they can support that Transform.
- o The GCKS could store the list of Transforms, and adjust the current group policy based on the capabilities of the devices as long as they fall within the acceptable security policy of the GCKS.

### 3.2. Counter-based modes of operation

Several new counter-based modes of operation have been specified for ESP (e.g., AES-CTR [RFC3686], AES-GCM [RFC4106], AES-CCM [RFC4309], AES-GMAC [RFC4543]) and AH (e.g., AES-GMAC [RFC4543]). These counter-based modes require that no two senders in the group ever send a packet with the same Initialization Vector (IV) using the same

cipher key and mode. This requirement is met in G-IKEv2 when the following requirements are met:

- o The GCKS distributes a unique key for each Data-Security SA.
- o The GCKS uses the method described in [RFC6054], which assigns each sender a portion of the IV space by provisioning each sender with one or more unique SID values.

When at least one Data-Security SA included in the group policy includes a counter-mode, the GCKS automatically allocates and distributes one SID to each group member acting in the role of sender on the Data-Security SA. The SID value is used exclusively by the group member to which it was allocated. The group member uses the same SID for each Data-Security SA specifying the use of a counter-based mode of operation. A GCKS MUST distribute unique keys for each Data-Security SA including a counter-based mode of operation in order to maintain a unique key and nonce usage.

During registration, the group member can choose to request one or more SID values. Requesting a value of 1 is not necessary since the GCKS will automatically allocate exactly one to the group member. A group member MUST request as many SIDs matching the number of encryption modules in which it will be installing the TEKs in the outbound direction. Alternatively, a group member MAY request more than one SID and use them serially. This could be useful when it is anticipated that the group member will exhaust their range of Data-Security SA nonces using a single SID too quickly (e.g., before the time-based policy in the TEK expires).

When group policy includes a counter-based mode of operation, a GCKS SHOULD use the following method to allocate SID values, which ensures that each SID will be allocated to just one group member.

1. A GCKS maintains an SID-counter, which records the SIDs that have been allocated. SIDs are allocated sequentially, with the first SID allocated to be zero.
2. Each time an SID is allocated, the current value of the counter is saved and allocated to the group member. The SID-counter is then incremented in preparation for the next allocation.
3. When the GCKS specifies a counter-based mode of operation in the Data Security SA a group member may request a count of SIDs during registration in a Notify payload information type SEND\_ID\_REQUEST. When the GCKS receives this request, it increments the SID-counter once for each requested SID, and distributes each SID value to the group member.

4. A GCKS allocates new SID values for each GSA\_REGISTRATION exchange originated by a sender, regardless of whether a group member had previously contacted the GCKS. In this way, the GCKS does not have a requirement of maintaining a record of which SID values it had previously allocated to each group member. More importantly, since the GCKS cannot reliably detect whether the group member had sent data on the current group Data-Security SAs it does not know what Data-Security counter-mode nonce values that a group member has used. By distributing new SID values, the key server ensures that each time a conforming group member installs a Data-Security SA it will use a unique set of counter-based mode nonces.

5. When the SID-counter maintained by the GCKS reaches its final SID value, no more SID values can be distributed. Before distributing any new SID values, the GCKS MUST delete the Data-Security SAs for the group, followed by creation of new Data-Security SAs, and resetting the SID-counter to its initial value.

6. The GCKS SHOULD send a GSA\_REKEY message deleting all Data-Security SAs and the Rekey SA for the group. This will result in the group members initiating a new GSA\_REGISTRATION exchange, in which they will receive both new SID values and new Data-Security SAs. The new SID values can safely be used because they are only used with the new Data-Security SAs. Note that deletion of the Rekey SA is necessary to ensure that group members receiving a GSA\_REKEY exchange before the re-register do not inadvertently use their old SIDs with the new Data-Security SAs. Using the method above, at no time can two group members use the same IV values with the same Data-Security SA key.

### 3.3. G-IKEv2 group maintenance channel

The GCKS indicates that it will be delivering group rekey messages when the KEK policy and keys are present in the G-IKEv2 GSA and KD payloads. Though the G-IKEv2 Rekey is optional, it plays a crucial role for large and dynamic groups. The GCKS is responsible for rekeying of the secure group per the group policy. The GCKS uses multicast to transport the rekey message. The G-IKEv2 protocol uses GSA\_REKEY exchange type in G-IKEv2 header identifying it as a rekey message. This rekey message is protected by the registration exchanges.

#### 3.3.1. G-IKEv2 GSA\_REKEY exchange

The GCKS initiates the G-IKEv2 Rekey securely using IP multicast. Since multicast rekey does not require a response and it sends to multiple GMs, G-IKEv2 rekeying MUST NOT support windowing. The GCKS rekey message replaces the rekey GSA KEK or KEK array, and/or creates

a new Data-Security GSA TEK. The SID Download attribute in the Key Download payload (defined in Section 4.8.4) MUST NOT be part of the Rekey Exchange as this is sender specific information and the Rekey Exchange is group specific. The GCKS initiates the GSA\_REKEY exchange as following:

Members (Responder)	GCKS (Initiator)
-----	-----
	<-- HDR, SK { GSA, KD, [D,] AUTH }

HDR is defined in Section 4.1. The Message ID in this message will start with the same value the GCKS sent to group member in the KEK attribute KEK\_MESSAGE\_ID during registration; this Message ID will be increasing each time a new GSA\_REKEY message is sent to the group members.

The GSA payload contains the current rekey and data security SAs. The GSA may contain a new data security SA and/or a new rekey SA, which, optionally contains an LKH rekey SA, Section 4.4.

The KD represents the keys for the policy included in the GSA. If the data security SA is being refreshed in this rekey message, the IPsec keys are updated in the KD, and/or if the rekey SA is being refreshed in this rekey message, the rekey Key or the LKH KEK array is updated in the KD payload.

The Delete payload MAY be included to instruct the GM to delete existing SAs.

The AUTH payload is included to authenticate GSA\_REKEY message using a method defined in the IKEv2 Authentication Method IANA registry [IKEV2-IANA]. The method SHOULD be a digital signature authentication scheme to ensure that the message was originated from an authorized GCKS. Shared Key Integrity Code SHOULD NOT be used as it doesn't provide source origin authentication (although a small group may not require source origin authentication). During group member registration, the GCKS sends the authentication key in the GSAK payload KEK\_AUTH\_KEY attribute, which the group member uses to authenticate the key server. Before the current Authentication Key expires, the GCKS will send a new KEK\_AUTH\_KEY to the group members in a GSA\_REKEY message. The AUTH key that is used in the rekey message may not be the same as the authentication key used in GSA\_AUTH. Typically rekey message is sent as multicast and received by all group members, the same AUTH key is distributed to all group members.

After adding the AUTH payload to the rekey message, the current KEK encryption key encrypts all payloads following the HDR.



### 3.3.2. Forward and Backward Access Control

Through G-IKEv2 rekey, the G-IKEv2 supports algorithms such as LKH that have the property of denying access to a new group key by a member removed from the group (forward access control) and to an old group key by a member added to the group (backward access control). An unrelated notion to PFS, "forward access control" and "backward access control" have been called "perfect forward security" and "perfect backward security" in the literature [RFC2627].

Group management algorithms providing forward and backward access control other than LKH have been proposed in the literature, including OFT [OFT] and Subset Difference [NNL]. These algorithms could be used with G-IKEv2, but are not specified as a part of this document.

Support for group management algorithms is supported via the KEY\_MANAGEMENT\_ALGORITHM attribute which is sent in the GSA KEK policy. G-IKEv2 specifies one method by which LKH can be used for forward and backward access control. Other methods of using LKH, as well as other group management algorithms such as OFT or Subset Difference may be added to G-IKEv2 as part of a later document.

### 3.3.3. Forward Access Control Requirements

When group membership is altered using a group management algorithm new GSA TEKs (and their associated keys) are usually also needed. New GSAs and keys ensure that members who were denied access can no longer participate in the group.

If forward access control is a desired property of the group, new GSA TEKs and the associated key packets in the KD payload MUST NOT be included in a G-IKEv2 rekey message which changes group membership. This is required because the GSA TEK policy and the associated key packets in the KD payload are not protected with the new KEK. A second G-IKEv2 rekey message can deliver the new GSA TEKs and their associated keys because it will be protected with the new KEK, and thus will not be visible to the members who were denied access.

If forward access control policy for the group includes keeping group policy changes from members that are denied access to the group, then two sequential G-IKEv2 rekey messages changing the group KEK MUST be sent by the GCKS. The first G-IKEv2 rekey message creates a new KEK for the group. Group members, which are denied access, will not be able to access the new KEK, but will see the group policy since the G-IKEv2 rekey message is protected under the current KEK. A subsequent G-IKEv2 rekey message containing the changed group policy and again changing the KEK allows complete forward access control. A

G-IKEv2 rekey message MUST NOT change the policy without creating a new KEK.

If other methods of using LKH or other group management algorithms are added to G-IKEv2, those methods MAY remove the above restrictions requiring multiple G-IKEv2 rekey messages, providing those methods specify how forward access control policy is maintained within a single G-IKEv2 rekey message.

#### 3.3.4. Deletion of SAs

There are occasions when the GCKS may want to signal to group members to delete policy at the end of a broadcast, or if group policy has changed. Deletion of keys MAY be accomplished by sending the G-IKEv2 Delete Payload [RFC7296], section 3.11 as part of the GSA\_REKEY Exchange as shown below.

Members (Responder)	GCKS (Initiator)
-----	-----
	<-- HDR, SK {
	[GSA ], [KD ], [D, ] AUTH }

The GSA MAY specify the remaining active time of the remaining policy by using the DTD attribute in the GSA GAP. If a GCKS has no further SAs to send to group members, the GSA and KD payloads MUST be omitted from the message. There may be circumstances where the GCKS may want to start over with a clean slate. If the administrator is no longer confident in the integrity of the group, the GCKS can signal deletion of all policy of a particular TEK protocol by sending a TEK with a SPI value equal to zero in the delete payload. For example, if the GCKS wishes to remove all the KEKs and all the TEKs in the group, the GCKS SHOULD send a Delete payload with a SPI of zero and a protocol\_id of a TEK protocol\_id value defined in Section 4.6, followed by another Delete payload with a SPI of zero and protocol\_id of zero, indicating that the KEK SA should be deleted.

#### 3.3.5. GSA\_REKEY GCKS Operations

The GCKS may initiate a rekey message if group membership and/or policy has changed, or if the keys are about to expire. The GCKS builds the rekey message with a Message ID value that is one greater than the value included in the previous rekey. If the message is using a new KEK attribute, the Message ID is reset to 1 in this message. The GSA and KD follow with the same characteristics as in the GSA Registration exchange. The AUTH payload is the final payload added to the message. It is created by hashing the string "G-IKEv2" and the message created so far, and then digitally signed. Finally,

the payloads following the HDR are encrypted and authenticated using the current KEK keys.

Because GSA\_REKEY messages are not acknowledged and could be discarded by the network, one or more GMs may not receive the message. To mitigate such lost messages, during a rekey event the GCKS SHOULD transmit several GSA\_REKEY messages with the new policy. A GCKS MUST NOT re-transmit the same GSA\_REKEY message, because time-to-live lifetimes in the message will be incorrect, resulting in GMs with unsynchronized TEK and KEK lifetimes.

### 3.3.6. GSA\_REKEY GM Operations

The group member receives the Rekey Message from the GCKS, decrypts the message using the current KEK, validates the signature using the public key retrieved in a previous G-IKEv2 exchange, verifies the Message ID, and processes the GSA and KD payloads. The group member then downloads the new data security SA and/or new Rekey GSA. The parsing of the payloads is identical to the registration exchange.

Replay protection is achieved when the group member rejects GSA\_REKEY message which has a Message ID smaller than the current Message ID that the GM is expecting. The GM expects the Message ID in the first GSA\_REKEY message it receives to be equal or greater than the message id it receives in the KEK\_MESSAGE\_ID attribute. The GM expects the message ID in the subsequent GSA\_REKEY message to be greater than the last valid GSA\_REKEY message it received.

If the GSA payload includes Data-Security SA including a counter-modes of operation and the receiving group member is a sender for that SA, the group member uses its current SID value with the Data-Security SAs to create counter-mode nonces. If it is a sender and does not hold a current SID value, it MUST NOT install the Data-Security SAs. It MAY initiate a GSA\_REGISTRATION exchange to the GCKS in order to obtain an SID value (along with current group policy).

If the GM receives a notification that a Data-Security SA is about to expire (such as a "soft lifetime" expiration described in Section 4.4.2.1 of [RFC4301]), it SHOULD initiate a registration to the GCKS. This registration serves as a request for current SAs, and will result in the download of replacement SAs, assuming the GCKS policy has created them.

#### 4. Header and Payload Formats

Refer to IKEv2 [RFC7296] for existing payloads.

##### 4.1. The G-IKEv2 Header

G-IKEv2 uses the same IKE header format as specified in RFC 7296 section 3.1.

Several new payload formats are required in the group security exchanges.

Next Payload Type -----	Value -----
Group Identification (IDg)	50
Group Security Association (GSA)	51
Key Download (KD)	52

New exchange types GSA\_AUTH, GSA\_REGISTRATION and GSA\_REKEY are added to the IKEv2 [RFC7296] protocol.

Exchange Type -----	Value -----
GSA_AUTH	39
GSA_REGISTRATION	40
GSA_REKEY	41

Major Version is 2 and Minor Version is 0 as in IKEv2 [RFC7296]. IKE SA Initiator's SPI, IKE SA Responder's SPI, Flags, Message ID, and Length are as specified in [RFC7296].

##### 4.2. Group Identification (IDg) Payload

The IDg Payload allows the group member to indicate which group it wants to join. The payload is constructed by using the IKEv2 Identification Payload (section 3.5 of [RFC7296]). ID type ID\_KEY\_ID MUST be supported. ID types ID\_IPV4\_ADDR, ID\_FQDN, ID\_RFC822\_ADDR, ID\_IPV6\_ADDR SHOULD be supported. ID types ID\_DER\_ASN1\_DN and ID\_DER\_ASN1\_GN are not expected to be used.

##### 4.3. Security Association - GM Supported Transforms (SAg)

The SAg payload declares which Transforms that a GM is willing to accept. The payload is constructed by using the IKEv2 Security Association payload (section 3.3 of [RFC7296]).

#### 4.4. Group Security Association Payload

The Group Security Association payload is used by the GCKS to assert security attributes for both Rekey and Data-security SAs.

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Next Payload |C|  RESERVED  |                               Payload Length |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The Security Association Payload fields are defined as follows:

- o Next Payload (1 octet) -- Identifies the next payload type for the G-IKEv2 registration or the G-IKEv2 rekey message.
- o Critical (1 bit) -- Set according to [RFC7296].
- o RESERVED (7 bits) -- Must be zero.
- o Payload Length (2 octets) -- Is the octet length of the current payload including the generic header and all TEK and KEK policies.

##### 4.4.1. GSA Policy

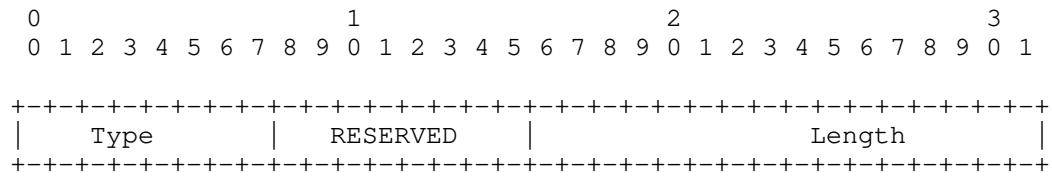
Following GSA generic payload header are GSA policies for group rekeying (KEK) and/or data traffic SAs (TEK) and/or Group Associated Policy (GAP). There may be zero or one GSA KEK policy, zero or more GAP policy, and zero or more GSA TEK policies, where either one GSA KEK or GSA TEK payload MUST be present.

This latitude allows various group policies to be accommodated. For example if the group policy does not require the use of a Rekey SA, the GCKS would not need to send an GSA KEK attribute to the group member since all SA updates would be performed using the Registration SA. Alternatively, group policy might use a Rekey SA but choose to download a KEK to the group member only as part of the Registration SA. Therefore, the GSA KEK policy would not be necessary as part of the GSA\_REKEY message.

Specifying multiple GSA TEKs allows multiple related data streams (e.g., video, audio, and text) to be associated with a session, but each protected with an individual security association policy.

A GAP payload allows for the distribution of group-wise policy, such as instructions as to when to activate and de-activate SAs.

Policies following the GSA payload has common header

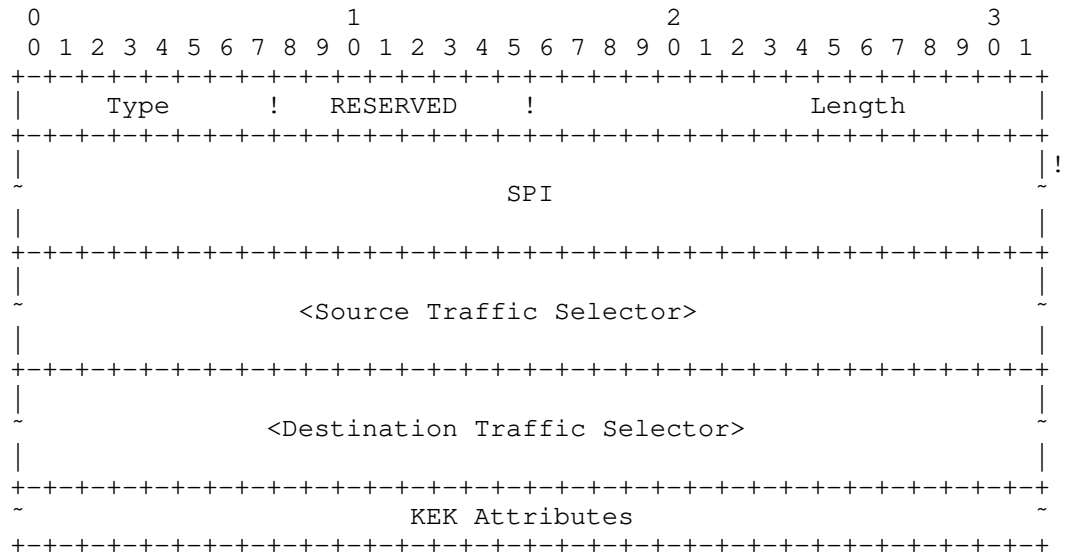


Type is defined as follows:

ID Class	Value
-----	-----
RESERVED	0
KEK	1
GAP	2
TEK	3
Expert Review	4-127
Private Use	128-255

#### 4.5. KEK Policy

The GSA KEK (GSAK) policy contains security attributes for the KEK method for a group and parameters specific to the G-IKEv2 registration operation. The source and destination traffic selectors describe the network identities used for the rekey messages.



The GSAK Payload fields are defined as follows:

- o Type (1 octet) -- Identifies the GSA payload type KEK present in the G-IKEv2 registration or the G-IKEv2 rekey message.
- o RESERVED (1 octet) -- Must be zero.
- o Length (2 octets) -- Length of this structure including KEK attributes.
- o SPI (16 octets) -- Security Parameter Index for the rekey message. The SPI must be the IKEv2 Header SPI pair where the first 8 octets become the "Initiator's SPI" field of the G-IKEv2 rekey message IKEv2 HDR, and the second 8 octets become the "Responder's SPI" in the same HDR. As described above, these SPIs are assigned by the GCKS.
- o Source & Destination Traffic Selectors - Substructures describing the source and destination of the network identities. These identities refer to the source and destination of the next KEK rekey SA. Defined format and values are specified by IKEv2 [RFC7296], section 3.13.1.
- o KEK Attributes -- Contains KEK policy attributes associated with the group. The following sections describe the possible attributes. Any or all attributes may be optional, depending on the group policy.

#### 4.5.1. KEK Attributes

The following attributes may be present in a GSA KEK policy. The attributes must follow the format defined in IKEv2 [RFC7296] section 3.3.5. In the table, attributes that are defined as TV are marked as Basic (B); attributes that are defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC5226]. The registration procedure is Expert Review.

ID Class	Value	Type
-----	-----	-----
Reserved	0	
KEK_MANAGEMENT_ALGORITHM	1	B
KEK_ENCR_ALGORITHM	2	B
KEK_KEY_LENGTH	3	B
KEK_KEY_LIFETIME	4	V
KEK_INTEGRITY_ALGORITHM	5	B
KEK_AUTH_METHOD	6	B
KEK_AUTH_HASH	7	B
KEK_MESSAGE_ID	8	V
Unassigned	9-16383	
Private Use	16384-32767	

The following attributes may only be included in a G-IKEv2 registration message: KEK\_MANAGEMENT\_ALGORITHM.

Minimum attributes that must be sent as part of an GSA KEK: KEK\_ENCR\_ALGORITHM, KEK\_KEY\_LENGTH (if the cipher definition includes a variable length key), KEK\_MESSAGE\_ID, KEK\_KEY\_LIFETIME, KEK\_INTEGRITY\_ALGORITHM, KEK\_AUTH\_METHOD and KEK\_AUTH\_HASH (except for DSA based algorithms).

#### 4.5.2. KEK\_MANAGEMENT\_ALGORITHM

The KEK\_MANAGEMENT\_ALGORITHM attribute specifies the group KEK management algorithm used to provide forward or backward access control (i.e., used to exclude group members). Defined values are specified in the following table. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC5226]. The registration procedure is Expert Review.

KEK Management Type	Value
-----	-----
Reserved	0
LKH	1
Unassigned	2-16383
Private Use	16384-32767

#### 4.5.3. KEK\_ENCR\_ALGORITHM

The KEK\_ENCR\_ALGORITHM attribute specifies the encryption algorithm using with the KEK. This value is a value from the IKEv2 Transform Type 1 - Encryption Algorithm Transform IDs registry[IKEV2-IANA]. If a KEK\_MANAGEMENT\_ALGORITHM is defined which defines multiple keys (e.g., LKH), and if the management algorithm does not specify the algorithm for those keys, then the algorithm defined by the



KEK\_ENCR\_ALGORITHM attribute MUST be used for all keys which are included as part of the management.

#### 4.5.4. KEK\_KEY\_LENGTH

The KEK\_KEY\_LENGTH attribute specifies the KEK Algorithm key length (in bits).

The Group Controller/Key Server (GCKS) adds the KEK\_KEY\_LENGTH attribute to the GSA payload when distributing KEK policy to group members. The group member verifies whether or not it has the capability of using a cipher key of that size. If the cipher definition includes a fixed key length, the group member can make its decision solely using KEK\_ENCR\_ALGORITHM attribute and does not need the KEK\_KEY\_LENGTH attribute. Sending the KEK\_KEY\_LENGTH attribute in the GSA payload is OPTIONAL if the KEK cipher has a fixed key length.

#### 4.5.5. KEK\_KEY\_LIFETIME

The KEK\_KEY\_LIFETIME attribute specifies the maximum time for which the KEK is valid. The GCKS may refresh the KEK at any time before the end of the valid period. The value is a four (4) octet number defining a valid time period in seconds.

#### 4.5.6. KEK\_INTEGRITY\_ALGORITHM

The KEK\_INTEGRITY attribute specifies the integrity algorithm used to protect the rekey message. This integrity algorithm is a value from the IKEv2 Transform Type 3 - Integrity Algorithm Transform IDs registry [IKEV2-IANA].

#### 4.5.7. KEK\_AUTH\_METHOD

The KEK\_AUTH\_METHOD attribute specifies the method of authentication used. This value is from the IKEv2 IKEv2 Authentication Method registry [IKEV2-IANA].

#### 4.5.8. KEK\_AUTH\_HASH

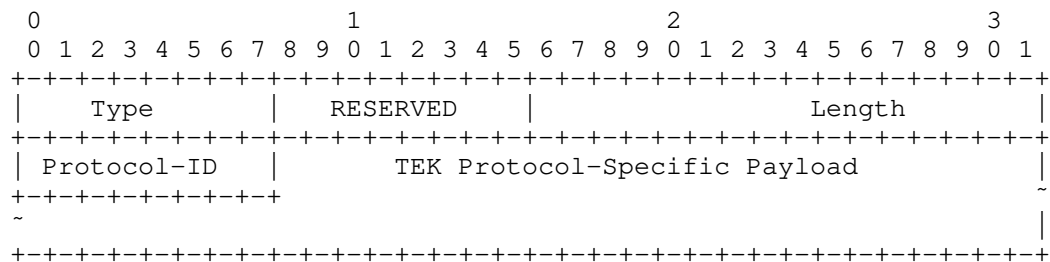
The KEK\_AUTH\_HASH attribute specifies the hash algorithm uses to generate AUTH key to authenticate GSA\_REKEY message. Hash algorithms are defined in IANA registry IKEv2 Hash Algorithms [IKEV2-IANA]. This attribute can be used by group member to determine in advance if it support the algorithm used in the rekey message.

## 4.5.9. KEK\_MESSAGE\_ID

The KEK\_MESSAGE\_ID attribute defines the initial Message ID to be used by the GCKS in the GSA\_REKEY messages. The Message ID is 4 octets unsigned integer in network byte order.

## 4.6. GSA TEK Policy

The GSA TEK (GSAT) policy contains security attributes for a single TEK associated with a group.



The GSAT Payload fields are defined as follows:

- o Type (1 octet) -- Identifies the GSA payload type TEK present in the G-IKEv2 registration or the G-IKEv2 rekey message.
- o RESERVED (1 octet) -- Must be zero.
- o Length (2 octets) -- Length of this structure, including the TEK Protocol-Specific Payload.
- o Protocol-ID (1 octet) -- Value specifying the Security Protocol. The following table defines values for the Security Protocol. Support for the GSA\_PROTO\_IPSEC\_AH GSA TEK is OPTIONAL. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC5226]. The registration procedure is Expert Review.

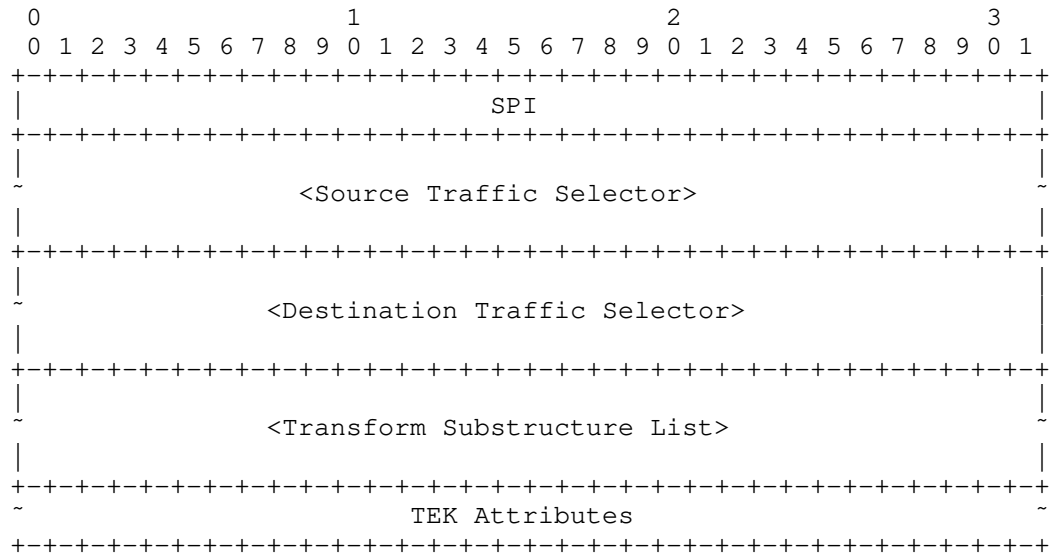
Protocol ID	Value
-----	-----
Reserved	0
GSA_PROTO_IPSEC_ESP	1
GSA_PROTO_IPSEC_AH	2
Unassigned	3-127
Private Use	128-255

- o TEK Protocol-Specific Payload (variable) -- Payload which describes the attributes specific for the Protocol-ID.

#### 4.6.1. TEK ESP and AH Protocol-Specific Policy

The TEK Protocol-Specific policy contains of two traffic selectors for source and destination of the protecting traffic, SPI, Transforms, and Attributes.

The TEK Protocol-Specific policy for ESP and AH is as follows:



The GSAT Policy fields are defined as follows:

- o SPI (4 octets) -- Security Parameter Index.
- o Source & Destination Traffic Selectors - The traffic selectors describe the source and the destination of the protecting traffic. The format and values are defined in IKEv2 [RFC7296], section 3.13.1.
- o Transform Substructure List -- A list of Transform Substructures specifies the transform information. The format and values are defined in IKEv2 [RFC7296], section 3.3.2. Valid Transform Types for ESP are ENCR, INTEG, and ESN. Valid Transform Types for AH are INTEG and ESN. As described in the IKEv2 registries [IKEV2-IANA]. The Last Substruc value in each Transform Substructure will be set to 3 except for the last one in the list, which is set to 0.

- o TEK Attributes -- Contains TEK policy attributes associated with the group, in the format defined in Section 3.3.5 of [RFC7296]. All attributes are optional, depending on the group policy.

Attribute Types are as follows. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC5226]. The registration procedure is Expert Review.

ID Class	Value	Type
-----	-----	----
Reserved	0	
TEK_KEY_LIFETIME	1	V
TEK_MODE	2	B
Unassigned	3-16383	
Private Use	16384-32767	

It is NOT RECOMMENDED that the GCKS distribute both ESP and AH Protocol-Specific Policy for the same set of Traffic Selectors.

#### 4.6.1.1. TEK\_KEY\_LIFETIME

The TEK\_KEY\_LIFETIME attribute specifies the maximum time for which the TEK is valid. When the TEK expires, the AH or ESP security association and all keys downloaded under the security association are discarded. The GCKS may refresh the KEK at any time before the end of the valid period.

The value is a four (4) octet number defining a valid time period in seconds. If unspecified, the default value shall be assumed to be 28800 seconds (8 hours).

#### 4.6.1.2. TEK\_MODE

In the absence of this attribute tunnel mode will be used. Value of 1 is used for transport mode.

### 4.7. GSA Group Associated Policy

Group specific policy that does not belong to rekey policy (GSA KEK) or traffic encryption policy (GSA TEK) can be distributed to all group member using GSA GAP (Group Associated Policy).

The GSA GAP payload is defined as follows:

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|  Type      !  RESERVED  !                               Length  |
+-----+-----+-----+-----+-----+-----+-----+-----+
~                               Group Associated Policy Attributes ~
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The GSA GAP payload fields are defined as follows:

- o Type (1 octet) -- Identifies the GSA payload type GAP present in the G-IKEv2 registration or the G-IKEv2 rekey message.
- o RESERVED (1 octet) -- Must be zero.
- o Length (2 octets) -- Length of this structure, including the GSA GAP header and Attributes.
- o Group Associated Policy Attributes (variable) -- Contains attributes following the format defined in Section 3.3.5 of [RFC7296].

Attribute Types are as follows. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC5226]. The registration procedure is Expert Review.

Attribute Type	Value	Type
Reserved	0	
ACTIVATION_TIME_DELAY	1	B
DEACTIVATION_TIME_DELAY	2	B
Unassigned	3-16383	
Private Use	16384-32767	

#### 4.7.1. ACTIVATION\_TIME\_DELAY/DEACTIVATION\_TIME\_DELAY

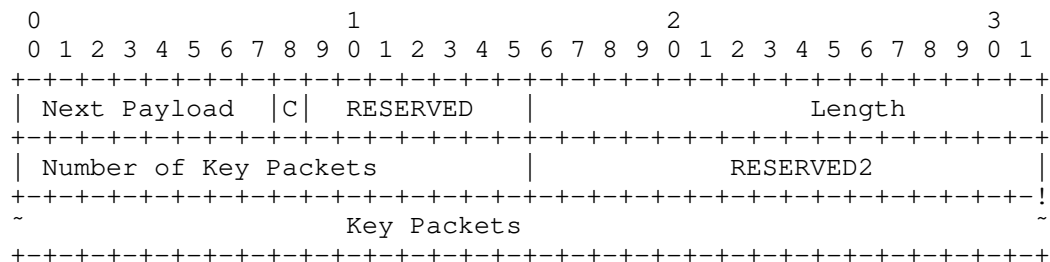
Section 4.2.1 of RFC 5374 specifies a key rollover method that requires two values be provided to group members. The ACTIVATION\_TIME\_DELAY attribute allows a GCKS to set the Activation Time Delay (ATD) for SAs generated from TEKs. The ATD defines how long after receiving new SAs that they are to be activated by the GM. The ATD value is in seconds.

The DEACTIVATION\_TIME\_DELAY allows the GCKS to set the Deactivation Time Delay (DTD) for previously distributed SAs. The DTD defines how long after receiving new SAs it should deactivate SAs that are destroyed by the rekey event. The value is in seconds.

The values of ATD and DTD are independent. However, the DTD value should be larger, which allows new SAs to be activated before older SAs are deactivated. Such a policy ensures that protected group traffic will always flow without interruption.

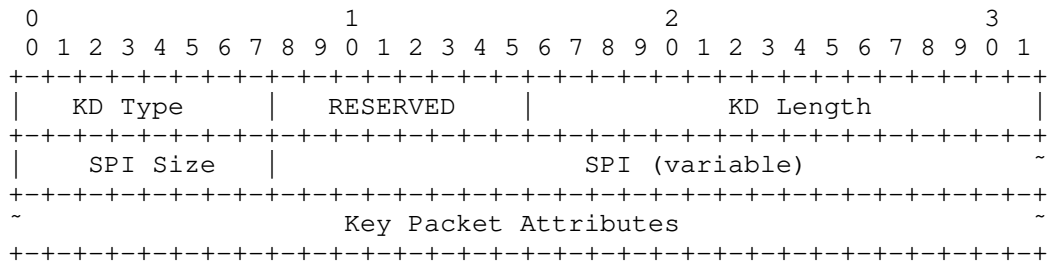
#### 4.8. Key Download Payload

The Key Download Payload contains group keys for the group specified in the GSA Payload. These key download payloads can have several security attributes applied to them based upon the security policy of the group as defined by the associated GSA Payload.



The Key Download Payload fields are defined as follows:

- o Next Payload (1 octet) -- Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be zero.
- o Critical (1 bit) -- Set according to [RFC7296].
- o RESERVED (7 bits) -- Unused, set to zero.
- o Payload Length (2 octets) -- Length in octets of the current payload, including the generic payload header.
- o Number of Key Packets (2 octets) -- Contains the total number of Key Packets passed in this data block.
- o Key Packets (variable) -- Contains Key Packets. Several types of key packets are defined. Each Key Packet has the following format.



- o Key Download (KD) Type (1 octet) -- Identifier for the Key Data field of this Key Packet. In the following table the terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC5226]. The registration procedure is Expert Review.

Key Download Type	Value
Reserved	0
TEK	1
KEK	2
LKH	3
SID	4
Unassigned	5-127
Private Use	128-255

- o RESERVED (1 octet) -- Unused, set to zero.
- o Key Download Length (2 octets) -- Length in octets of the Key Packet data, including the Key Packet header.
- o SPI Size (1 octet) -- Value specifying the length in octets of the SPI as defined by the Protocol-Id.
- o SPI (variable length) -- Security Parameter Index which matches a SPI previously sent in an GSAK or GSAT Payload.
- o Key Packet Attributes (variable length) -- Contains Key information. The format of this field is specific to the value of the KD Type field. The following sections describe the format of each KD Type.

#### 4.8.1. TEK Download Type

The following attributes may be present in a TEK Download Type. Exactly one attribute matching each type sent in the GSAT payload MUST be present. The attributes must follow the format defined in IKEv2 (Section 3.3.5 of [RFC7296]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked

as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC5226]. The registration procedure is Expert Review.

TEK Class	Value	Type
-----	-----	-----
Reserved	0	
TEK_ALGORITHM_KEY	1	V
TEK_INTEGRITY_KEY	2	V
Unassigned	3-16383	
Private Use	16384-32767	

It is possible that the GCKS will send no TEK key packets in a Registration KD payload (as well as no corresponding GSAT payloads in the GSA payload), after which the TEK payloads will be sent in a rekey message. At least one TEK MUST be included in each Rekey KD payload.

#### 4.8.1.1. TEK\_ALGORITHM\_KEY

The TEK\_ALGORITHM\_KEY attribute contains encryption keying material for the corresponding SPI. This keying material will be used with the encryption algorithm specified in the GSAT payload, and according to the IPsec transform describing that encryption algorithm. The keying material is treated equivalent to IKEv2 KEYMAT derived for that IPsec transform. If the encryption algorithm requires a nonce (e.g., AES-GCM), the nonce is chosen as shown in Section 3.2.

#### 4.8.1.2. TEK\_INTEGRITY\_KEY

The TEK\_INTEGRITY\_KEY class declares that the integrity key for the corresponding SPI is contained as the Key Packet Attribute. Readers should refer to [IKEV2-IANA] for the latest values.

#### 4.8.2. KEK Download Type

The following attributes may be present in a KEK Download Type. Exactly one attribute matching each type sent in the GSAK payload MUST be present. The attributes must follow the format defined in IKEv2 (Section 3.3.5 of [RFC7296]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC5226]. The registration procedure is Expert Review.



KEK Class -----	Value -----	Type -----
Reserved	0	
KEK_ENCR_KEY	1	V
KEK_INTEGRITY_KEY	2	V
KEK_AUTH_KEY	3	V
Unassigned	4-16383	
Private Use	16384-32767	

If the KEK Key Packet is included, there MUST be only one present in the KD payload.

#### 4.8.2.1. KEK\_ENCR\_KEY

The KEK\_ENCR\_KEY attribute declares that the encryption key for the corresponding SPI is contained in the Key Packet Attribute. The encryption algorithm that will use this key was specified in the GSAK payload.

If the mode of operation for the algorithm requires an Initialization Vector (IV), an explicit IV MUST be included in the KEK\_ALGORITHM\_KEY before the actual key.

#### 4.8.2.2. KEK\_INTEGRITY\_KEY

The KEK\_INTEGRITY\_KEY class declares the integrity key for this SPI is contained in the Key Packet Attribute. The integrity algorithm that will use this key was specified in the GSAK payload.

#### 4.8.2.3. KEK\_AUTH\_KEY

The KEK\_AUTH\_KEY class declares that the authentication key for this SPI is contained in the Key Packet Attribute. The signature algorithm that will use this key was specified in the GSAK payload. An RSA public key format is defined in RFC 3447, Section A.1.1. DSS public key format is defined in RFC 3279 Section 2.3.2. For ECDSA Public keys, use format described in RFC 5480 Section 2.2.

#### 4.8.3. LKH Download Type

The LKH key packet is comprised of attributes representing different leaves in the LKH key tree.

The following attributes are used to pass an LKH KEK array in the KD payload. The attributes must follow the format defined in IKEv2 (Section 3.3.5 of [RFC7296]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to

be applied as defined in [RFC5226]. The registration procedure is Expert Review.

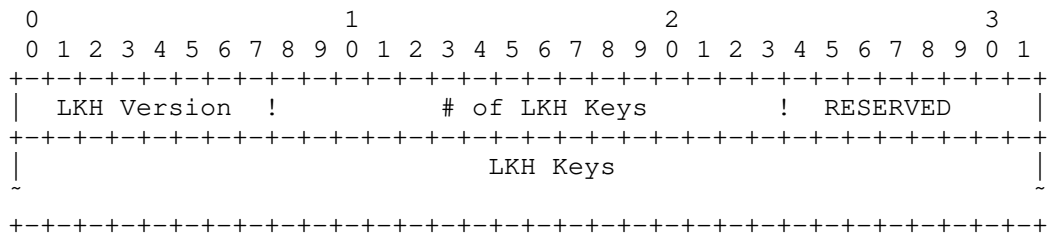
LKH Download Class	Value	Type
-----	-----	----
Reserved	0	
LKH_DOWNLOAD_ARRAY	1	V
LKH_UPDATE_ARRAY	2	V
Unassigned	3-16383	
Private Use	16384-32767	

If an LKH key packet is included in the KD payload, there MUST be only one present.

#### 4.8.3.1. LKH\_DOWNLOAD\_ARRAY

This attribute is used to download a set of keys to a group member. It MUST NOT be included in a IKEv2 rekey message KD payload if the IKEv2 rekey is sent to more than one group member. If an LKH\_DOWNLOAD\_ARRAY attribute is included in a KD payload, there MUST be only one present.

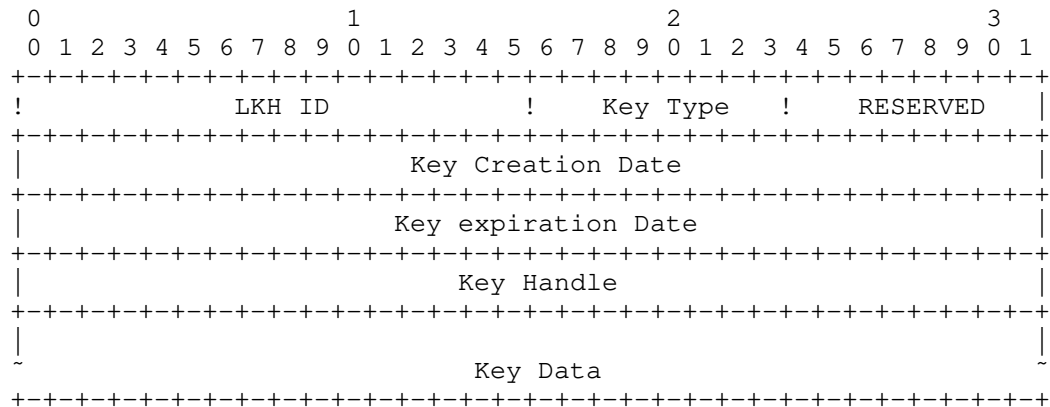
This attribute consists of a header block, followed by one or more LKH keys.



The KEK\_LKH attribute fields are defined as follows:

- o LKH version (1 octet) -- Contains the version of the LKH protocol which the data is formatted in. Must be one.
- o Number of LKH Keys (2 octets) -- This value is the number of distinct LKH keys in this sequence.
- o RESERVED (1 octet) -- Unused, set to zero.

Each LKH Key is defined as follows:



- o LKH ID (2 octets) -- This is the position of this key in the binary tree structure used by LKH.
- o Key Type (1 octet) -- This is the encryption algorithm for which this key data is to be used. This value is specified in Section 4.5.3.
- o RESERVED (1 octet) -- Unused, set to zero.
- o Key Creation Date (4 octets) -- This is the time value of when this key data was originally generated. A time value of zero indicates that there is no time before which this key is not valid.
- o Key Expiration Date (4 octets) -- This is the time value of when this key is no longer valid for use. A time value of zero indicates that this key does not have an expiration time.
- o Key Handle (4 octets) -- This is the randomly generated value to uniquely identify a key within an LKH ID.
- o Key Data (variable length) -- This is the actual encryption key data, which is dependent on the Key Type algorithm for its format. If the mode of operation for the algorithm requires an Initialization Vector (IV), an explicit IV MUST be included in the Key Data field before the actual key.

The Key Creation Date and Key expiration Dates MAY be zero. This is necessary in the case where time synchronization within the group is not possible.

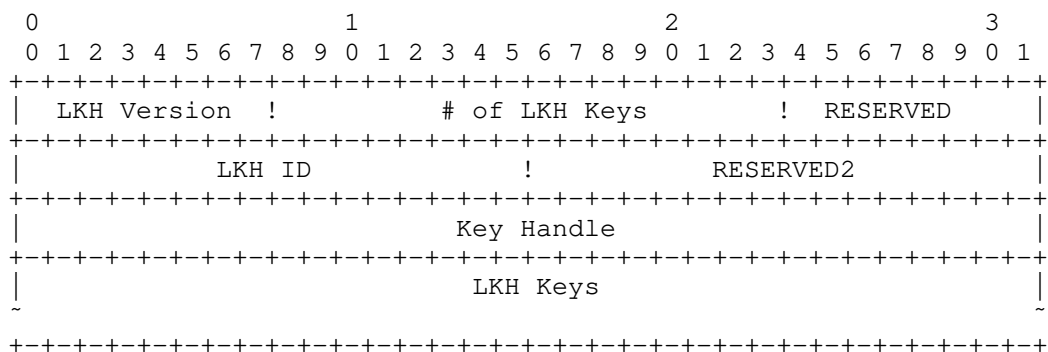
The first LKH Key structure in an LKH\_DOWNLOAD\_ARRAY attribute contains the Leaf identifier and key for the group member. The rest

of the LKH Key structures contain keys along the path of the key tree in the order starting from the leaf, culminating in the group KEK.

#### 4.8.3.2. LKH\_UPDATE\_ARRAY

This attribute is used to update the keys for a group. It is most likely to be included in a G-IKEv2 rekey message KD payload to rekey the entire group. This attribute consists of a header block, followed by one or more LKH keys, as defined in Section 4.8.3.1.

There may be any number of UPDATE\_ARRAY attributes included in a KD payload.



- o LKH version (1 octet) -- Contains the version of the LKH protocol which the data is formatted in. Must be one.
- o Number of LKH Keys (2 octets) -- This value is the number of distinct LKH keys in this sequence.
- o RESERVED (1 octet) -- Unused, set to zero.
- o LKH ID (2 octets) -- This is the node identifier associated with the key used to encrypt the first LKH Key.
- o RESERVED2 (2 octets) -- Unused, set to zero.
- o Key Handle (4 octets) -- This is the value to uniquely identify the key within the LKH ID which was used to encrypt the first LKH key.

The LKH Keys are as defined in Section 4.8.3.1. The LKH Key structures contain keys along the path of the key tree in the order from the LKH ID found in the LKH\_UPDATE\_ARRAY header, culminating in the group KEK. The Key Data field of each LKH Key is encrypted with the LKH key preceding it in the LKH\_UPDATE\_ARRAY attribute. The

first LKH Key is encrypted under the key defined by the LKH ID and Key Handle found in the LKH\_UPDATE\_ARRAY header.

#### 4.8.4. SID Download Type

This attribute is used to download one or use more Sender-ID (SID) values for the exclusive use of a group member. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC5226]. The registration procedure is Expert Review.

SID Download Class	Value	Type
-----	-----	----
Reserved	0	
NUMBER_OF_SID_BITS	1	B
SID_VALUE	2	V
Unassigned	3-16383	
Private Use	16384-32767	

Because a SID value is intended for a single group member, the SID Download type MUST NOT be distributed in a GSA\_REKEY message distributed to multiple group members.

##### 4.8.4.1. NUMBER\_OF\_SID\_BITS

The NUMBER\_OF\_SID\_BITS class declares how many bits of the cipher nonce in which to represent an SID value. This value applied to each SID value is distributed in the SID Download.

##### 4.8.4.2. SID\_VALUE

The SID\_VALUE class declares a single SID value for the exclusive use of the a group member. Multiple SID\_VALUE attributes MAY be included in a SID Download.

##### 4.8.4.3. GM Semantics

The SID\_VALUE attribute value distributed to the group member MUST be used by that group member as the SID field portion of the IV for all Data-Security SAs including a counter-based mode of operation distributed by the GCKS as a part of this group. When the Sender-Specific IV (SSIV) field for any Data-Security SA is exhausted, the group member MUST NOT act as a sender on that SA using its active SID. The group member SHOULD re-register, at which time the GCKS will issue a new SID to the group member, along with either the same Data-Security SAs or replacement ones. The new SID replaces the existing SID used by this group member, and also resets the SSIV value to its starting value. A group member MAY re-register prior to the actual exhaustion of the SSIV field to avoid dropping data

packets due to the exhaustion of available SSIV values combined with a particular SID value.

A group member MUST NOT process an SID Download Type KD payload present in a GSA-REKEY message.

#### 4.8.4.4. GCKS Semantics

If any KD payload includes keying material that is associated with a counter-mode of operation, a SID Download Type KD payload containing at least one SID\_VALUE attribute MUST be included. The GCKS MUST NOT send the SID Download Type KD payload as part of a GSA\_REKEY message, because distributing the same sender-specific policy to more than one group member will reduce the security of the group.

#### 4.9. Delete Payload

There are occasions when the GCKS may want to signal to receivers to delete policy at the end of a broadcast, or if policy has changed. Deletion of keys MAY be accomplished by sending an IKEv2 Delete Payload, section 3.11 of [RFC7296] as part of the GSA\_AUTH or GSA\_REKEY Exchange. One or more Delete payloads MAY be placed following the HDR payload in the GSA\_AUTH or GSA\_REKEY Exchange.

The Protocol ID MUST be 41 for GSA\_REKEY Exchange, 2 for AH or 3 for ESP. Note that only one protocol id value can be defined in a Delete payload. If a TEK and a KEK SA for GSA\_REKEY Exchange must be deleted, they must be sent in different Delete payloads. Similarly, if a TEK specifying ESP and a TEK specifying AH need to be deleted, they must be sent in different Delete payloads.

There may be circumstances where the GCKS may want to reset the policy and keying material for the group. The GCKS can signal deletion of all policy of a particular TEK protocol by sending a TEK with an SPI value equal to zero in the delete payload. In the event that the administrator is no longer confident in the integrity of the group they may wish to remove all the KEKs and all the TEKs in the group. This is done by having the GCKS send a delete payload with an SPI of zero and a Protocol-ID of AH or ESP Protocol-ID value to delete all TEKs, followed by another delete payload with an SPI value of zero and Protocol-ID of KEK SA to delete the KEK SA.

#### 4.10. Notify Payload

G-IKEv2 uses the same Notify payload as specified in [RFC7296], section 3.10.

There are additional Notify Message types introduced by G-IKEv2 to communicate error conditions and status.

NOTIFY messages - error types	Value
-----	-----
INVALID_GROUP_ID -	45
Indicates the group id sent during registration process is invalid.	
AUTHORIZATION_FAILED -	46
Sent in the response to GSA_AUTH message when authorization failed.	
NOTIFY messages - status types	Value
-----	-----
SENDER_REQUEST_ID -	16429
Sent in GSA_AUTH or GSA_REGISTRATION to request SIDs from GCKS. The data includes a count of how many SID values it desires.	

#### 4.11. Authentication Payload

G-IKEv2 uses the same Authentication payload as specified in [RFC7296], section 3.8, to sign the rekey message.

### 5. Security Considerations

#### 5.1. GSA registration and secure channel

G-IKEv2 registration exchange uses IKEv2 IKE\_SA\_INIT protocols, inheriting all the security considerations documented in [RFC7296] section 5 Security Considerations, including authentication, confidentiality, protection against man-in-the-middle, protection against replay/reflection attacks, and denial of service protection. The GSA\_AUTH and GSA\_REGISTRATION exchanges also take advantage of those protections. In addition, G-IKEv2 brings in the capability to authorize a particular group member regardless of whether they have the IKEv2 credentials.

#### 5.2. GSA maintenance channel

The GSA maintenance channel is cryptographically and integrity protected using the cryptographic algorithm and key negotiated in the GSA member registration exchanged.

##### 5.2.1. Authentication/Authorization

Authentication is implicit, the public key of the identity is distributed during the registration, and the receiver of the rekey message uses that public key and identity to verify the message is come from the authorized GCKS.

### 5.2.2. Confidentiality

Confidentiality is provided by distributing a confidentiality key as part of the GSA member registration exchange.

### 5.2.3. Man-in-the-Middle Attack Protection

GSA maintenance channel is integrity protected by using digital signature.

### 5.2.4. Replay/Reflection Attack Protection

The GSA\_REKEY message includes a monotonically increasing sequence number to protect against replay and reflection attacks. A group member will recognize a replayed message by comparing the Message ID number to that of the last received rekey message, any rekey message contains Message ID number less than or equal to the last received value MUST be discarded. Implementations should keep a record of recently received GSA rekey messages for this comparison.

## 6. IANA Considerations

### 6.1. New registries

A new set of registries should be created for G-IKEv2, on a new page titled Group Key Management using IKEv2 (G-IKEv2) Parameters. The following registries should be placed on that page. The terms Reserved, Expert Review and Private Use are to be applied as defined in [RFC5226].

GSA Policy Type Registry, see Section 4.4.1

KEK Attributes Registry, see Section 4.5.1

KEK Management Algorithm Registry, see Section 4.5.2

GSA TEK Payload Protocol ID Type Registry, see Section 4.6

TEK Attributes Registry, see Section 4.6

Key Download Type Registry, see Section 4.8

TEK Download Type Attributes Registry, see Section 4.8.1

KEK Download Type Attributes Registry, see Section 4.8.2

LKH Download Type Attributes Registry, see Section 4.8.3



SID Download Type Attributes Registry, see Section 4.8.4

## 6.2. New payload and exchange types to existing IKEv2 registry

The following new payloads and exchange types specified in this memo have already been allocated by IANA and require no further action, other than replacing the draft name with an RFC number.

The present document describes new IKEv2 Next Payload types, see Section 4.1

The present document describes new IKEv2 Exchanges types, see Section 4.1

The present document describes new IKEv2 notification types, see Section 4.10

## 7. Acknowledgements

The authors thank Lakshminath Dondeti and Jing Xiang for first exploring the use of IKEv2 for group key management and providing the basis behind the protocol.

## 8. Contributors

The following individuals made substantial contributions to early versions of this memo.

Sheela Rowles  
Cisco Systems  
170 W. Tasman Drive  
San Jose, California 95134-1706  
USA

Phone: +1-408-527-7677  
Email: sheela@cisco.com

Aldous Yeung  
Cisco Systems  
170 W. Tasman Drive  
San Jose, California 95134-1706  
USA

Phone: +1-408-853-2032  
Email: cyyeung@cisco.com

Paulina Tran  
Cisco Systems  
170 W. Tasman Drive  
San Jose, California 95134-1706  
USA

Phone: +1-408-526-8902  
Email: ptran@cisco.com

## 9. References

### 9.1. Normative References

- [RFC6054] McGrew, D. and B. Weis, "Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic", RFC 6054, DOI 10.17487/RFC6054, November 2010, <<http://www.rfc-editor.org/info/rfc6054>>.
- [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>>.

### 9.2. Informative References

- [IKE-HASH] Kivinen, T., "Fixing IKE Phase 1 & 2 Authentication HASHs", November 2001, <<http://tools.ietf.org/html/draft-ietf-ipsec-ike-hash-revised-03>>.
- [IKEV2-IANA] IANA, "Internet Key Exchange Version 2 (IKEv2) Parameters", February 2016, <<http://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-7>>.
- [NNL] Naor, D., Noal, M., and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers", Advances in Cryptology, Crypto '01, Springer-Verlag LNCS 2139, 2001, pp. 41-62, 2001, <<http://www.wisdom.weizmann.ac.il/~naor/>>.
- [OFT] McGrew, D. and A. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees", Manuscript, submitted to IEEE Transactions on Software Engineering, 1998, <<http://download.nai.com/products/media/nai/misc/oft052098.ps>>.

- [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>>.
- [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>>.
- [RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, DOI 10.17487/RFC2407, November 1998, <<http://www.rfc-editor.org/info/rfc2407>>.
- [RFC2408] Maughan, D., Schertler, M., Schneider, M., and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, DOI 10.17487/RFC2408, November 1998, <<http://www.rfc-editor.org/info/rfc2408>>.
- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998, <<http://www.rfc-editor.org/info/rfc2409>>.
- [RFC2627] Wallner, D., Harder, E., and R. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, DOI 10.17487/RFC2627, June 1999, <<http://www.rfc-editor.org/info/rfc2627>>.
- [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>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [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>>.

#### Appendix A. Differences between G-IKEv2 and RFC 6407

KE Payload - The KE payload is no longer needed with the availability of newer algorithms such as AES and GCM which provide adequate protection therefore not needing the PFS capability the KE payload offers.

SIG Payload - The AUTH payload is used for the same purpose instead.

DOI/Situation - The DOI and Situation fields in the SA payload are no longer needed in the G-IKEv2 protocol as port 848 will distinguish the IKEv2 messages from the G-IKEv2 messages.

SEQ Payload - The SEQ payload is no longer needed since IKEv2 header has message id which is used to prevent message replay attacks.

#### Authors' Addresses

Brian Weis  
Cisco Systems  
170 W. Tasman Drive  
San Jose, California 95134-1706  
USA

Phone: +1-408-526-4796  
Email: [bew@cisco.com](mailto:bew@cisco.com)

Yoav Nir  
Check Point Software Technologies Ltd.  
5 Hasolelim St.  
Tel Aviv 67897  
Israel

Email: [ynir.ietf@gmail.com](mailto: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](mailto:svan@elvis.ru)

Network Working Group  
Internet-Draft  
Obsoletes: 6407 (if approved)  
Intended status: Standards Track  
Expires: January 9, 2020

B. Weis  
Independent  
V. Smyslov  
ELVIS-PLUS  
July 8, 2019

Group Key Management using IKEv2  
draft-yeung-g-ikev2-16

Abstract

This document presents a set of IKEv2 exchanges that comprise a group key management protocol. The protocol is in conformance with the Multicast Security (MSEC) key management architecture, which contains two components: member registration and group rekeying. Both components require a Group Controller/Key Server to download IPsec group security associations to authorized members of a group. The group members then exchange IP multicast or other group traffic as IPsec packets. This document obsoletes RFC 6407.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction and Overview . . . . .	3
1.1. Requirements Language . . . . .	5
1.2. G-IKEv2 Integration into IKEv2 Protocol . . . . .	5
1.2.1. G-IKEv2 Transport and Port . . . . .	5
1.2.2. IKEv2 Header Initialization . . . . .	6
1.3. G-IKEv2 Protocol . . . . .	6
1.3.1. G-IKEv2 Payloads . . . . .	6
1.4. G-IKEv2 Member Registration and Secure Channel Establishment . . . . .	7
1.4.1. GSA_AUTH exchange . . . . .	7
1.4.2. GSA_REGISTRATION Exchange . . . . .	9
1.4.3. GM Registration Operations . . . . .	10
1.4.4. GCKS Registration Operations . . . . .	11
1.4.5. Group Maintenance Channel . . . . .	12
1.4.6. Counter-based modes of operation . . . . .	19
1.5. Interaction with IKEv2 Protocol Extensions . . . . .	21
1.5.1. Postquantum Preshared Keys for IKEv2 . . . . .	21
2. Header and Payload Formats . . . . .	23
2.1. The G-IKEv2 Header . . . . .	23
2.2. Group Identification (IDg) Payload . . . . .	24
2.3. Security Association - GM Supported Transforms (SAg) . . . . .	24
2.4. Group Security Association Payload . . . . .	24
2.4.1. GSA Policy . . . . .	24
2.4.2. KEK Policy . . . . .	26
2.4.3. GSA TEK Policy . . . . .	29
2.4.4. GSA Group Associated Policy . . . . .	33
2.5. Key Download Payload . . . . .	34
2.5.1. TEK Download Type . . . . .	36
2.5.2. KEK Download Type . . . . .	37
2.5.3. LKH Download Type . . . . .	38
2.5.4. SID Download Type . . . . .	40
2.6. Delete Payload . . . . .	42
2.7. Notify Payload . . . . .	42
2.8. Authentication Payload . . . . .	43
3. Security Considerations . . . . .	43
3.1. GSA Registration and Secure Channel . . . . .	43
3.2. GSA Maintenance Channel . . . . .	44
3.2.1. Authentication/Authorization . . . . .	44
3.2.2. Confidentiality . . . . .	44
3.2.3. Man-in-the-Middle Attack Protection . . . . .	44
3.2.4. Replay/Reflection Attack Protection . . . . .	44

4.	IANA Considerations . . . . .	44
4.1.	New Registries . . . . .	44
4.2.	New Payload and Exchange Types Added to the Existing IKEv2 Registry . . . . .	45
4.3.	Changes to Previous Allocations . . . . .	45
5.	Acknowledgements . . . . .	45
6.	Contributors . . . . .	46
7.	References . . . . .	46
7.1.	Normative References . . . . .	47
7.2.	Informative References . . . . .	48
Appendix A.	Use of LKH in G-IKEv2 . . . . .	50
A.1.	Group Creation . . . . .	50
A.2.	Group Member Exclusion . . . . .	51
Authors' Addresses	. . . . .	52

## 1. Introduction and Overview

A group key management protocol provides IPsec keys and policy to a set of IPsec devices which are authorized to communicate using a Group Security Association (GSA) defined in [RFC3740]. The data communications within the group (e.g., IP multicast packets) are protected by a key pushed to the group members (GMs) by the Group Controller/Key Server (GCKS). This document presents a set of IKEv2 [RFC7296] exchanges that comprise a group key management protocol.

A GM begins a "registration" exchange when it first joins the group. With G-IKEv2, the GCKS authenticates and authorizes GMs, then pushes policy and keys used by the group to the GM. G-IKEv2 includes two "registration" exchanges. The first is the GSA\_AUTH exchange (Section 1.4.1), which follows an IKE\_SA\_INIT exchange. The second is the GSA\_REGISTRATION exchange (Section 1.4.2), which a GM can use within an established IKE SA. Group rekeys are accomplished using either the GSA\_REKEY exchange (a single message distributed to all GMs, usually as a multicast message), or as a GSA\_INBAND\_REKEY exchange delivered individually to group members using existing IKE SAs).

Large and small groups may use different sets of these protocols. When a large group of devices are communicating, the GCKS is likely to use the GSA\_REKEY message for efficiency. This is shown in Figure 1. (Note: For clarity, IKE\_SA\_INIT is omitted from the figure.)



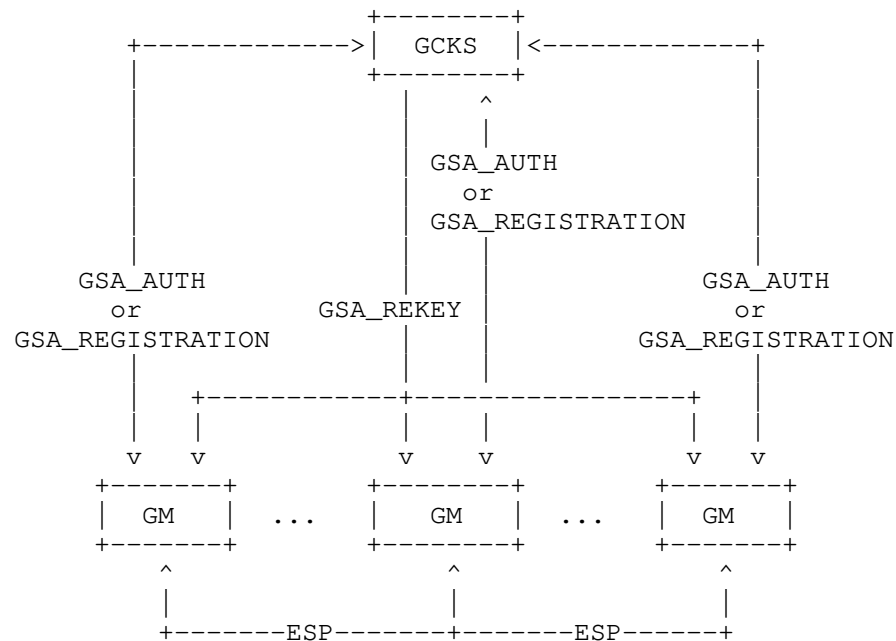


Figure 1: G-IKEv2 used in large groups

Alternatively, a small group may simply use the GSA\_AUTH as a registration protocol, where the GCKS issues rekeys using the GSA\_INBAND\_REKEY within the same IKEv2 SA. The GCKS is also likely to be a GM in a small group (as shown in Figure 2.)

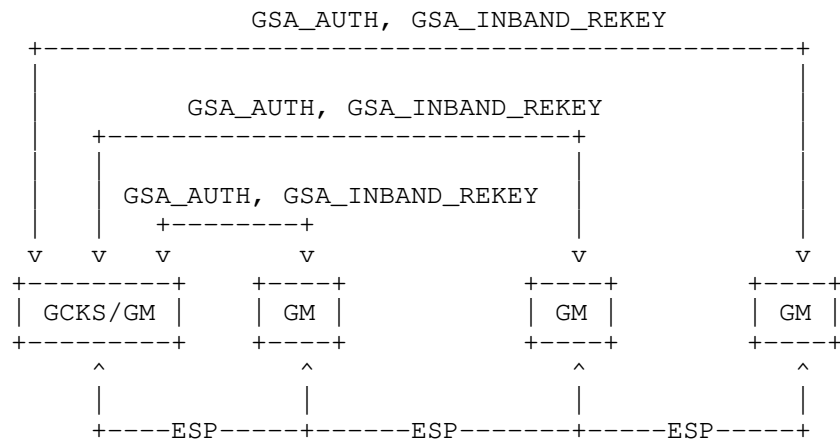


Figure 2: G-IKEv2 used in small groups

IKEv2 message semantics are preserved in that all communications consists of message request-response pairs. The exception to this rule is the GSA\_REKEY exchange, which is a single message delivering group updates to the GMs.

G-IKEv2 conforms with the Multicast Group Security Architecture [RFC3740], and the Multicast Security (MSEC) Group Key Management Architecture [RFC4046]. G-IKEv2 replaces GDOI [RFC6407], which defines a similar group key management protocol using IKEv1 [RFC2409] (since deprecated by IKEv2). When G-IKEv2 is used, group key management use cases can benefit from the simplicity, increased robustness and cryptographic improvements of IKEv2 (see Appendix A of [RFC7296]).

### 1.1. Requirements Language

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

### 1.2. G-IKEv2 Integration into IKEv2 Protocol

G-IKEv2 uses the security mechanisms of IKEv2 (peer authentication, confidentiality, message integrity) to ensure that only authenticated devices have access to the group policy and keys. The G-IKEv2 exchange further provides group authorization, and secure policy and key download from the GCKS to GMs. Some IKEv2 extensions require special handling if used with G-IKEv2. See Section 1.5 for more details.

It is assumed that readers are familiar with the IKEv2 protocol, so this document skips many details that are described in [RFC7296].

#### 1.2.1. G-IKEv2 Transport and Port

G-IKEv2 SHOULD use UDP port 848, the same as GDOI [RFC6407], because they serve a similar function. They can use the same ports, just as IKEv1 and IKEv2 can share port 500. The version number in the IKE header distinguishes the G-IKEv2 protocol from GDOI protocol [RFC6407]. G-IKEv2 MAY also use the IKEv2 ports (500, 4500), which would provide a better integration with IKEv2. G-IKEv2 MAY also use TCP transport for registration (unicast) IKE SA, as defined in [RFC8229].

### 1.2.2. IKEv2 Header Initialization

The Major Version is (2) and Minor Version is (0) according to IKEv2 [RFC7296], and maintained in this document. The G-IKEv2 IKE\_SA\_INIT, GSA\_AUTH, GSA\_REGISTRATION and GSA\_INBAND\_REKEY use the IKE SPI according to IKEv2 [RFC7296], section 2.6.

## 1.3. G-IKEv2 Protocol

### 1.3.1. G-IKEv2 Payloads

In the following descriptions, the payloads contained in the G-IKEv2 messages are indicated by names as listed below.

Notation	Payload
AUTH	Authentication
CERT	Certificate
CERTREQ	Certificate Request
GSA	Group Security Association
HDR	IKEv2 Header
IDg	Identification - Group
IDi	Identification - Initiator
IDr	Identification - Responder
KD	Key Download
KE	Key Exchange
Ni, Nr	Nonce
SA	Security Association
SAg	Security Association - GM Supported Transforms

Payloads defined as part of other IKEv2 extensions MAY also be included in these messages. Payloads that may optionally appear will be shown in brackets, such as [ CERTREQ ], to indicate that a certificate request payload can optionally be included.

G-IKEv2 defines several new payloads not used in IKEv2:

- o IDg (Group ID) - The GM requests the GCKS for membership into the group by sending its IDg payload.
- o GSA (Group Security Association) - The GCKS sends the group policy to the GM using this payload.
- o KD (Key Download) - The GCKS sends the control and data keys to the GM using the KD payload.

- o SAg (Security Association - GM Supported Transforms) - the GM sends supported transforms, so that GCKS may select a policy appropriate for all members of the group.

The details of the contents of each payload are described in Section 2.

#### 1.4. G-IKEv2 Member Registration and Secure Channel Establishment

The registration protocol consists of a minimum of two messages exchanges, `IKE_SA_INIT` and `GSA_AUTH`; member registration may have a few more messages exchanged if the EAP method, cookie challenge (for DoS protection) or negotiation of Diffie-Hellman group is included. Each exchange consists of request/response pairs. The first exchange `IKE_SA_INIT` is defined in IKEv2 [RFC7296]. This exchange negotiates cryptographic algorithms, exchanges nonces and does a Diffie-Hellman exchange between the group member (GM) and the Group Controller/Key Server (GCKS).

The second exchange `GSA_AUTH` authenticates the previous messages, exchanges identities and certificates. These messages are encrypted and integrity protected with keys established through the `IKE_SA_INIT` exchange, so the identities are hidden from eavesdroppers and all fields in all the messages are authenticated. The GCKS SHOULD authorize group members to be allowed into the group as part of the `GSA_AUTH` exchange. Once the GCKS accepts a group member to join a group it will download the data security keys (TEKs) and/or group key encrypting key (KEK) or KEK array as part of the `GSA_AUTH` response message.

##### 1.4.1. `GSA_AUTH` exchange

After the group member and GCKS use the `IKE_SA_INIT` exchange to negotiate cryptographic algorithms, exchange nonces, and perform a Diffie-Hellman exchange as defined in IKEv2 [RFC7296], the `GSA_AUTH` exchange MUST complete before any other exchanges can be done. The security properties of the `GSA_AUTH` exchange are the same as the properties of the `IKE_AUTH` exchange. It is used to authenticate the `IKE_SA_INIT` messages, exchange identities and certificates. G-IKEv2 also uses this exchange for group member registration and authorization. Even though the `IKE_AUTH` does contain the SA2, TSi, and TSr payload the `GSA_AUTH` does not. They are not needed because policy is not negotiated between the group member and the GCKS, but instead downloaded from the GCKS to the group member.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK { IDi, [CERT,] [CERTREQ, ] [IDr, ] AUTH, IDg, [SAg, ] [N ] }	-->

Figure 3: GSA\_AUTH Request

After the IKE\_SA\_INIT exchange completes, the group member initiates a GSA\_AUTH request to join a group indicated by the IDg payload. The GM MAY include an SAg payload declaring which Transforms that it is willing to accept. A GM that intends to emit data packets SHOULD include a Notify payload status type of SENDER, which enables the GCKS to provide any additional policy necessary by group senders.

Initiator (Member)	Responder (GCKS)
-----	-----
	<-- HDR, SK { IDr, [CERT, ] AUTH, [ GSA, KD, ] [D, ] }

Figure 4: GSA\_AUTH Normal Response

The GCKS responds with IDr, optional CERT, and AUTH material as if it were an IKE\_AUTH. It also informs the group member of the cryptographic policies of the group in the GSA payload and the key material in the KD payload. The GCKS can also include a Delete (D) payload instructing the group member to delete existing SAs it might have as the result of a previous group member registration. Note, that since the GCKS generally doesn't know which SAs the GM has, the SPI field in the Delete payload(s) SHOULD be set to zero in this case. (See more discussion on the Delete payload in Section 2.6.)

In addition to the IKEv2 error handling, the GCKS can reject the registration request when the IDg is invalid or authorization fails, etc. In these cases, see Section 2.7, the GSA\_AUTH response will not include the GSA and KD, but will include a Notify payload indicating errors. If the group member included an SAg payload, and the GCKS chooses to evaluate it, and it detects that that group member cannot support the security policy defined for the group, then the GCKS SHOULD return a NO\_PROPOSAL\_CHOSEN. Other types of notifications can be AUTHORIZATION\_FAILED or REGISTRATION\_FAILED.

Initiator (Member)	Responder (GCKS)
-----	-----
	<-- HDR, SK { IDr, [CERT, ] AUTH, N }

Figure 5: GSA\_AUTH Error Response

If the group member finds the policy sent by the GCKS is unacceptable, the member SHOULD initiate GSA\_REGISTRATION exchange sending IDg and the Notify NO\_PROPOSAL\_CHOSEN (see Section 1.4.2)).

#### 1.4.2. GSA\_REGISTRATION Exchange

When a secure channel is already established between a GM and the GCKS, the GM registration for a group can reuse the established secure channel. In this scenario the GM will use the GSA\_REGISTRATION exchange. Payloads in the exchange are generated and processed as defined in Section 1.4.1.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK {IDg, [SAg, ][N ] } -->	
	<-- HDR, SK { GSA, KD, [D ] }

Figure 6: GSA\_REGISTRATION Normal Exchange

As with GSA\_AUTH exchange, the GCKS can reject the registration request when the IDg is invalid or authorization fails, or GM cannot support the security policy defined for the group (which can be concluded by GCKS by evaluation of SAg payload). In this case the GCKS returns an appropriate error notification as described in Section 1.4.1.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK {IDg, [SAg, ][N ] } -->	
	<-- HDR, SK { N }

Figure 7: GSA\_REGISTRATION Error Exchange

This exchange can also be used if the group member finds the policy sent by the GCKS is unacceptable or for some reason wants to unregister itself from the group. The group member SHOULD notify the GCKS by sending IDg and the Notify type NO\_PROPOSAL\_CHOSEN or REGISTRATION\_FAILED, as shown below. The GCKS MUST unregister the group member.

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK {IDg, N } -->	
	<-- HDR, SK {}

Figure 8: GM Reporting Errors in GSA\_REGISTRATION Exchange

#### 1.4.3. GM Registration Operations

A G-IKEv2 Initiator (GM) requesting registration contacts the GCKS using the IKE\_SA\_INIT exchange and receives the response from the GCKS. This exchange is unchanged from the IKE\_SA\_INIT in IKEv2 protocol.

Upon completion of parsing and verifying the IKE\_SA\_INIT response, the GM sends the GSA\_AUTH message with the IKEv2 payloads from IKE\_AUTH (without the SAi2, TSi and TSr payloads) along with the Group ID informing the GCKS of the group the initiator wishes to join. An initiator intending to emit data traffic SHOULD send a SENDER Notify payload status. The SENDER not only signifies that it is a sender, but provides the initiator the ability to request Sender-ID values, in case the Data Security SA supports a counter mode cipher. Section 1.4.6) includes guidance on requesting Sender-ID values.

An initiator may be limited in the types of Transforms that it is able or willing to use, and may find it useful to inform the GCKS which Transforms that it is willing to accept. It can OPTIONALLY include an SAg payload, which can include ESP and/or AH Proposals. Each Proposal contains a list of Transforms that it is willing to support for that protocol. A Proposal of type ESP can include ENCR, INTEG, and ESN Transforms. A Proposal of type AH can include INTEG, and ESN Transforms. The SPI length of each Proposal in an SAg is set to zero, and thus the SPI field is null. The GCKS MUST ignore SPI field in the SAg payload. Generally, a single Proposal of each type will suffice, because the group member is not negotiating Transform sets, simply alerting the GCKS to restrictions it may have, however if the GM has restrictions on combination of algorithms, this can be expressed by sending several proposals.

Upon receiving the GSA\_AUTH response, the initiator parses the response from the GCKS authenticating the exchange using the IKEv2 method, then processes the GSA and KD.

The GSA payload contains the security policy and cryptographic protocols used by the group. This policy describes the Rekey SA (KEK), if present, Data-security SAs (TEK), and other group policy (GAP). If the policy in the GSA payload is not acceptable to the GM, it SHOULD notify the GCKS by initiating a GSA\_REGISTRATION exchange with a NO\_PROPOSAL\_CHOSEN Notify payload (see Section 1.4.2). Note, that this should normally not happen if the GM includes SAg payload in the GSA\_AUTH request and the GCKS takes it into account. Finally the KD is parsed providing the keying material for the TEK and/or KEK. The GM interprets the KD key packets, where each key packet includes the keying material for SAs distributed in the GSA payload.

Keying material is matched by comparing the SPIs in the key packets to SPIs previously included in the GSA payloads. Once TEK keys and policy are matched, the GM provides them to the data security subsystem, and it is ready to send or receive packets matching the TEK policy.

The GSA KEK policy MUST include KEK attribute KEK\_MESSAGE\_ID with a Message ID. The Message ID in the KEK\_MESSAGE\_ID attribute MUST be checked against any previously received Message ID for this group. If it is less than the previously received number, it should be considered stale and ignored. This could happen if two GSA\_AUTH exchanges happened in parallel, and the Message ID changed. This KEK\_MESSAGE\_ID is used by the GM to prevent GSA\_REKEY message replay attacks. The first GSA\_REKEY message that the GM receives from the GCKS must have a Message ID greater or equal to the Message ID received in the KEK\_MESSAGE\_ID attribute.

Once a GM has received GSA\_REKEY policy during a registration the IKE SA may be closed. However, the GM SHOULD NOT close IKE SA, it is the GCKS who makes the decision whether to close or keep it, because depending on the policy the IKE SA may be used for inband rekeying for small groups.

#### 1.4.4. GCKS Registration Operations

A G-IKEv2 GCKS passively listens for incoming requests from group members. When the GCKS receives an IKE\_SA\_INIT request, it selects an IKE proposal and generates a nonce and DH to include them in the IKE\_SA\_INIT response.

Upon receiving the GSA\_AUTH request, the GCKS authenticates the group member using the same procedures as in the IKEv2 IKE\_AUTH. The GCKS then authorizes the group member according to group policy before preparing to send the GSA\_AUTH response. If the GCKS fails to authorize the GM, it will respond with an AUTHORIZATION\_FAILED notify message.

The GSA\_AUTH response will include the group policy in the GSA payload and keys in the KD payload. If the GCKS policy includes a group rekey option, this policy is constructed in the GSA KEK and the key is constructed in the KD KEK. The GSA KEK MUST include the KEK\_MESSAGE\_ID attribute, specifying the starting Message ID the GCKS will use when sending the GSA\_REKEY message to the group member. This Message ID is used to prevent GSA\_REKEY message replay attacks and will be increased each time a GSA\_REKEY message is sent to the group. The GCKS data traffic policy is included in the GSA TEK and keys are included in the KD TEK. The GSA GAP MAY also be included to provide the ATD and/or DTD (Section 2.4.4.1) specifying activation



and deactivation delays for SAs generated from the TEKs. If the group member has indicated that it is a sender of data traffic and one or more Data Security SAs distributed in the GSA payload included a counter mode of operation, the GCKS responds with one or more SIDs (see Section 1.4.6).

If the GCKS receives a GSA\_REGISTRATION exchange with a request to register a GM to a group, the GCKS will need to authorize the GM with the new group (IDg) and respond with the corresponding group policy and keys. If the GCKS fails to authorize the GM, it will respond with the AUTHORIZATION\_FAILED notification.

If a group member includes an SAg in its GSA\_AUTH or GSA\_REGISTRATION request, the GCKS MAY evaluate it according to an implementation specific policy.

- o The GCKS could evaluate the list of Transforms and compare it to its current policy for the group. If the group member did not include all of the ESP or AH Transforms in its current policy, then it could return a NO\_PROPOSAL\_CHOSEN Notification.
- o The GCKS could store the list of Transforms, with the goal of migrating the group policy to a different Transform when all of the group members indicate that they can support that Transform.
- o The GCKS could store the list of Transforms and adjust the current group policy based on the capabilities of the devices as long as they fall within the acceptable security policy of the GCKS.

Depending on its policy, the GCKS may have no need for the IKE SA (e.g., it does not plan to initiate an GSA\_INBAND\_REKEY exchange). If the GM does not initiate another registration exchange or Notify (e.g., NO\_PROPOSAL\_CHOSEN), and also does not close the IKE SA and the GCKS is not intended to use the SA, then after a short period of time the GCKS SHOULD close the IKEv2 SA. The delay before closing provides for receipt of a GM's error notification in the event of packet loss.

#### 1.4.5. Group Maintenance Channel

The GCKS is responsible for rekeying the secure group per the group policy. Rekeying is an operation whereby the GCKS provides replacement TEKs and KEK, deleting TEKs, and/or excluding group members. The GCKS may initiate a rekey message if group membership and/or policy has changed, or if the keys are about to expire. Two forms of group maintenance channels are provided in G-IKEv2 to push new policy to group members.

**GSA\_REKEY** The GSA\_REKEY exchange is an exchange initiated by the GCKS, where the rekey policy is usually delivered to group members using IP multicast as a transport. This is valuable for large and dynamic groups, and where policy may change frequently and an scalable rekeying method is required. When the GSA\_REKEY exchange is used, the IKEv2 SA protecting the member registration exchanges is terminated, and group members await policy changes from the GCKS via the GSA\_REKEY exchange.

**GSA\_INBAND\_REKEY** The GSA\_INBAND\_REKEY exchange is a rekey method using the IKEv2 SA that was setup to protecting the member registration exchange. This exchange allows the GCKS to rekey without using an independent GSA\_REKEY exchange. The GSA\_INBAND\_REKEY exchange is useful when G-IKEv2 is used with a small group of cooperating devices.

#### 1.4.5.1. GSA\_REKEY Exchange

The GCKS initiates the G-IKEv2 Rekey securely, usually using IP multicast. Since this rekey does not require a response and it sends to multiple GMs, G-IKEv2 rekeying **MUST NOT** support IKE SA windowing. The GCKS rekey message replaces the rekey GSA KEK or KEK array, and/or creates a new Data-Security GSA TEK. The SID Download attribute in the Key Download payload (defined in Section 2.5.4) **MUST NOT** be part of the Rekey Exchange as this is sender specific information and the Rekey Exchange is group specific. The GCKS initiates the GSA\_REKEY exchange as following:

Members (Responder)	GCKS (Initiator)
-----	-----
	<-- HDR, SK { GSA, KD, [D,] [AUTH] }

Figure 9: GSA\_REKEY Exchange

HDR is defined in Section 2.1. The Message ID in this message will start with the same value the GCKS sent to the group members in the KEK attribute KEK\_MESSAGE\_ID during registration; this Message ID will be increased each time a new GSA\_REKEY message is sent to the group members.

The GSA payload contains the current rekey and data security SAs. The GSA may contain a new rekey SA and/or a new data security SA, which, optionally contains an LKH rekey SA, Section 2.4.

The KD payload contains the keys for the policy included in the GSA. If the data security SA is being refreshed in this rekey message, the IPsec keys are updated in the KD, and/or if the rekey SA is being

refreshed in this rekey message, the rekey Key or the LKH KEK array is updated in the KD payload.

A Delete payload MAY be included to instruct the GM to delete existing SAs.

The AUTH payload MUST be included to authenticate the GSA\_REKEY message if the authentication method is based on public key signatures and MUST NOT be included if it is based on shared secret. In a latter case, the fact that a GM can decrypt the GSA\_REKEY message and verify its ICV proves that the sender of this message knows the current KEK, thus authenticating that the sender is a member of the group. Shared secret authentication doesn't provide source origin authentication. For this reason using it as authentication method for multicast Rekey is NOT RECOMMENDED unless source origin authentication is not required (for example, in a small group of highly trusted GMs). If AUTH payload is included then the Auth Method field MUST be one specifying using digital signatures.

During group member registration, the GCKS sends the authentication key in the GSA KEK payload, KEK\_AUTH\_KEY attribute, which the group member uses to authenticate the key server. Before the current Authentication Key expires, the GCKS will send a new KEK\_AUTH\_KEY to the group members in a GSA\_REKEY message. The AUTH key that is used in the rekey message may be not the same as the authentication key used in GSA\_AUTH.

#### 1.4.5.1.1. GSA\_REKEY GCKS Operations

The GCKS builds the rekey message with a Message ID value that is one greater than the value included in the previous rekey. If the message is using a new KEK attribute, the Message ID is reset to 1 in this message. The GSA, KD, and D payloads follow with the same characteristics as in the GSA Registration exchange.

If present the AUTH payload is created as follows. First the message is prepared, all payloads are formed and included in the message, but the content of the Encrypted payload is not yet encrypted. However, the Encrypted payload must be fully formed, including correct values in IV, Padding and Pad Length and fields. The AUTH payload is included in the message with the correct values in the Payload Header (including Next Payload, Payload Length and Auth Method fields). The Authentication Data field is zeroed for the purposes of signature calculation, but if Digital Signature authentication method is in use, then the ASN.1 Length and the AlgorithmIdentifier fields must be properly filled in, see [RFC7427]. The signature is computed using the signature algorithm from the KEK\_AUTH\_METHOD attribute (along with the KEK\_AUTH\_HASH if KEK\_AUTH\_METHOD is not Digital Signature)

and the private key corresponding to the public key from the KEK\_AUTH\_KEY attribute. It is computed over the block of data starting from the first octet of IKE Header (but non including non-ESP marker if it is present) to the last octet of the (not yet encrypted) Encrypted Payload (i.e. up to and including Pad Length field). Then the signature is placed into the Signature Value of the AUTH payload, the content of the Encrypted payload is encrypted and the ICV is computed using current KEK keys.

Because GSA\_REKEY messages are not acknowledged and could be discarded by the network, one or more GMs may not receive the message. To mitigate such lost messages, during a rekey event the GCKS may transmit several GSA\_REKEY messages with the new policy. The retransmitted messages MUST be bitwise identical and SHOULD be sent within a short time interval (a few seconds) to ensure that time-to-live would not be substantially skewed for the GMs that would receive different copies of the messages.

GCKS may also include one or several KEK\_NEXT\_SPI/TEK\_NEXT\_SPI attributes specifying SPIs for the prospected rekeys, so that listening GMs are able to detect lost rekey messages and recover from this situation. See Sections Section 2.4.2.1.6 and Section 2.4.3.1.4 for more detail.

#### 1.4.5.1.2. GSA\_REKEY GM Operations

When a group member receives the Rekey Message from the GCKS it decrypts the message using the current KEK, validates the signature using the public key retrieved in a previous G-IKEv2 exchange if AUTH payload is present, verifies the Message ID, and processes the GSA and KD payloads. The group member then downloads the new data security SA and/or new Rekey SA. The parsing of the payloads is identical to the parsing done in the registration exchange.

Replay protection is achieved by a group member rejecting a GSA\_REKEY message which has a Message ID smaller than the current Message ID that the GM is expecting. The GM expects the Message ID in the first GSA\_REKEY message it receives to be equal or greater than the message id it receives in the KEK\_MESSAGE\_ID attribute. The GM expects the message ID in subsequent GSA\_REKEY messages to be greater than the last valid GSA\_REKEY message ID it received.

If the GSA payload includes a Data-Security SA including a counter-modes of operation and the receiving group member is a sender for that SA, the group member uses its current SID value with the Data-Security SAs to create counter-mode nonces. If it is a sender and does not hold a current SID value, it MUST NOT install the Data-Security SAs. It MAY initiate a GSA\_REGISTRATION exchange to the

GCKS in order to obtain an SID value (along with current group policy).

Once a new Rekey SA is installed as a result of GSA\_REKEY message, the current Rekey SA (over which the message was received) MUST be silently deleted after waiting DEACTIVATION\_TIME\_DELAY interval regardless of its expiration time. If the GSA TEK payload includes TEK\_REKEY\_SPI attribute then after installing a new Data-Security SA the old one, identified by the SPI in this attribute, MUST be silently deleted after waiting DEACTIVATION\_TIME\_DELAY interval regardless of its expiration time.

If a Data-Security SA is not rekeyed yet and is about to expire (a "soft lifetime" expiration is described in Section 4.4.2.1 of [RFC4301]), the GM SHOULD initiate a registration to the GCKS. This registration serves as a request for current SAs, and will result in the download of replacement SAs, assuming the GCKS policy has created them. A GM SHOULD also initiate a registration request if a Rekey SA is about to expire and not yet replaced with a new one.

#### 1.4.5.1.3. Forward and Backward Access Control

Through the G-IKEv2 rekey, G-IKEv2 supports algorithms such as LKH that have the property of denying access to a new group key by a member removed from the group (forward access control) and to an old group key by a member added to the group (backward access control). An unrelated notion to PFS, "forward access control" and "backward access control" have been called "perfect forward security" and "perfect backward security" in the literature [RFC2627].

Group management algorithms providing forward and backward access control other than LKH have been proposed in the literature, including OFT [OFT] and Subset Difference [NNL]. These algorithms could be used with G-IKEv2, but are not specified as a part of this document.

Support for group management algorithms are supported via the KEY\_MANAGEMENT\_ALGORITHM attribute which is sent in the GSA KEK policy. G-IKEv2 specifies one method by which LKH can be used for forward and backward access control. Other methods of using LKH, as well as other group management algorithms such as OFT or Subset Difference may be added to G-IKEv2 as part of a later document.

##### 1.4.5.1.3.1. Forward Access Control Requirements

When group membership is altered using a group management algorithm new GSA TEKs (and their associated keys) are usually also needed.

New GSAs and keys ensure that members who were denied access can no longer participate in the group.

If forward access control is a desired property of the group, new GSA TEKs and the associated key packets in the KD payload MUST NOT be included in a G-IKEv2 rekey message which changes group membership. This is required because the GSA TEK policy and the associated key packets in the KD payload are not protected with the new KEK. A second G-IKEv2 rekey message can deliver the new GSA TEKs and their associated key packets because it will be protected with the new KEK, and thus will not be visible to the members who were denied access.

If forward access control policy for the group includes keeping group policy changes from members that are denied access to the group, then two sequential G-IKEv2 rekey messages changing the group KEK MUST be sent by the GCKS. The first G-IKEv2 rekey message creates a new KEK for the group. Group members, which are denied access, will not be able to access the new KEK, but will see the group policy since the G-IKEv2 rekey message is protected under the current KEK. A subsequent G-IKEv2 rekey message containing the changed group policy and again changing the KEK allows complete forward access control. A G-IKEv2 rekey message MUST NOT change the policy without creating a new KEK.

If other methods of using LKH or other group management algorithms are added to G-IKEv2, those methods MAY remove the above restrictions requiring multiple G-IKEv2 rekey messages, providing those methods specify how the forward access control policy is maintained within a single G-IKEv2 rekey message.

#### 1.4.5.1.4. Fragmentation

IKE fragmentation [RFC7383] can be used to perform fragmentation of large GSA\_REKEY messages, however when the GSA\_REKEY message is emitted as an IP multicast packet there is a lack of response from the GMs. This has the following implications.

- o Policy regarding the use of IKE fragmentation is implicit. If a GCKS detects that all GMs have negotiated support of IKE fragmentation in IKE\_SA\_INIT, then it MAY use IKE fragmentation on large GSA\_REKEY exchange messages.
- o The GCKS must always use IKE fragmentation based on a known fragmentation threshold (unspecified in this memo), as there is no way to check if fragmentation is needed by first sending unfragmented messages and waiting for response.

- o PMTU probing cannot be performed due to lack of GSA\_REKEY response message.

#### 1.4.5.2. GSA\_INBAND\_REKEY Exchange

When the IKEv2 SA protecting the member registration exchange is maintained while group member participates in the group, the GCKS can use the GSA\_INBAND\_REKEY exchange to individually provide policy updates to the group member.

Member (Responder)		GCKS (Initiator)
-----		-----
	<--	HDR, SK { GSA, KD, [D,] }
HDR, SK {}	-->	

Figure 10: GSA\_INBAND\_REKEY Exchange

Because this is an IKEv2 exchange, the HDR is treated as defined in [RFC7296].

##### 1.4.5.2.1. GSA\_INBAND\_REKEY GCKS Operations

The GSA, KD, and D payloads are built in the same manner as in a registration exchange.

##### 1.4.5.2.2. GSA\_INBAND\_REKEY GM Operations

The GM processes the GSA, KD, and D payloads in the same manner as if they were received in a registration exchange.

##### 1.4.5.3. Deletion of SAs

There are occasions when the GCKS may want to signal to group members to delete policy at the end of a broadcast, or if group policy has changed. Deletion of keys MAY be accomplished by sending the G-IKEv2 Delete Payload [RFC7296], section 3.11 as part of the GSA\_REKEY Exchange as shown below.

Members (Responder)		GCKS (Initiator)
-----		-----
	<--	HDR, SK { [GSA ], [KD ], [D, ] [AUTH ] }

Figure 11: SA Deletion in GSA\_REKEY

The GSA MAY specify the remaining active time of the remaining policy by using the DTD attribute in the GSA GAP. If a GCKS has no further SAs to send to group members, the GSA and KD payloads MUST be omitted from the message. There may be circumstances where the GCKS may want

to start over with a clean slate. If the administrator is no longer confident in the integrity of the group, the GCKS can signal deletion of all the policies of a particular TEK protocol by sending a TEK with a SPI value equal to zero in the delete payload. For example, if the GCKS wishes to remove all the KEKs and all the TEKs in the group, the GCKS SHOULD send a Delete payload with a SPI of zero and a protocol\_id of a TEK protocol\_id value defined in Section 2.4.3, followed by another Delete payload with a SPI of zero and protocol\_id of zero, indicating that the KEK SA should be deleted.

#### 1.4.6. Counter-based modes of operation

Several new counter-based modes of operation have been specified for ESP (e.g., AES-CTR [RFC3686], AES-GCM [RFC4106], AES-CCM [RFC4309], AES-GMAC [RFC4543]) and AH (e.g., AES-GMAC [RFC4543]). These counter-based modes require that no two senders in the group ever send a packet with the same Initialization Vector (IV) using the same cipher key and mode. This requirement is met in G-IKEv2 when the following requirements are met:

- o The GCKS distributes a unique key for each Data-Security SA.
- o The GCKS uses the method described in [RFC6054], which assigns each sender a portion of the IV space by provisioning each sender with one or more unique SID values.

##### 1.4.6.1. Allocation of SIDs

When at least one Data-Security SA included in the group policy includes a counter-based mode of operation, the GCKS automatically allocates and distributes one SID to each group member acting in the role of sender on the Data-Security SA. The SID value is used exclusively by the group member to which it was allocated. The group member uses the same SID for each Data-Security SA specifying the use of a counter-based mode of operation. A GCKS MUST distribute unique keys for each Data-Security SA including a counter-based mode of operation in order to maintain unique key and nonce usage.

During registration, the group member can choose to request one or more SID values. Requesting a value of 1 is not necessary since the GCKS will automatically allocate exactly one to the group member. A group member MUST request as many SIDs matching the number of encryption modules in which it will be installing the TEKs in the outbound direction. Alternatively, a group member MAY request more than one SID and use them serially. This could be useful when it is anticipated that the group member will exhaust their range of Data-Security SA nonces using a single SID too quickly (e.g., before the time-based policy in the TEK expires).



When the group policy includes a counter-based mode of operation, a GCKS SHOULD use the following method to allocate SID values, which ensures that each SID will be allocated to just one group member.

1. A GCKS maintains an SID-counter, which records the SIDs that have been allocated. SIDs are allocated sequentially, with zero as the first allocated SID.

2. Each time an SID is allocated, the current value of the counter is saved and allocated to the group member. The SID-counter is then incremented in preparation for the next allocation.

3. When the GCKS specifies a counter-based mode of operation in the Data Security SA a group member may request a count of SIDs during registration in a Notify payload information of type SENDER. When the GCKS receives this request, it increments the SID-counter once for each requested SID, and distributes each SID value to the group member. The GCKS SHOULD have a policy-defined upper bound for the number of SIDs that it will return irrespective of the number requested by the GM.

4. A GCKS allocates new SID values for each GSA\_REGISTRATION exchange originated by a sender, regardless of whether a group member had previously contacted the GCKS. In this way, the GCKS is not required to maintaining a record of which SID values it had previously allocated to each group member. More importantly, since the GCKS cannot reliably detect whether the group member had sent data on the current group Data-Security SAs it does not know what Data-Security counter-mode nonce values that a group member has used. By distributing new SID values, the key server ensures that each time a conforming group member installs a Data-Security SA it will use a unique set of counter-based mode nonces.

5. When the SID-counter maintained by the GCKS reaches its final SID value, no more SID values can be distributed. Before distributing any new SID values, the GCKS MUST delete the Data-Security SAs for the group, followed by creation of new Data-Security SAs, and resetting the SID-counter to its initial value.

6. The GCKS SHOULD send a GSA\_REKEY message deleting all Data-Security SAs and the Rekey SA for the group. This will result in the group members initiating a new GSA\_REGISTRATION exchange, in which they will receive both new SID values and new Data-Security SAs. The new SID values can safely be used because they are only used with the new Data-Security SAs. Note that deletion of the Rekey SA is necessary to ensure that group members receiving a GSA\_REKEY exchange before the re-register do not inadvertently use their old SIDs with the new Data-Security SAs. Using the method above, at no time can

two group members use the same IV values with the same Data-Security SA key.

#### 1.4.6.2. GM Usage of SIDs

A GM applies the SID to Data Security SA as follows.

1. The most significant bits `NUMBER_OF_SID_BITS` of the IV are taken to be the SID field of the IV.
2. The SID is placed in the least significant bits of the SID field, where any unused most significant bits are set to zero. If the SID value doesn't fit into the `NUMBER_OF_SID_BITS` bits, then the GM MUST treat this as a fatal error and re-register to the group.

#### 1.5. Interaction with IKEv2 Protocol Extensions

IKEv2 defines a number of extensions that can be used to extend protocol functionality. G-IKEv2 is compatible with most of such extensions. In particular, EAP authentication defined in [RFC7296] can be used to establish registration IKE SA, as well as Secure Password authentication ([RFC6467]). G-IKEv2 is compatible with and can use IKEv2 Session Resumption [RFC5723] except that a GM would include the initial ticket request in a `GSA_AUTH` exchange instead of an `IKE_AUTH` exchange. G-IKEv2 is also compatible with Quantum Safe Key Exchange framework, defined in [I-D.tjhai-ipsecme-hybrid-qske-ikev2].

Some IKEv2 extensions however require special handling if used in G-IKEv2.

##### 1.5.1. Postquantum Preshared Keys for IKEv2

G-IKEv2 can take advantage of the protection provided by Postquantum Preshared Keys (PPK) for IKEv2 [I-D.ietf-ipsecme-qv-ikev2]. However, the use of PPK leaves the initial IKE SA susceptible to quantum computer (QC) attacks. So, if PPK was used for IKE SA setup, the GCKS MUST NOT send GSA and KD payloads in the `GSA_AUTH` response message. Instead, the GCKS MUST return a new notification `REKEY_IS_NEEDED`. Upon receiving this notification in the `GSA_AUTH` response the GM MUST perform an IKE SA rekey and then initiate a new `GSA_REGISTRATION` request for the same group. Below are possible scenarios involving using PPK.

GM begins `IKE_SA_INIT` requesting PPK, and GCKS responds with willingness to do it, or aborts according to its "mandatory\_or\_not" flag:

```

Initiator (Member)                      Responder (GCKS)
-----
HDR, SAi1, KEi, Ni, N(USE_PPK) --->
<--- HDR, SAr1, KEr, Nr, [CERTREQ],
      N(USE_PPK)

```

Figure 12: IKE\_SA\_INIT Exchange requesting using PPK

GM begins GSA\_AUTH with PPK\_ID; if using PPK is not mandatory for the GM, N(NO\_PPK\_AUTH) is included too:

```

Initiator (Member)                      Responder (GCKS)
-----
HDR, SK {IDi, AUTH, IDg,
N(PPK_IDENTITY), N(NO_PPK_AUTH) } --->

```

Figure 13: GSA\_AUTH Request using PPK

If GCKS has no such PPK and using PPK is not mandatory for it and N(NO\_PPK\_AUTH) is included, then the GCKS continues w/o PPK; in this case no rekey is needed:

```

Initiator (Member)                      Responder (GCKS)
-----
<--- HDR, SK { IDr, AUTH, GSA, KD }

```

Figure 14: GSA\_AUTH Response using no PPK

If GCKS has no such PPK and either N(NO\_PPK\_AUTH) is missing or using PPK is mandatory for GCKS, the GCKS aborts the exchange:

```

Initiator (Member)                      Responder (GCKS)
-----
<--- HDR, SK { N(AUTHENTICATION_FAILED) }

```

Figure 15: GSA\_AUTH Error Response

Assuming GCKS has a proper PPK the GCKS continues with request to GM to immediately perform a rekey:

```

Initiator (Member)                      Responder (GCKS)
-----
<--- HDR, SK{IDr, AUTH, N(PPK_IDENTITY),
      N(REKEY_IS_NEEDED) }

```

Figure 16: GSA\_AUTH Response using PPK

GM initiates CREATE\_CHILD\_SA to rekey IKE SA and then makes a new registration request for the same group over the new IKE SA:

Initiator (Member)	Responder (GCKS)
-----	-----
HDR, SK {SA, Ni, KEi } --->	
	<--- HDR, SK {SA, Nr, KEr }
HDR, SK {IDg } --->	
	<--- HDR, SK { GSA, KD }

Figure 17: Rekeying IKE SA followed by GSA\_REGISTRATION Exchange

## 2. Header and Payload Formats

Refer to IKEv2 [RFC7296] for existing payloads. Some payloads used in G-IKEv2 exchanges are not aligned to 4-octet boundaries, which is also the case for some IKEv2 payloads (see Section 3.2 of [RFC7296]).

### 2.1. The G-IKEv2 Header

G-IKEv2 uses the same IKE header format as specified in [RFC7296] section 3.1.

Several new payload formats are required in the group security exchanges.

Next Payload Type	Value
-----	-----
Group Identification (IDg)	50
Group Security Association (GSA)	51
Key Download (KD)	52

New exchange types GSA\_AUTH, GSA\_REGISTRATION and GSA\_REKEY are added to the IKEv2 [RFC7296] protocol.

Exchange Type	Value
-----	-----
GSA_AUTH	39
GSA_REGISTRATION	40
GSA_REKEY	41
GSA_INBAND_REKEY	TBD

Major Version is 2 and Minor Version is 0 as in IKEv2 [RFC7296]. IKE SA Initiator's SPI, IKE SA Responder's SPI, Flags, Message ID, and Length are as specified in [RFC7296].

## 2.2. Group Identification (IDg) Payload

The IDg Payload allows the group member to indicate which group it wants to join. The payload is constructed by using the IKEv2 Identification Payload (section 3.5 of [RFC7296]). ID type ID\_KEY\_ID MUST be supported. ID types ID\_IPV4\_ADDR, ID\_FQDN, ID\_RFC822\_ADDR, ID\_IPV6\_ADDR SHOULD be supported. ID types ID\_DER\_ASN1\_DN and ID\_DER\_ASN1\_GN are not expected to be used.

## 2.3. Security Association - GM Supported Transforms (Sag)

The Sag payload declares which Transforms a GM is willing to accept. The payload is constructed using the format of the IKEv2 Security Association payload (section 3.3 of [RFC7296]). The Payload Type for Sag is identical to the SA Payload Type (33).

## 2.4. Group Security Association Payload

The Group Security Association payload is used by the GCKS to assert security attributes for both Rekey and Data-security SAs.

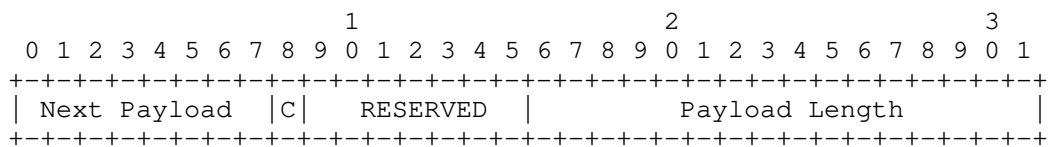


Figure 18: GSA Payload Format

The Security Association Payload fields are defined as follows:

- o Next Payload (1 octet) -- Identifies the next payload type for the G-IKEv2 registration or the G-IKEv2 rekey message.
- o Critical (1 bit) -- Set according to [RFC7296].
- o RESERVED (7 bits) -- Must be zero.
- o Payload Length (2 octets) -- Is the octet length of the current payload including the generic header and all TEK and KEK policies.

### 2.4.1. GSA Policy

Following the GSA generic payload header are GSA policies for group rekeying (KEK), data traffic SAs (TEK) and/or Group Associated Policy (GAP). There may be zero or one GSA KEK policy, zero or one GAP policies, and zero or more GSA TEK policies, where either one GSA KEK or GSA TEK payload MUST be present.

This latitude allows various group policies to be accommodated. For example if the group policy does not require the use of a Rekey SA, the GCKS would not need to send a GSA KEK attribute to the group member since all SA updates would be performed using the Registration SA. Alternatively, group policy might use a Rekey SA but choose to download a KEK to the group member only as part of the Registration SA. Therefore, the GSA KEK policy would not be necessary as part of the GSA\_REKEY message.

Specifying multiple GSA TEKs allows multiple related data streams (e.g., video, audio, and text) to be associated with a session, but each protected with an individual security association policy.

A GAP payload allows for the distribution of group-wise policy, such as instructions for when to activate and de-activate SAs.

Policies are distributed in substructures to the GSA payload, and include the following header.

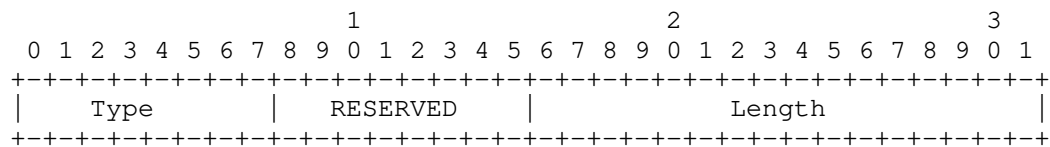


Figure 19: GSA Policy Generic Header Format

The payload fields are defined as follows:

- o Type (1 octet) -- Identifies the substructure type. In the following table the terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

Type	Value
-----	-----
Reserved	0
KEK	1
GAP	2
TEK	3
Unassigned	4-127
Private Use	128-255

- o RESERVED (1 octet) -- Unused, set to zero.
- o Length (2 octets) -- Length in octets of the substructure, including its header.

### 2.4.2. KEK Policy

The GSA KEK policy contains security attributes for the KEK method for a group and parameters specific to the G-IKEv2 registration operation. The source and destination traffic selectors describe the network identities used for the rekey messages.

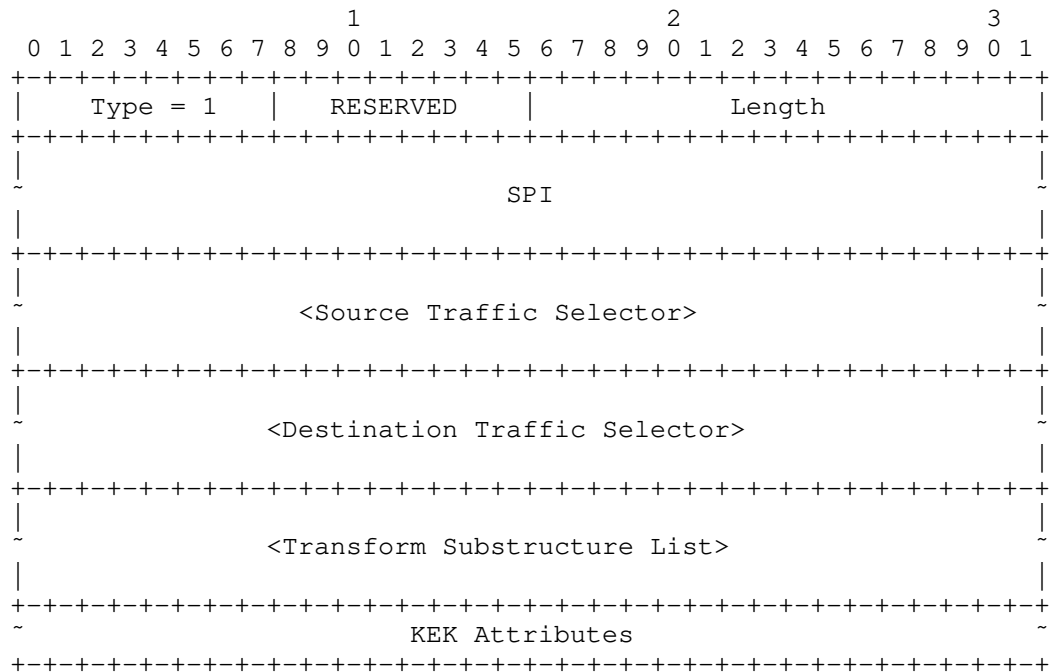


Figure 20: KEK Policy Format

The GSA KEK Payload fields are defined as follows:

- o Type = 1 (1 octet) -- Identifies the GSA payload type as KEK in the G-IKEv2 registration or the G-IKEv2 rekey message.
- o RESERVED (1 octet) -- Must be zero.
- o Length (2 octets) -- Length of this structure including KEK attributes.
- o SPI (16 octets) -- Security Parameter Index for the rekey message. The SPI must be the IKEv2 Header SPI pair where the first 8 octets become the "Initiator's SPI" field in the G-IKEv2 rekey message IKEv2 HDR, and the second 8 octets become the "Responder's SPI" in the same HDR. As described above, these SPIs are assigned by the

GCKS. When selecting SPI the GCKS MUST make sure that the sole first 8 octets (corresponding to "Initiator's SPI" field in the IKEv2 header) uniquely identify the Rekey SA.

- o Source & Destination Traffic Selectors -- Substructures describing the source and destination of the network identities. These identities refer to the source and destination of the next KEK rekey SA. Defined format and values are specified by IKEv2 [RFC7296], section 3.13.1.
- o Transform Substructure List -- A list of Transform Substructures specifies the transform information. The format is defined in IKEv2 [RFC7296], section 3.3.2, and values are described in the IKEv2 registries [IKEV2-IANA]. Valid Transform Types are ENCR, INTEG. The Last Substruc value in each Transform Substructure will be set to 3 except for the last one in the list, which is set to 0.
- o KEK Attributes -- Contains KEK policy attributes associated with the group. The following sections describe the possible attributes. Any or all attributes may be optional, depending on the group policy.

#### 2.4.2.1. KEK Attributes

The following attributes may be present in a GSA KEK policy. The attributes must follow the format defined in the IKEv2 [RFC7296] section 3.3.5. In the table, attributes that are defined as TV are marked as Basic (B); attributes that are defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

KEK Attributes	Value	Type	Mandatory
-----	----	----	-----
Reserved	0		
KEK_MANAGEMENT_ALGORITHM	1	B	N
Reserved	2		
Reserved	3		
KEK_KEY_LIFETIME	4	V	Y
Reserved	5		
KEK_AUTH_METHOD	6	B	Y
KEK_AUTH_HASH	7	B	N
KEK_MESSAGE_ID	8	V	Y (*)
KEK_NEXT_SPI	9	V	N
Unassigned	10-16383		
Private Use	16384-32767		



(\*) the KEK\_MESSAGE\_ID MUST be included in a G-IKEv2 registration message and MUST NOT be included in rekey messages.

The following attributes may only be included in a G-IKEv2 registration message: KEK\_MANAGEMENT\_ALGORITHM, KEK\_MESSAGE\_ID.

#### 2.4.2.1.1. KEK\_MANAGEMENT\_ALGORITHM

The KEK\_MANAGEMENT\_ALGORITHM attribute specifies the group KEK management algorithm used to provide forward or backward access control (i.e., used to exclude group members). Defined values are specified in the following table. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

KEK Management Type	Value
-----	-----
Reserved	0
LKH	1
Unassigned	2-16383
Private Use	16384-32767

#### 2.4.2.1.2. KEK\_KEY\_LIFETIME

The KEK\_KEY\_LIFETIME attribute specifies the maximum time for which the KEK is valid. The GCKS may refresh the KEK at any time before the end of the valid period. The value is a four (4) octet number defining a valid time period in seconds.

#### 2.4.2.1.3. KEK\_AUTH\_METHOD

The KEK\_AUTH\_METHOD attribute specifies the method of authentication used. This value is from the IKEv2 Authentication Method registry [IKEV2-IANA]. The method must either specify using some public key signatures or Shared Key Message Integrity Code. Other authentication methods MUST NOT be used.

#### 2.4.2.1.4. KEK\_AUTH\_HASH

The KEK\_AUTH\_HASH attribute specifies the hash algorithm used to generate the AUTH key to authenticate GSA\_REKEY messages. Hash algorithms are defined in IANA registry IKEv2 Hash Algorithms [IKEV2-IANA].

This attribute SHOULD NOT be sent if the KEK\_AUTH\_METHOD implies a particular hash algorithm (e.g., for DSA-based algorithms). Furthermore, it is not necessary for the GCKS to send it if the GM is known to support the algorithm because it declared it in a

SIGNATURE\_HASH\_ALGORITHMS notification during registration (see [RFC7427]).

#### 2.4.2.1.5. KEK\_MESSAGE\_ID

The KEK\_MESSAGE\_ID attribute defines the initial Message ID to be used by the GCKS in the GSA\_REKEY messages. The Message ID is a 4 octet unsigned integer in network byte order.

#### 2.4.2.1.6. KEK\_NEXT\_SPI

The KEK\_NEXT\_SPI attribute may optionally be included by GCKS in GSA\_REKEY message, indicating what IKE SPIs are intended be used for the next rekey SA. The attribute data MUST be 16 octets in length specifying the pair of IKE SPIs as they appear in the IKE header. Multiple attributes of this type MAY be included, meaning that any of the supplied SPIs can be used for the next rekey.

The GM may save these values and if later the GM starts receiving IKE messages with one of these SPIs without seeing a rekey message over the current rekey SA, this may be used as an indication, that the rekey message was lost on its way to this GM. In this case the GM SHOULD re-register to the group.

Note, that this method of detecting missed rekeys can only be used by passive GMs, i.e. those, that only listen and don't send data. It's also no point to include this attribute in the GSA\_INBAND\_REKEY messages, since they use reliable transport. Note also, that the GCKS is free to forget its promises and not to use the SPIs it sent in the KEK\_NEXT\_SPI attributes before (e.g. in case of GCKS reboot), so the GM must only treat these information as a "best effort" made by GCKS to prepare for future rekeys.

#### 2.4.3. GSA TEK Policy

The GSA TEK policy contains security attributes for a single TEK associated with a group.

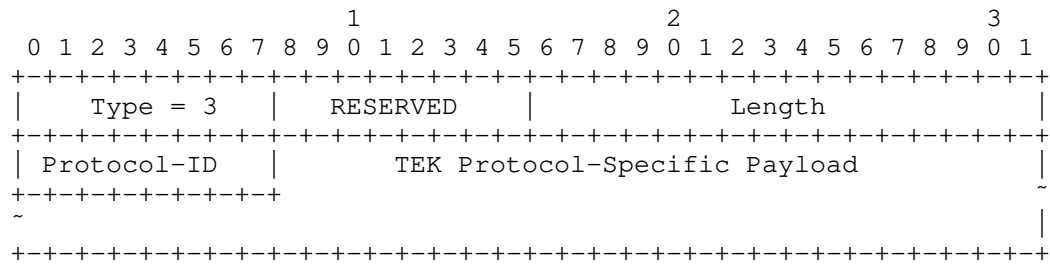


Figure 21: TEK Policy Generic Header Format

The GSA TEK Payload fields are defined as follows:

- o Type = 3 (1 octet) -- Identifies the GSA payload type as TEK in the G-IKEv2 registration or the G-IKEv2 rekey message.
- o RESERVED (1 octet) -- Must be zero.
- o Length (2 octets) -- Length of this structure, including the TEK Protocol-Specific Payload.
- o Protocol-ID (1 octet) -- Value specifying the Security Protocol. The following table defines values for the Security Protocol. Support for the GSA\_PROTO\_IPSEC\_AH GSA TEK is OPTIONAL. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

Protocol ID	Value
-----	-----
Reserved	0
GSA_PROTO_IPSEC_ESP	1
GSA_PROTO_IPSEC_AH	2
Unassigned	3-127
Private Use	128-255

- o TEK Protocol-Specific Payload (variable) -- Payload which describes the attributes specific for the Protocol-ID.

#### 2.4.3.1. TEK ESP and AH Protocol-Specific Policy

The TEK Protocol-Specific policy contains two traffic selectors one for the source and one for the destination of the protected traffic, SPI, Transforms, and Attributes.

The TEK Protocol-Specific policy for ESP and AH is as follows:

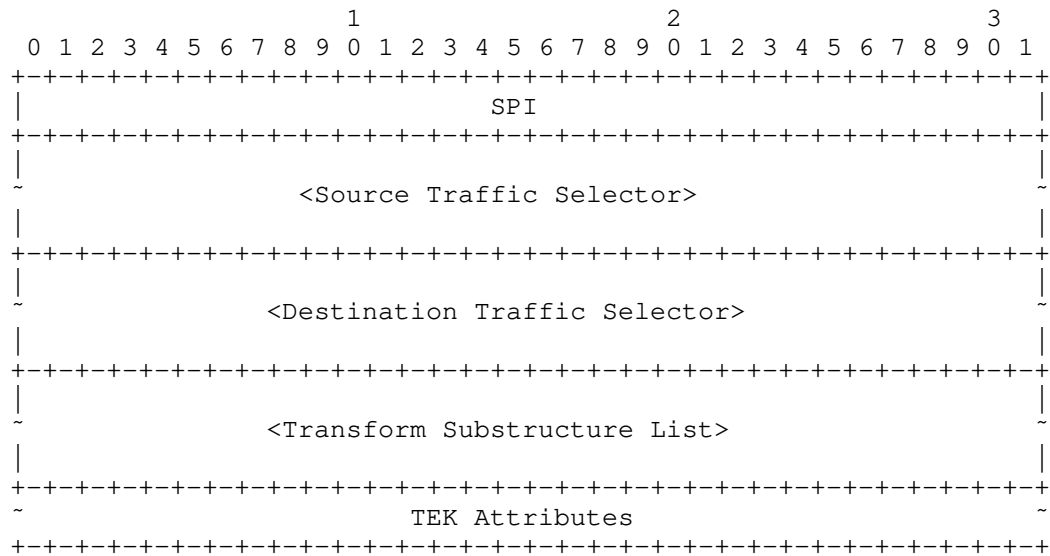


Figure 22: AH and ESP TEK Policy Format

The GSA TEK Policy fields are defined as follows:

- o SPI (4 octets) -- Security Parameter Index.
- o Source & Destination Traffic Selectors - The traffic selectors describe the source and the destination of the protected traffic. The format and values are defined in IKEv2 [RFC7296], section 3.13.1.
- o Transform Substructure List -- A list of Transform Substructures specifies the transform information. The format is defined in IKEv2 [RFC7296], section 3.3.2, and values are described in the IKEv2 registries [IKEV2-IANA]. Valid Transform Types for ESP are ENCR, INTEG, and ESN. Valid Transform Types for AH are INTEG and ESN. The Last Substruc value in each Transform Substructure will be set to 3 except for the last one in the list, which is set to 0. A Transform Substructure with attributes (e.g., the ENCR Key Length), they are included within the Transform Substructure as usual.
- o TEK Attributes -- Contains the TEK policy attributes associated with the group, in the format defined in Section 3.3.5 of [RFC7296]. All attributes are optional, depending on the group policy.

Attribute Types are as follows. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

TEK Attributes -----	Value -----	Type ----	Mandatory -----
Reserved	0		
TEK_KEY_LIFETIME	1	V	N
TEK_MODE	2	B	Y
TEK_REKEY_SPI	3	V	N
TEK_NEXT_SPI	4	V	N
Unassigned	5-16383		
Private Use	16384-32767		

It is NOT RECOMMENDED that the GCKS distribute both ESP and AH Protocol-Specific Policies for the same set of Traffic Selectors.

#### 2.4.3.1.1. TEK\_KEY\_LIFETIME

The TEK\_KEY\_LIFETIME attribute specifies the maximum time for which the TEK is valid. When the TEK expires, the AH or ESP security association and all keys downloaded under the security association are discarded. The GCKS may refresh the TEK at any time before the end of the valid period.

The value is a four (4) octet number defining a valid time period in seconds. If unspecified the default value of 28800 seconds (8 hours) shall be assumed.

#### 2.4.3.1.2. TEK\_MODE

The value of 0 is used for tunnel mode and 1 for transport mode. In the absence of this attribute tunnel mode will be used.

#### 2.4.3.1.3. TEK\_REKEY\_SPI

This attribute contains an SPI for the SA that is being rekeyed. The size of SPI depends on the protocol, for ESP and AH it is 4 octets, so the size of the data MUST be 4 octets for AH and ESP.

If this attribute is included in the rekey message, the GM SHOULD delete the SA corresponding to this SPI once the new SA is installed and regardless of the expiration time of the SA to be deleted (but after waiting DEACTIVATION\_TIME\_DELAY time period).

## 2.4.3.1.4. TEK\_NEXT\_SPI

This attribute contains an SPI that the GCKS reserved for the next rekey. The size of SPI depends on the protocol, for ESP and AH it is 4 octets, so the size of the data MUST be 4 octets for AH and ESP. Multiple attributes of this type MAY be included, which means that any of the provided SPIs can be used in the next rekey.

The GM may save these values and if later the GM starts receiving IPsec messages with one of these SPIs without seeing a rekey message for it, this may be used as an indication, that the rekey message was lost on its way to this GM. In this case the GM SHOULD re-register to the group.

Note, that this method of detecting missed rekey messages can only be used by passive GMs, i.e. those, that only listen and don't send data. It's also no point to include this attribute in the GSA\_INBAND\_REKEY messages, since they use reliable transport. Note also, that the GCKS is free to forget its promises and not to use the SPIs it sent in the TEK\_NEXT\_SPI attributes before (e.g. in case of GCKS reboot), so the GM must only treat these information as a "best effort" made by GCKS to prepare for future rekeys.

## 2.4.4. GSA Group Associated Policy

Group specific policy that does not belong to rekey policy (GSA KEK) or traffic encryption policy (GSA TEK) can be distributed to all group member using GSA GAP (Group Associated Policy).

The GSA GAP payload is defined as follows:

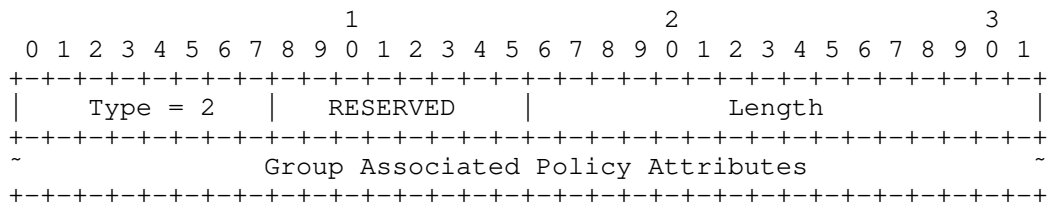


Figure 23: GAP Policy Format

The GSA GAP payload fields are defined as follows:

- o Type = 2 (1 octet) -- Identifies the GSA payload type as GAP in the G-IKEv2 registration or the G-IKEv2 rekey message.
- o RESERVED (1 octet) -- Must be zero.

- o Length (2 octets) -- Length of this structure, including the GSA GAP header and Attributes.
- o Group Associated Policy Attributes (variable) -- Contains attributes following the format defined in Section 3.3.5 of [RFC7296].

Attribute Types are as follows. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

Attribute Type	Value	Type
-----	-----	----
Reserved	0	
ACTIVATION_TIME_DELAY	1	B
DEACTIVATION_TIME_DELAY	2	B
Unassigned	3-16383	
Private Use	16384-32767	

#### 2.4.4.1. ACTIVATION\_TIME\_DELAY/DEACTIVATION\_TIME\_DELAY

Section 4.2.1 of [RFC5374] specifies a key rollover method that requires two values be provided to group members. The ACTIVATION\_TIME\_DELAY attribute allows a GCKS to set the Activation Time Delay (ATD) for SAs generated from TEKs. The ATD defines how long after receiving new SAs that they are to be activated by the GM. The ATD value is in seconds.

The DEACTIVATION\_TIME\_DELAY allows the GCKS to set the Deactivation Time Delay (DTD) for previously distributed SAs. The DTD defines how long after receiving new SAs it should deactivate SAs that are destroyed by the rekey event. The value is in seconds.

The values of ATD and DTD are independent. However, the DTD value should be larger, which allows new SAs to be activated before older SAs are deactivated. Such a policy ensures that protected group traffic will always flow without interruption.

#### 2.5. Key Download Payload

The Key Download Payload contains the group keys for the group specified in the GSA Payload. These key download payloads can have several security attributes applied to them based upon the security policy of the group as defined by the associated GSA Payload.

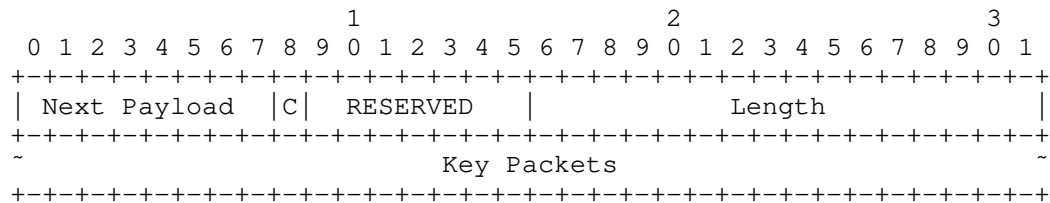


Figure 24: Key Download Payload Format

The Key Download Payload fields are defined as follows:

- o Next Payload (1 octet) -- Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be zero.
- o Critical (1 bit) -- Set according to [RFC7296].
- o RESERVED (7 bits) -- Unused, set to zero.
- o Payload Length (2 octets) -- Length in octets of the current payload, including the generic payload header.
- o Key Packets (variable) -- Contains Key Packets. Several types of key packets are defined. Each Key Packet has the following format.

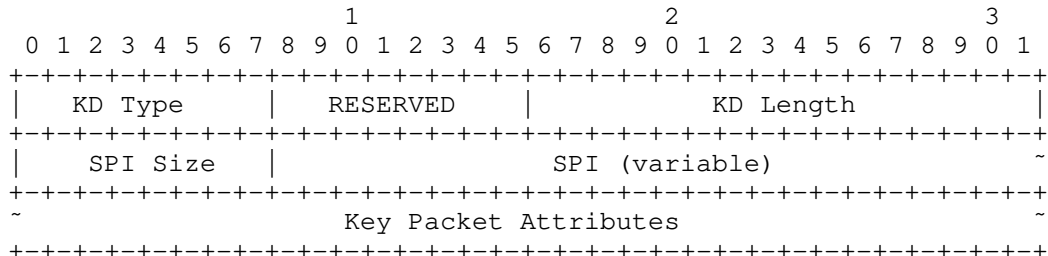


Figure 25: Key Packet Format

- o Key Download (KD) Type (1 octet) -- Identifier for the Key Data field of this Key Packet. In the following table the terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.



Key Download Type	Value
-----	-----
Reserved	0
TEK	1
KEK	2
LKH	3
SID	4
Unassigned	5-127
Private Use	128-255

- o RESERVED (1 octet) -- Unused, set to zero.
- o Key Download Length (2 octets) -- Length in octets of the Key Packet data, including the Key Packet header.
- o SPI Size (1 octet) -- Value specifying the length in octets of the SPI as defined by the Protocol-Id.
- o SPI (variable length) -- Security Parameter Index which matches a SPI previously sent in an GSA KEK or GSA TEK Payload.
- o Key Packet Attributes (variable length) -- Contains Key information. The format of this field is specific to the value of the KD Type field. The following sections describe the format of each KD Type.

#### 2.5.1. TEK Download Type

The following attributes may be present in a TEK Download Type. Exactly one attribute matching each type sent in the GSA TEK payload MUST be present. The attributes must follow the format defined in IKEv2 (Section 3.3.5 of [RFC7296]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

TEK KD Attributes	Value	Type	Mandatory
-----	-----	-----	-----
Reserved	0-2		
TEK_KEYMAT	3	V	Y
Unassigned	4-16383		
Private Use	16384-32767		

It is possible that the GCKS will send no TEK key packets in a Registration KD payload (as well as no corresponding GSA TEK payloads in the GSA payload), after which the TEK payloads will be sent in a rekey message.

## 2.5.1.1. TEK\_KEYMAT

The TEK\_KEYMAT attribute contains keying material for the corresponding SPI. This keying material will be used with the transform specified in the GSA TEK payload. The keying material is treated equivalent to IKEv2 KEYMAT derived for that IPsec transform.

## 2.5.2. KEK Download Type

The following attributes may be present in a KEK Download Type. Exactly one attribute matching each type sent in the GSA KEK payload MUST be present. The attributes must follow the format defined in IKEv2 (Section 3.3.5 of [RFC7296]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

KEK KD Attributes	Value	Type	Mandatory
-----	----	----	-----
Reserved	0		
KEK_ENCR_KEY	1	V	Y
KEK_INTEGRITY_KEY	2	V	N
KEK_AUTH_KEY	3	V	N
Unassigned	4-16383		
Private Use	16384-32767		

If the KEK Key Packet is included, there MUST be only one present in the KD payload.

## 2.5.2.1. KEK\_ENCR\_KEY

The KEK\_ENCR\_KEY attribute type declares that the encryption key for the corresponding SPI is contained in the Key Packet Attribute. The encryption algorithm that will use this key was specified in the GSA KEK payload.

## 2.5.2.2. KEK\_INTEGRITY\_KEY

The KEK\_INTEGRITY\_KEY attribute type declares the integrity key for this SPI is contained in the Key Packet Attribute. The integrity algorithm that will use this key was specified in the GSA KEK payload.

### 2.5.2.3. KEK\_AUTH\_KEY

The KEK\_AUTH\_KEY attribute type declares that the authentication key for this SPI is contained in the Key Packet Attribute. The signature algorithm that will use this key was specified in the GSA KEK payload. An RSA public key format is defined in [RFC3447], Section A.1.1. DSS public key format is defined in [RFC3279] Section 2.3.2. For ECDSA Public keys, use format described in [RFC5480] Section 2.2. Other algorithms added to the IKEv2 Authentication Method registry are also expected to include a format of the public key included in the algorithm specification.

### 2.5.3. LKH Download Type

The LKH key packet is comprised of attributes representing different leaves in the LKH key tree.

The following attributes are used to pass an LKH KEK array in the KD payload. The attributes must follow the format defined in IKEv2 (Section 3.3.5 of [RFC7296]). In the table, attributes defined as TV are marked as Basic (B); attributes defined as TLV are marked as Variable (V). The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

LKH KD Attributes	Value	Type
-----	-----	----
Reserved	0	
LKH_DOWNLOAD_ARRAY	1	V
LKH_UPDATE_ARRAY	2	V
Unassigned	3-16383	
Private Use	16384-32767	

If an LKH key packet is included in the KD payload, there MUST be only one present.

#### 2.5.3.1. LKH\_DOWNLOAD\_ARRAY

The LKH\_DOWNLOAD\_ARRAY attribute type is used to download a set of LKH keys to a group member. It MUST NOT be included in a IKEv2 rekey message KD payload if the IKEv2 rekey is sent to more than one group member. If an LKH\_DOWNLOAD\_ARRAY attribute is included in a KD payload, there MUST be only one present.

This attribute consists of a header block, followed by one or more LKH keys.

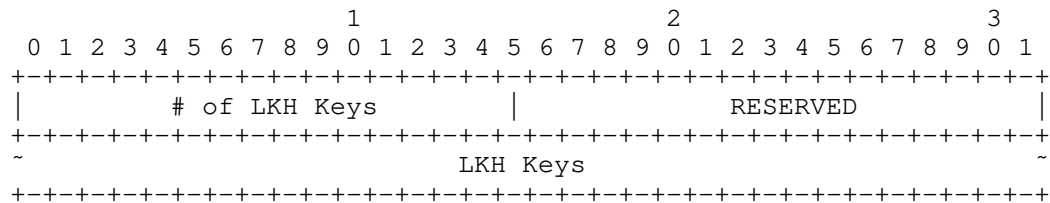


Figure 26: LKH\_DOWNLOAD\_ARRAY Format

The KEK\_LKH attribute fields are defined as follows:

- o Number of LKH Keys (2 octets) -- This value is the number of distinct LKH keys in this sequence.
- o RESERVED (2 octets) -- Unused, set to zero.

Each LKH Key is defined as follows:

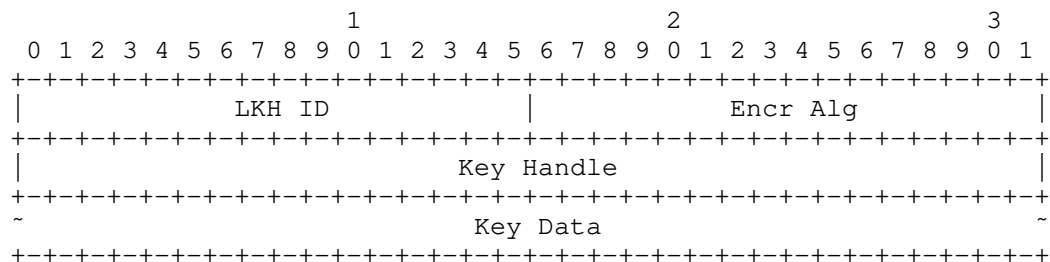


Figure 27: LKH Key Format

- o LKH ID (2 octets) -- This is the position of this key in the binary tree structure used by LKH.
- o Encr Alg (2 octets) -- This is the encryption algorithm for which this key data is to be used. This value is specified in the ENCR transform in the GSA payload.
- o Key Handle (4 octets) -- This is a randomly generated value to uniquely identify a key within an LKH ID.
- o Key Data (variable length) -- This is the actual encryption key data, which is dependent on the Encr Alg algorithm for its format.

The first LKH Key structure in an LKH\_DOWNLOAD\_ARRAY attribute contains the Leaf identifier and key for the group member. The rest of the LKH Key structures contain keys along the path of the key tree in the order starting from the leaf, culminating in the group KEK.

### 2.5.3.2. LKH\_UPDATE\_ARRAY

The LKH\_UPDATE\_ARRAY attribute type is used to update the LKH keys for a group. It is most likely to be included in a G-IKEv2 rekey message KD payload to rekey the entire group. This attribute consists of a header block, followed by one or more LKH keys, as defined in Section 2.5.3.1.

There may be any number of LKH\_UPDATE\_ARRAY attributes included in a KD payload.

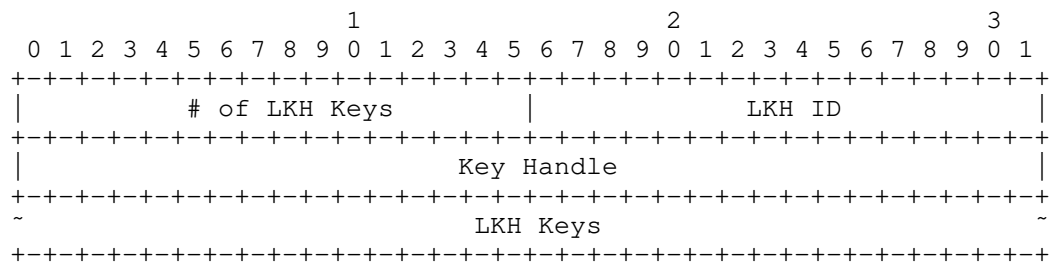


Figure 28: LKH\_UPDATE\_ARRAY Format

- o Number of LKH Keys (2 octets) -- This value is the number of distinct LKH keys in this sequence.
- o LKH ID (2 octets) -- This is the node identifier associated with the key used to encrypt the first LKH Key.
- o Key Handle (4 octets) -- This is the value that uniquely identifies the key within the LKH ID which was used to encrypt the first LKH key.

The LKH Keys are as defined in Section 2.5.3.1. The LKH Key structures contain keys along the path of the key tree in the order from the LKH ID found in the LKH\_UPDATE\_ARRAY header, culminating in the group KEK. The Key Data field of each LKH Key is encrypted with the LKH key preceding it in the LKH\_UPDATE\_ARRAY attribute. The first LKH Key is encrypted under the key defined by the LKH ID and Key Handle found in the LKH\_UPDATE\_ARRAY header.

### 2.5.4. SID Download Type

The SID attribute is used to download one or more Sender-ID (SID) values for the exclusive use of a group member. The terms Reserved, Unassigned, and Private Use are to be applied as defined in [RFC8126]. The registration procedure is Expert Review.

SID KD Attributes	Value	Type
-----	-----	-----
Reserved	0	
NUMBER_OF_SID_BITS	1	B
SID_VALUE	2	V
Unassigned	3-16383	
Private Use	16384-32767	

Because a SID value is intended for a single group member, the SID Download type MUST NOT be distributed in a GSA\_REKEY message distributed to multiple group members.

#### 2.5.4.1. NUMBER\_OF\_SID\_BITS

The NUMBER\_OF\_SID\_BITS attribute type declares how many bits of the cipher nonce in which to represent an SID value. The bits are applied as the most significant bits of the IV, as shown in Figure 1 of [RFC6054] and specified in Section 1.4.6.2. Guidance for a GCKS choosing the NUMBER\_OF\_SID\_BITS is provided in Section 3 of [RFC6054].

This value is applied to each SID value distributed in the SID Download.

#### 2.5.4.2. SID\_VALUE

The SID\_VALUE attribute type declares a single SID value for the exclusive use of this group member. Multiple SID\_VALUE attributes MAY be included in a SID Download.

#### 2.5.4.3. GM Semantics

The SID\_VALUE attribute value distributed to the group member MUST be used by that group member as the SID field portion of the IV for all Data-Security SAs including a counter-based mode of operation distributed by the GCKS as a part of this group. When the Sender-Specific IV (SSIV) field for any Data-Security SA is exhausted, the group member MUST NOT act as a sender on that SA using its active SID. The group member SHOULD re-register, at which time the GCKS will issue a new SID to the group member, along with either the same Data-Security SAs or replacement ones. The new SID replaces the existing SID used by this group member, and also resets the SSIV value to its starting value. A group member MAY re-register prior to the actual exhaustion of the SSIV field to avoid dropping data packets due to the exhaustion of available SSIV values combined with a particular SID value.

A group member MUST ignore an SID Download Type KD payload present in a GSA-REKEY message, otherwise more than one GM may end up using the same SID.

#### 2.5.4.4. GCKS Semantics

If any KD payload includes keying material that is associated with a counter-mode of operation, an SID Download Type KD payload containing at least one SID\_VALUE attribute MUST be included. The GCKS MUST NOT send the SID Download Type KD payload as part of a GSA\_REKEY message, because distributing the same sender-specific policy to more than one group member will reduce the security of the group.

#### 2.6. Delete Payload

There are occasions when the GCKS may want to signal to group members to delete policy at the end of a broadcast, if group policy has changed, or the GCKS needs to reset the policy and keying material for the group due to an emergency. Deletion of keys MAY be accomplished by sending an IKEv2 Delete Payload, section 3.11 of [RFC7296] as part of a registration or rekey Exchange. Whenever an SA is to be deleted, the GCKS SHOULD send the Delete Payload in both registration and rekey exchanges, because GMs with previous group policy may contact the GCKS using either exchange.

The Protocol ID MUST be 41 for GSA\_REKEY Exchange, 2 for AH or 3 for ESP. Note that only one protocol id value can be defined in a Delete payload. If a TEK and a KEK SA for GSA\_REKEY Exchange must be deleted, they must be sent in different Delete payloads. Similarly, if a TEK specifying ESP and a TEK specifying AH need to be deleted, they must be sent in different Delete payloads.

There may be circumstances where the GCKS may want to reset the policy and keying material for the group. The GCKS can signal deletion of all policy of a particular TEK by sending a TEK with a SPI value equal to zero in the delete payload. In the event that the administrator is no longer confident in the integrity of the group they may wish to remove all KEK and all the TEKs in the group. This is done by having the GCKS send a delete payload with a SPI of zero and a Protocol-ID of AH or ESP to delete all TEKs, followed by another delete payload with a SPI value of zero and Protocol-ID of KEK SA to delete the KEK SA.

#### 2.7. Notify Payload

G-IKEv2 uses the same Notify payload as specified in [RFC7296], section 3.10.

There are additional Notify Message types introduced by G-IKEv2 to communicate error conditions and status.

NOTIFY messages - error types	Value
-----	-----
INVALID_GROUP_ID -	45
AUTHORIZATION_FAILED -	46
REGISTRATION_FAILED -	TBD

INVALID\_GROUP\_ID indicates the group id sent during the registration process is invalid.

AUTHORIZATION\_FAILED is sent in the response to a GSA\_AUTH message when authorization failed.

REGISTRATION\_FAILED is sent by the GCKS when the GM registration request cannot be satisfied.

NOTIFY messages - status types	Value
-----	-----
SENDER -	16429
REKEY_IS_NEEDED -	TBD

SENDER notification is sent in GSA\_AUTH or GSA\_REGISTRATION to indicate that the GM intends to be sender of data traffic. The data includes a count of how many SID values the GM desires. The count MUST be 4 octets long and contain the big endian representation of the number of requested SIDs.

REKEY\_IS\_NEEDED is sent in GSA\_AUTH response message to indicate that the GM must perform an immediate rekey of IKE SA to make it secure against quantum computers and then start a registration request over.

## 2.8. Authentication Payload

G-IKEv2 uses the same Authentication payload as specified in [RFC7296], section 3.8, to sign the rekey message.

## 3. Security Considerations

### 3.1. GSA Registration and Secure Channel

G-IKEv2 registration exchange uses IKEv2 IKE\_SA\_INIT protocols, inheriting all the security considerations documented in [RFC7296] section 5 Security Considerations, including authentication, confidentiality, protection against man-in-the-middle, protection against replay/reflection attacks, and denial of service protection. The GSA\_AUTH and GSA\_REGISTRATION exchanges also take advantage of



those protections. In addition, G-IKEv2 brings in the capability to authorize a particular group member regardless of whether they have the IKEv2 credentials.

### 3.2. GSA Maintenance Channel

The GSA maintenance channel is cryptographically and integrity protected using the cryptographic algorithm and key negotiated in the GSA member registration exchanged.

#### 3.2.1. Authentication/Authorization

Authentication is implicit, the public key of the identity is distributed during the registration, and the receiver of the rekey message uses that public key and identity to verify the message came from the authorized GCKS.

#### 3.2.2. Confidentiality

Confidentiality is provided by distributing a confidentiality key as part of the GSA member registration exchange.

#### 3.2.3. Man-in-the-Middle Attack Protection

GSA maintenance channel is integrity protected by using a digital signature.

#### 3.2.4. Replay/Reflection Attack Protection

The GSA\_REKEY message includes a monotonically increasing sequence number to protect against replay and reflection attacks. A group member will recognize a replayed message by comparing the Message ID number to that of the last received rekey message, any rekey message containing a Message ID number less than or equal to the last received value MUST be discarded. Implementations should keep a record of recently received GSA rekey messages for this comparison.

## 4. IANA Considerations

### 4.1. New Registries

A new set of registries should be created for G-IKEv2, on a new page titled Group Key Management using IKEv2 (G-IKEv2) Parameters. The following registries should be placed on that page. The terms Reserved, Expert Review and Private Use are to be applied as defined in [RFC8126].

GSA Policy Type Registry, see Section 2.4.1

KEK Attributes Registry, see Section 2.4.2.1

KEK Management Algorithm Registry, see Section 2.4.2.1.1

GSA TEK Payload Protocol ID Type Registry, see Section 2.4.3

TEK Attributes Registry, see Section 2.4.3

Key Download Type Registry, see Section 2.5

TEK Download Type Attributes Registry, see Section 2.5.1

KEK Download Type Attributes Registry, see Section 2.5.2

LKH Download Type Attributes Registry, see Section 2.5.3

SID Download Type Attributes Registry, see Section 2.5.4

#### 4.2. New Payload and Exchange Types Added to the Existing IKEv2 Registry

The following new payloads and exchange types specified in this memo have already been allocated by IANA and require no further action, other than replacing the draft name with an RFC number.

The present document describes new IKEv2 Next Payload types, see Section 2.1

The present document describes new IKEv2 Exchanges types, see Section 2.1

The present document describes new IKEv2 notification types, see Section 2.7

#### 4.3. Changes to Previous Allocations

Section 4.7 indicates an allocation in the IKEv2 Notify Message Types - Status Types registry has been made. This NOTIFY type was allocated earlier in the development of G-IKEv2. The number is 16429, and was allocated with the name SENDER\_REQUEST\_ID. The name should be changed to SENDER.

#### 5. Acknowledgements

The authors thank Lakshminath Dondeti and Jing Xiang for first exploring the use of IKEv2 for group key management and providing the basis behind the protocol. Mike Sullenberger and Amjad Inamdar were

instrumental in helping resolve many issues in several versions of the document.

## 6. Contributors

The following individuals made substantial contributions to early versions of this memo.

Sheela Rowles  
Cisco Systems  
170 W. Tasman Drive  
San Jose, California 95134-1706  
USA

Phone: +1-408-527-7677  
Email: sheela@cisco.com

Aldous Yeung  
Cisco Systems  
170 W. Tasman Drive  
San Jose, California 95134-1706  
USA

Phone: +1-408-853-2032  
Email: cyyeung@cisco.com

Paulina Tran  
Cisco Systems  
170 W. Tasman Drive  
San Jose, California 95134-1706  
USA

Phone: +1-408-526-8902  
Email: ptran@cisco.com

Yoav Nir  
Dell EMC  
9 Andrei Sakharov St  
Haifa 3190500  
Israel

Email: ynir.ietf@gmail.com

## 7. References

## 7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2627] Wallner, D., Harder, E., and R. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, DOI 10.17487/RFC2627, June 1999, <<https://www.rfc-editor.org/info/rfc2627>>.
- [RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, DOI 10.17487/RFC3740, March 2004, <<https://www.rfc-editor.org/info/rfc3740>>.
- [RFC4046] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, DOI 10.17487/RFC4046, April 2005, <<https://www.rfc-editor.org/info/rfc4046>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC6054] McGrew, D. and B. Weis, "Using Counter Modes with Encapsulating Security Payload (ESP) and Authentication Header (AH) to Protect Group Traffic", RFC 6054, DOI 10.17487/RFC6054, November 2010, <<https://www.rfc-editor.org/info/rfc6054>>.
- [RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <<https://www.rfc-editor.org/info/rfc7296>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 7.2. Informative References

- [I-D.ietf-ipsecme-qr-ikev2]  
Fluhrer, S., McGrew, D., Kampanakis, P., and V. Smyslov,  
"Postquantum Preshared Keys for IKEv2", draft-ietf-  
ipsecme-qr-ikev2-08 (work in progress), March 2019.
- [I-D.tjhai-ipsecme-hybrid-qske-ikev2]  
Tjhai, C., Tomlinson, M., grbartle@cisco.com, g., Fluhrer,  
S., Geest, D., Garcia-Morchon, O., and V. Smyslov,  
"Framework to Integrate Post-quantum Key Exchanges into  
Internet Key Exchange Protocol Version 2 (IKEv2)", draft-  
tjhai-ipsecme-hybrid-qske-ikev2-03 (work in progress),  
January 2019.
- [IKEV2-IANA]  
IANA, "Internet Key Exchange Version 2 (IKEv2)  
Parameters", February 2016,  
<[http://www.iana.org/assignments/ikev2-parameters/  
ikev2-parameters.xhtml#ikev2-parameters-7](http://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml#ikev2-parameters-7)>.
- [NNL]  
Naor, D., Noal, M., and J. Lotspiech, "Revocation and  
Tracing Schemes for Stateless Receivers", Advances in  
Cryptography, Crypto '01, Springer-Verlag LNCS 2139, 2001,  
pp. 41-62, 2001,  
<<http://www.wisdom.weizmann.ac.il/~naor/>>.
- [OFT]  
McGrew, D. and A. Sherman, "Key Establishment in Large  
Dynamic Groups Using One-Way Function Trees", Manuscript,  
submitted to IEEE Transactions on Software Engineering,  
1998, <[http://download.nai.com/products/media/nai/misc/  
oft052098.ps](http://download.nai.com/products/media/nai/misc/oft052098.ps)>.
- [RFC2409]  
Harkins, D. and D. Carrel, "The Internet Key Exchange  
(IKE)", RFC 2409, DOI 10.17487/RFC2409, November 1998,  
<<https://www.rfc-editor.org/info/rfc2409>>.
- [RFC3279]  
Bassham, L., Polk, W., and R. Housley, "Algorithms and  
Identifiers for the Internet X.509 Public Key  
Infrastructure Certificate and Certificate Revocation List  
(CRL) Profile", RFC 3279, DOI 10.17487/RFC3279, April  
2002, <<https://www.rfc-editor.org/info/rfc3279>>.
- [RFC3447]  
Jonsson, J. and B. Kaliski, "Public-Key Cryptography  
Standards (PKCS) #1: RSA Cryptography Specifications  
Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February  
2003, <<https://www.rfc-editor.org/info/rfc3447>>.

- [RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, DOI 10.17487/RFC3686, January 2004, <<https://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, <<https://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, <<https://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, <<https://www.rfc-editor.org/info/rfc4543>>.
- [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, <<https://www.rfc-editor.org/info/rfc5374>>.
- [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <<https://www.rfc-editor.org/info/rfc5480>>.
- [RFC5723] Sheffer, Y. and H. Tschofenig, "Internet Key Exchange Protocol Version 2 (IKEv2) Session Resumption", RFC 5723, DOI 10.17487/RFC5723, January 2010, <<https://www.rfc-editor.org/info/rfc5723>>.
- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <<https://www.rfc-editor.org/info/rfc6407>>.
- [RFC6467] Kivinen, T., "Secure Password Framework for Internet Key Exchange Version 2 (IKEv2)", RFC 6467, DOI 10.17487/RFC6467, December 2011, <<https://www.rfc-editor.org/info/rfc6467>>.
- [RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation", RFC 7383, DOI 10.17487/RFC7383, November 2014, <<https://www.rfc-editor.org/info/rfc7383>>.

- [RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in the Internet Key Exchange Version 2 (IKEv2)", RFC 7427, DOI 10.17487/RFC7427, January 2015, <<https://www.rfc-editor.org/info/rfc7427>>.
- [RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <<https://www.rfc-editor.org/info/rfc8229>>.

## Appendix A. Use of LKH in G-IKEv2

Section 5.4 of [RFC2627] describes the LKH architecture, and how a GCKS uses LKH to exclude group members. This section clarifies how the LKH architecture is used with G-IKEv2.

### A.1. Group Creation

When a GCKS forms a group, it creates a key tree as shown in the figure below. The key tree contains logical keys (represented as numbers in the figure) and a private key shared with only a single GM (represented as letters in the figure). Note that the use of numbers and letters is used for explanatory purposes; in fact, each key would have an LKH ID, which is two-octet identifier chosen by the GCKS. The GCKS may create a complete tree as shown, or a partial tree which is created on demand as members join the group. The top of the key tree (i.e., "1" in Figure 29) is used as the KEK for the group.

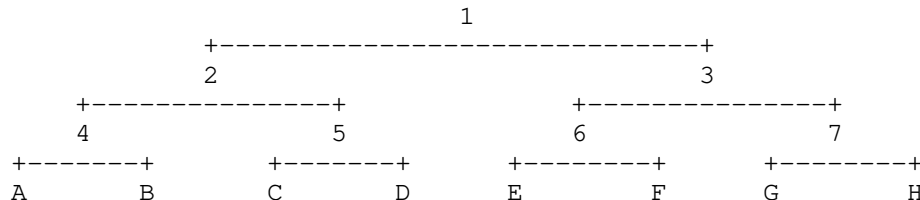


Figure 29: Initial LKH tree

When GM "A" joins the group, the GCKS provides an LKH\_DOWNLOAD\_ARRAY in the KD payload of the GSA\_AUTH or GSA\_REGISTRATION exchange. Given the tree shown in figure above, the LKH\_DOWNLOAD\_ARRAY will contain four LKH Key payloads, each containing an LKH ID and Key Data. If the LKH ID values were chosen as shown in the figure, four LKH Keys would be provided to GM "A", in the following order: A, 4, 2, 1. When GM "B" joins the group, it would also be given four LKH Keys in the following order: B, 4, 2, 1. And so on, until GM "H" joins the group and is given H, 7, 3, 1.

## A.2. Group Member Exclusion

If the GKCS has reason to believe that a GM should be excluded, then it can do so by sending a GSA\_REKEY exchange that includes a set of LKH\_UPDATE\_ARRAY attributes in the KD payload. Each LKH\_UPDATE\_ARRAY contains a set of LKH Key payloads, in which every GM other than the excluded GM will be able to determine a set of new logical keys, which culminate in a new key "1". The excluded GM will observe the set of LKH\_UPDATE\_ARRAY attributes, but cannot determine the new logical keys because each of the "Key Data" fields is encrypted with a key held by other GMs. The GM will hold no keys to properly decrypt any of the "Key Data" fields, including key "1" (i.e., the new KEK). When a subsequent GSA\_REKEY exchange is delivered by the GKCS and protected by the new KEK, the excluded GM will no longer be able to see the contents of the GSA\_REKEY, including new TEKs that will be delivered to replace existing TEKs. At this point, the GM will no longer be able to participate in the group.

In the example below, new keys are represented as the number followed by a "prime" symbol (e.g., "1" becomes "1'"). Each key is encrypted by another key. This is represented as "{key1}key2", where key2 encrypts key1. For example, "{1'}2'" states that a new key "1'" is encrypted with a new key "2'".

If GM "B" is to be excluded, the GKCS will need to include three LKH\_UPDATE\_ARRAY attributes in the GSA\_REKEY message. The order of the attributes does not matter; only the order of the keys within each attribute.

- o One will provide GM "A" with new logical keys that are shared with B: {4'}A, {2'}4', {1'}2'
- o One will provide all GMs holding key "5" with new logical keys: {2'}5, {1'}2'
- o One will provide all GMs holding key "3" with a new KEK: {1'}3

Each GM will look at each LKH\_UPDATE\_ARRAY attribute and observe an LKH ID which is present in an LKH Key delivered to them in the LKH\_DOWNLOAD\_ARRAY they were given. If they find a matching LKH ID, then they will decrypt the new key with the logical key immediately preceding that LKH Key, and so on until they have received the new 1' key.

The resulting key tree from this rekey event would be shown in Figure 30.



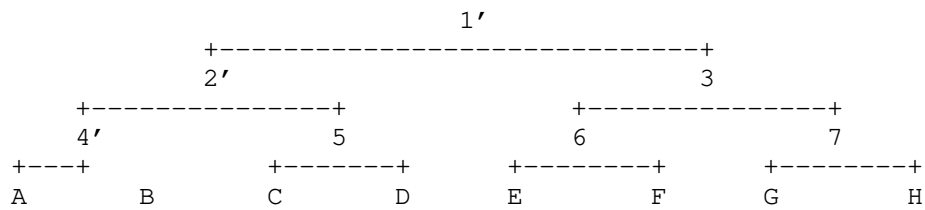


Figure 30: LKH tree after B has been excluded

## Authors' Addresses

Brian Weis  
Independent  
USA

Email: [bew.stds@gmail.com](mailto:bew.stds@gmail.com)

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

Phone: +7 495 276 0211  
Email: [svan@elvis.ru](mailto:svan@elvis.ru)