

Operational Security Capabilities for
IP Network Infrastructure (opsec)
Internet-Draft
Intended status: Informational
Expires: April 18, 2013

F. Gont
Huawei Technologies
October 15, 2012

Virtual Private Network (VPN) traffic leakages in dual-stack hosts/
networks
draft-gont-opsec-vpn-leakages-00

Abstract

The subtle way in which the IPv6 and IPv4 protocols co-exist in typical networks, together with the lack of proper IPv6 support in popular Virtual Private Network (VPN) products, may inadvertently result in VPN traffic leaks. That is, traffic meant to be transferred over a VPN connection may leak out of such connection and be transferred in the clear on the local network. This document discusses some scenarios in which such VPN leakages may occur, either as a side effect of enabling IPv6 on a local network, or as a result of a deliberate attack from a local attacker. Additionally, it discusses possible mitigations for the aforementioned issue.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

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 April 18, 2013.

Copyright Notice

Copyright (c) 2012 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
2. IPv4 and IPv6 co-existence	4
3. Virtual Private Networks in IPv4/IPv6 dual-stack hosts/networks	5
4. VPN traffic-leakages in legitimate scenarios	6
5. VPN traffic-leakage attacks	7
6. Mitigations to VPN traffic-leakage vulnerabilities	8
7. IANA Considerations	9
8. Security Considerations	10
9. Acknowledgements	11
10. References	12
10.1. Normative References	12
10.2. Informative References	12
Author's Address	14

1. Introduction

It is a very common practice for employees working at remote locations to establish a VPN connection with their office or home office. This is typically done to gain access to some resources only available within the company's network, but also to secure the host's traffic against attackers that might be connected to the same remote location. In some scenarios, it is even assumed that employing a VPN connection makes the use of insecure protocols (e.g. that transfer sensitive information in the clear) acceptable, as the VPN provides security services (such as confidentiality) for all communications made over the VPN.

Many VPN products that are typically employed for the aforementioned VPN connections only support the IPv4 protocol: that is, they perform the necessary actions such that IPv4 traffic is sent over the VPN connection, but they do nothing to secure IPv6 traffic originated from (or being received at) the host employing the VPN client. However, the hosts themselves are typically dual-stacked: they support (and enable by default) both IPv4 and IPv6 (even if such IPv6 connectivity is simply "dormant" when they connect to IPv4-only networks). When the IPv6 connectivity of such hosts is enabled, they may end up employing an IPv6-unaware VPN client in a dual-stack network. This may have "unexpected" consequences, as explained below.

The subtle way in which the IPv4 and IPv6 protocols interact and co-exist in dual-stacked networks might, either inadvertently or as a result of a deliberate attack, result in VPN traffic leakages -- that is, traffic meant to be transferred over a VPN connection could leak out of the VPN connection and be transmitted in the clear on the local network, without employing the VPN services at all.

Section 2 provides some background about IPv6 and IPv4 co-existence, summarizing how IPv4 and IPv4 interact on a typical dual-stacked network. Section 3 describes the underlying problem that leads to the aforementioned VPN traffic leakages. Section 4 describes legitimate scenarios in which such traffic leakages might occur, while Section 5 describes how VPN traffic leakages can be triggered by deliberate attacks.

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. IPv4 and IPv6 co-existence

The co-existence of the IPv4 and IPv6 protocols has a number of interesting and subtle aspects that may have "surprising" consequences. While IPv6 is not backwards-compatible with IPv4, the two protocols are "glued" together by the Domain Name System (DNS).

For example, consider a site (say, `www.example.com`) that has both IPv4 and IPv6 support. The corresponding domain name (`www.example.com`, in our case) will contain both A and AAAA DNS resource records (RRs). Each A record will contain one IPv4 address, while each AAAA record will contain one IPv6 address -- and there might be more than one instance of each of these record types. Thus, when a dual-stacked client application means to communicate with the aforementioned site, it can request both A and AAAA records, and use any of the available addresses. The preferred address family (IPv4 or IPv6) and the specific address that will be used (assuming more than one address of each family is available) varies from one protocol implementation to another, with many host implementations preferring IPv6 addresses over IPv4 addresses.

[RFC6724] specifies an algorithm for selecting a destination address from a list of IPv6 and IPv4 addresses. [RFC6555] discusses the challenge of selecting the most appropriate destination address, along with a proposed implementation approach that mitigates connection-establishment delays.

This "co-existence" between IPv6 and IPv4 means that, when a dual-stacked client means to communicate with some other system, the availability of A and AAAA DNS resource records will typically affect which protocol is employed to communicate with that system.

3. Virtual Private Networks in IPv4/IPv6 dual-stack hosts/networks

Many Virtual Private Network (VPN) implementations do not support the IPv6 protocol -- or, what is worse, they completely ignore IPv6. This typically means that, when establishing a VPN connection, the VPN software takes care of the IPv4 connectivity by, e.g. inserting an IPv4 default route that causes all IPv4 traffic to be sent over the VPN connection (as opposed to sending the traffic in the clear, employing the local router). However, if IPv6 is not supported (or completely ignored), any packets destined to an IPv6 address will be sent in the clear using the local IPv6 router. That is, the VPN software will do nothing about the IPv6 traffic.

The underlying problem here is that while IPv4 and IPv6 are two different protocols incompatible with each other, the two protocols are glued together by the Domain Name System. Therefore, for dual-stacked systems, it is not possible to secure secure the communication with another system without securing both protocols (IPv6 and IPv4).

4. VPN traffic-leakages in legitimate scenarios

Consider a dual-stacked host that employs IPv4-only VPN software to establish a VPN connection with a VPN server, and that the host now attaches to a dual-stacked network (that provides both IPv6 and IPv4 connectivity). If some application on the client needs to communicate with a dual-stacked destination, the client will typically query both A and AAAA DNS resource records. Since the host will have both IPv4 and IPv6 connectivity, and the intended destination will have both A and AAAA DNS resource records, one of the possible outcomes is that the host will employ IPv6 to communicate with the aforementioned system. Since the VPN software does not support IPv6, the IPv6 traffic will not employ the VPN connection, and will be sent in the clear on the local network.

This could inadvertently expose sensitive traffic that was assumed to be secured by the VPN software. In this particular scenario, the resulting VPN traffic leakage is a side-effect of employing IPv6-unaware software in a dual-stacked host/network.

5. VPN traffic-leakage attacks

A local attacker could deliberately trigger IPv6 connectivity on the victim host by sending forged ICMPv6 Router Advertisement messages. Such packets could be sent by employing standard software such as rtadvd [RTADVd], or by employing packet-crafting tools such as the [SI6-Toolkit] or THC-IPv6 [THC-IPv6]. Once IPv6 connectivity has been enabled, communications with dual-stacked systems could result in VPN traffic leakages, as previously mentioned.

While this attack may be useful enough (due to the increasing number of IPv6-enabled sites), it will only lead to traffic leakages when the destination system is dual-stacked. However, it is usually trivial for an attacker to trigger such VPN leakages for any destination systems: an attacker could simply advertise himself as the local recursive DNS server by sending forged Router Advertisement messages that include the corresponding RDNSS option, and then perform a DNS spoofing attack such that he can become a "Man in the Middle" and intercept the corresponding traffic. As with the previous attack scenario, packet-crafting tools such as [SI6-Toolkit] and [THC-IPv6] can readily perform this attack.

Some systems are known to prefer IPv6-based recursive DNS servers over IPv4-based ones, and hence the "malicious" recursive DNS servers would be preferred over the legitimate ones advertised by the VPN server.

6. Mitigations to VPN traffic-leakage vulnerabilities

There are a number of possible mitigations for the VPN traffic-leakage vulnerability discussed in this document.

If the VPN client is configured by administrative decision to redirect all traffic for IPv4 to the VPN, it should:

1. If IPv6 is not supported, disable IPv6 support in all network interfaces

For IPv6-unaware VPN clients, the most simple mitigation (although not necessarily the most desirable one) would be to disable IPv6 support in all network interface cards when a VPN connection is meant to be employed. Thus, applications on the host running the VPN client software will have no other option than to employ IPv4, and hence they will simply not even try to send/process IPv6 traffic.

2. If IPv6 is supported, ensure that all IPv6 traffic is also sent via the VPN

If the VPN client is configured to only send a subset of IPv4 networks to the VPN tunnel (split-tunnel mode), and the VPN client does not support IPv6, it should disable IPv6 as well. If it supports IPv6, it is the administrators responsibility to ensure that the correct corresponding sets of IPv4 and IPv6 networks get routed into the VPN tunnel.

Additionally, VPN clients that support IPv6 should mitigate all ND-based attacks that may introduce new entries in the routing table, such attacks based on forged RA messages containing more specific routes, forged ICMPv6 Redirect messages, etc.

A network may prevent local attackers from successfully performing the aforementioned attacks against other local hosts by implementing First-Hop Security solutions such as Router Advertisement Guard (RA-Guard) [RFC6105] and DHCPv6-Shield [I-D.gont-opsec-dhcpv6-shield]. However, for obvious reasons, a host cannot and should not rely on this type of mitigations when connecting to an open network (cybercafe, etc.).

Besides, popular implementations of RA-Guard are known to be vulnerable to evasion attacks
[I-D.ietf-v6ops-ra-guard-implementation].

7. IANA Considerations

This document has no actions for IANA.

8. Security Considerations

This document discusses how traffic meant to be transferred over a VPN connection can leak out of the VPN, and hence appear in the clear on the local network. This is the result of employing IPv6-unaware VPN client software on dual-stacked hosts.

Possible ways to mitigate this problem include fixing the VPN client software, or disabling IPv6 connectivity on all network interfaces when the previous option is not feasible.

9. Acknowledgements

The author would like to thank (in alphabetical order) Gert Doering and Tor Houghton, who providing comments on earlier versions of this document.

This documents has benefited from the input of Cameron Byrne, Gert Doering, Seth Hall, Tor Houghton, Alastair Johnson, Henrik Lund Kramshoj, and Jim Small, while discussing this topic on the ipv6hackers mailing-list [IPv6-Hackers]. It has also benefited from discussions with Andrew Yourtchenko on the opsec wg mailing-list [OPSEC-LIST].

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.
- [RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.

10.2. Informative References

- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, February 2011.
- [I-D.ietf-v6ops-ra-guard-implementation]
Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)",
draft-ietf-v6ops-ra-guard-implementation-04 (work in progress), May 2012.
- [I-D.gont-opsec-dhcpv6-shield]
Gont, F., "DHCPv6-Shield: Protecting Against Rogue DHCPv6 Servers", draft-gont-opsec-dhcpv6-shield-00 (work in progress), May 2012.
- [IPv6-Hackers]
"IPv6 Hackers mailing-list",
<http://lists.si6networks.com/listinfo/ipv6hackers/>.
- [OPSEC-LIST]
"OPSEC WG mailing-list",
<https://www.ietf.org/mailman/listinfo/opsec>.
- [SI6-Toolkit]
"SI6 Networks' IPv6 toolkit",

<<http://www.sisnetworks.com/tools/ipv6toolkit>>.

[THC-IPv6]

"The Hacker's Choice IPv6 Attack Toolkit",
<<http://www.thc.org/thc-ipv6/>>.

[RTADVD]

"rtadvd(8) manual page", <<http://www.freebsd.org/cgi/man.cgi?query=rtadvd&sektion=8>>.

Author's Address

Fernando Gont
Huawei Technologies
Evaristo Carriego 2644
Haedo, Provincia de Buenos Aires 1706
Argentina

Phone: +54 11 4650 8472
Email: fgont@si6networks.com
URI: <http://www.si6networks.com>

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: October 14, 2013

D. Harkins, Ed.
Aruba Networks
April 12, 2013

The (Real) Internet Key Exchange
draft-harkins-ikev3-01

Abstract

The current version (v2) of the Internet Key Exchange failed to address many of the shortcomings of the original version (v1). This memo defines a new version (v3) of the Internet Key Exchange that attempts to do so.

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 14, 2013.

Copyright Notice

Copyright (c) 2013 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	4
1.1. Requirements Language	4
2. Characteristics of Version 3 of IKE	4
3. Differences from Previous Versions of IKE	4
3.1. Identity Confidentiality	5
3.2. Single IKE SA, No Lifetime	5
3.3. Not A Request/Response Exchange	5
3.4. State Machine Definition	6
4. Cryptographic Tools	6
4.1. Authenticated Encryption	6
4.2. Hash Function	7
4.3. Discrete Logarithm Cryptography	7
4.4. Key Derivation Function	8
5. Authentication and Key Establishment	9
5.1. Public Key Authentication	9
5.1.1. KE Payload with Public Key Authentication	9
5.1.2. AU Payload with Public Key Authentication	10
5.2. PSK Authentication	10
5.2.1. Hunting and Pecking with ECP Groups	11
5.2.2. Hunting and Pecking with MODP Groups	12
5.2.3. KE Payload with PSK Authentication	13
5.2.4. AU Payload with PSK Authentication	14
5.3. Deriving Shared Secrets	15
6. The Internet Key Exchange Protocol	15
6.1. Message Flow	16
6.1.1. Init Messages	16
6.1.1.1. Construction of Init Messages	16
6.1.1.2. Processing of Init Messages	17
6.1.2. Auth Messages	18
6.1.2.1. Construction of Auth Messages	18
6.1.2.2. Processing of Auth Messages	19
6.2. IPsec Security Associations	21
6.3. State Machine	21
6.3.1. Parent Process	22
6.3.2. Components of State Machine	23
6.3.3. States	24
6.3.3.1. Nothing State	24
6.3.3.2. Initiation State	25
6.3.3.3. Reception State	26
6.3.3.4. Done State	27
6.3.4. Cleaning Up Protocol Instances	28
6.4. IKEv3 Payload Formats	28
6.4.1. IKE header	28
6.4.2. Generic IKE payload header	30
6.4.3. IKE Attributes payload	31
6.4.4. Identity Payload	33

6.4.5.	Nonce Payload	34
6.4.6.	Key Exchange Payload	34
6.4.7.	Certificate Payload	35
6.4.8.	Certificate Request Payload	36
6.4.9.	Authentication Payload	36
6.4.10.	Address Indication Payload	37
6.4.11.	Traffic Selector Payload	37
6.4.12.	Security Association Payload	39
6.4.13.	Vendor Indication Payload	40
7.	Acknowledgements	41
8.	IANA Considerations	41
9.	Security Considerations	41
10.	References	41
10.1.	Normative References	41
10.2.	Informative References	42
	Author's Address	43

1. Introduction

The Internet Key Exchange was first defined in [RFC2409] to generate security associations for the IPsec protocols. That specification was poorly written and suffered from too many options, many of which were unneeded and went unused. In short, it was confusing and complicated. An effort was made to come up with a simpler and less confusing version, and that resulted in [RFC4306], so-called IKEv2 ([RFC2409] was then dubbed IKEv1). While it was arguably simpler and less confusing, IKEv2 failed to achieve its goal. It went through an extensive clarification process that produced [RFC4718] and development of a replacement specification for IKEv2, in [RFC5996]. While [RFC5996] is definitely a cleaner protocol than [RFC2409] it still has too many options and is too complicated, and confusing.

This memo defines an IKEv3 in an effort to have a simpler, more easy to implement protocol, that that has a high probability of achieving interoperability while retaining security and utility.

1.1. Requirements Language

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

2. Characteristics of Version 3 of IKE

Version 3 of the Internet Key Exchange is a simple peer-to-peer protocol in which each side sends and receives two messages. It performs mutual authentication, derives a shared, secret and authenticated key, and negotiates parameters for IPsec security associations (SAs) to protect transient data between the peers.

Either side can initiate the exchange to the other and both sides can initiate simultaneously (hence the claim of a true "peer-to-peer" exchange). This is advantageous for certain smart devices-- aka "the Internet of things"-- or sensor network deployments where there is no strict roles of client or server, initiator or responder.

In an effort to keep the definition of the Internet Key Exchange as simple as possible, negotiation of the terms of its operation-- e.g. encryption algorithm, hash algorithm-- is kept to a minimum.

3. Differences from Previous Versions of IKE

3.1. Identity Confidentiality

IKEv3 does not provide identity confidentiality. This is a tradeoff made to increase simplicity in specification and, more importantly, implementation at the cost of a feature whose benefit is somewhat dubious.

While security on the Internet is a large issue (one that IKEv3 addresses) the problems associated with exposure of the identities of two peers that are engaging in secure communication is not.

If identity hiding critical for a particular deployment, IKEv3 supports obfuscation of identity using an ID blob which has meaning to the two peers of the exchange but has no meaning to any third party that may observe it.

3.2. Single IKE SA, No Lifetime

IKEv3 can handle the situation where both sides initiate to each other without resorting to carrying on two conversations and ending up with two IKE Security Associations.

The IKEv3 SA is also short-lived. Its purpose is to create SAs for IPsec and once it has done that the state that governed a particular protocol run can go away. There is no notion of a long-lived IKE SA.

There is no SA lifetime necessary for IKEv3 to negotiate. This also has the benefit of doing away with the Delete payloads and their corresponding complexity as well as the complexity associated with rekeying of SAs.

There is no need for "initial contact" notification or the need to negotiate, or rekey, multiple IKE SAs.

3.3. Not A Request/Response Exchange

[RFC2409] and [RFC5996] are both Request/Response protocols. There are defined roles-- one side is an "Initiator" and the other is a "Responder"-- and one side makes Requests and the other Responds to those requests.

In IKEv3 there are no roles involved-- no clients and servers, no Initiators and Responders-- just two peers who perform identical behavior. Since either side can initiate and both sides can initiate simultaneously, there is no need to deal with "Exchange Collisions". All the protocol specification complexity to address the problems that occur due to role-based protocol definition goes away.

Many of the IKEv1 and IKEv2 use cases involved strict roles and IKEv3 can support them because the state machine (see Section 6.3) can handle the case where one side initiates and the other responds just as easily as it can handle the case where both sides initiate simultaneously.

3.4. State Machine Definition

IKEv3 defines a very simple state machine that each side runs through to implement the protocol. Accurate compliance with the state machine ensures interoperability.

The state machine allows the protocol definition to be entirely from the point of view of the implementation, making protocol implementation much easier. The protocol is defined in terms of actions causing events which result in a deterministic advancement of state until the protocol is finished. The state machine definition assures that each side either completes the protocol or neither side completes the protocol.

Previous versions of IKE lacked a state machine definition and it showed. IKEv1 achieved interoperability through implementor "bake-offs" and not through a coherent specification. IKEv2 has gone through forty (40) revisions, in total, in an effort to clarify and straighten out ambiguous and confusing text and remains to this day a complicated, ambiguous and confusing specification.

4. Cryptographic Tools

IKEv3 makes use of certain cryptographic primitives to achieve its goals of key generation, mutual authentication, and security. Each of the following subsections indicate negotiable components of the IKE Security Association that are used during the protocol run. Different protocol runs can negotiate different components and the components to use in a particular run of the protocol are established by exchanging payloads in the Init message that describe the attributes of the IKE Security Association (see Section 6.4).

4.1. Authenticated Encryption

Authenticated encryption is employed by IKEv3 to protect the payloads that set up IPsec Security Associations as well as any vendor-specific payloads that are added to the final two messages of the protocol. It is therefore a negotiable component. The privacy algorithm negotiated MUST be a cipher mode, or construction of cipher mode plus integrity check, that provides authenticated encryption.

AES in SIV mode as defined in [RFC5297] is used in IKEv3 to accomplish this goal. SIV supports authenticated encryption with associated data (which is authenticated but not encrypted) and does not require complex management of a unique counter space to ensure security. It is simple, secure and robust. A perfect fit for IKEv3.

4.2. Hash Function

A hash function takes a arbitrary-sized input and deterministically produces a fixed sized output, called a digest. It is also a one-way function: it is very easy to produce a digest but computationally infeasible to reconstruct the arbitrary-sized input given a particular digest.

IKEv3 uses a hash function, in [RFC2104] mode for key derivation and key confirmation. IKEv3 also uses a hash function to construct a random function, $H()$:

$$H(x) = \text{HMAC-Hash}(0^n, x)$$

where Hash is the agreed-upon hash function and " 0^n " signifies a key of all zeros whose length equals the digest size of the hash function.

IKEv3 defines SHA-256 and SHA-512 (as defined in [RFC4634]) for use as hash functions.

4.3. Discrete Logarithm Cryptography

The Internet Key Exchange uses discrete logarithm cryptography. Each party to the protocol derives ephemeral public and private key pairs with respect to a particular domain parameter set, called a "group". The group can be based on either finite field cryptography (modular exponentiation, or MODP, groups) or elliptic curve cryptography (ECP groups).

In this memo, elements in a group are denoted in upper case and scalar values are in lower case-- element X and scalar x .

Groups are identified in messages using a convenient registry maintained by IANA (see Section 8). Each group's domain parameter set contains the following:

- o p - a prime number defining a finite field
- o G - a generator, a base element forming a group

- o q - a prime number indicating the order of the group defined by G

ECP groups additionally define "a" and "b" which are components of the equation of the elliptic curve-- $y^2 = x^3 + ax + b$. Some MODP groups are based on safe primes and do not have a specific order defined. For these groups only, the order, q , SHALL be $(p-1)/2$.

For each group, the following operations are defined:

- o "scalar operation" -- takes a scalar and an element in the group to produce another element in the group-- $Z = \text{scalar-op}(x, Y)$. For ECP groups this is multiplication of the element by the scalar; for MODP groups this is exponentiation of the element to the scalar.
- o "element operation" -- takes two elements in the group to produce a third element in the group-- $Z = \text{element-op}(X, Y)$. For ECP groups this is point addition; for MODP groups this is modular multiplication.
- o "inverse operation" -- takes an element in the group and returns another element in the group such that the element operation on the two produces the identity element of the group-- $Y = \text{inverse}(X)$.

ECP: $\text{element-op}(Y, \text{inverse}(Y)) = \text{"point at infinity"}$

MODP: $\text{element-op}(Y, \text{inverse}(Y)) = 1$

In addition, ECP groups require a mapping function, $r = F(R)$, to convert a group element to an integer. The mapping function used in this memo returns the x-coordinate of the point it is passed. MODP groups do not need a mapping function as group elements in MODP groups can be directly represented as integers. For the purpose of protocol definition, the function $F()$ when used with MODP groups will be the identity function-- i.e. $i = F(i)$.

4.4. Key Derivation Function

IKEv3 uses a key derivation function, $KDF()$, to stretch a random key to an indeterminate length and bind some arbitrary data to the stretched key.

For ease of programming, IKEv3 uses the $\text{prf}+()$ function from [RFC5996] which, in turn, was derived from the $\text{prf}()$ function in [RFC2409], as its $KDF()$.

The pseudo-random function at the core of the $\text{prf}+()$ construct is the

agreed-upon hash function in HMAC ([RFC2104]) mode. When KDF() is called for in this memo, it is prf+() from [RFC5996] using HMAC-Hash where hash is the agreed-upon hash algorithm.

5. Authentication and Key Establishment

The goal of any pairwise authenticated key exchange is key establishment and mutual authentication. The IKEv3 protocol achieves these goals. The particular method of key establishment is tied to the authentication method which is tied to the type of credential used for authentication-- a (certified) public key or a PSK.

5.1. Public Key Authentication

Public key authentication uses a Diffie-Hellman key exchange for key establishment and digital signatures by a private key whose public analog is trusted by the peer.

5.1.1. KE Payload with Public Key Authentication

The KE payload is used to present a Diffie-Hellman public value to the peer. Each peer generates a random number between one (1) and the order of the group, q , exclusive. This represents the peer's private value, $priv$. The peer then performs the group's scalar operation (see Section 4.3) with the group's generator to produce the public value, Pub :

$$Pub = \text{scalar-op}(priv, G)$$

The public value is encoded in to the body of the KE Payload (see Section 6.4) according to the integer to octet string conversion technique from [RFC6090].

The Diffie-Hellman key exchange is completed when both sides have finished sending and receiving an Init message. Each side generates the same shared secret, $secret$, by applying the mapping function, $F()$ (see Section 4.3), to the result of the group's scalar operation with the entities private value, $priv$, and the peer's public key, $PubPeer$:

$$secret = F(\text{scalar-op}(priv, PubPeer))$$

The secret is then used to generate three additional keys, the authenticated encryption key, the confirmation key, and a key-derivation key. (see Section 5.3).

5.1.2. AU Payload with Public Key Authentication

The AU payload contains a digital signature of the confirmation key and both peers' Init messages concatenated together, transmitter's Init message first. For example, assuming Alice sent the message InitA and Bob sent the message InitB, Alice's digital signature would be "sig" where:

$$\text{sig} = \text{Sign-Alice}(\text{cKEY} \mid \text{InitA} \mid \text{InitB})$$

where "|" signifies concatenation, and Sign-Alice() indicates a digital signature of that data passed to it using the public key of Alice. The portions of the Init messages that are covered by the digital signature consist of the IKE header (inclusive) to the end of the payload.

Bob would similarly send:

$$\text{sig} = \text{Sign-Bob}(\text{cKEY} \mid \text{InitB} \mid \text{InitA})$$

since Bob was the transmitter of InitB.

To maintain a consistent level of security for IKEv3, the hash algorithm used to generate the digital signature SHALL be the one negotiated in the IKE Security Association that is used for other hashing purposes in IKEv3. The body of the AU payload (see Section 6.4) SHALL consist of the digital signature as a bitstring.

5.2. PSK Authentication

PSK authentication uses the "dragonfly" key exchange to both generate a shared, and secret, key and to mutually authenticate the peers to each other. Each side proves knowledge of the PSK in a manner that is resistant to dictionary attack.

Each side proves possession of a single PSK (or password), there is no notion of a "client's password" and a "server's password"; there is just the one. This single PSK MUST be provisioned on the two peers prior to beginning the IKEv3 exchange. Since there is only one PSK it SHOULD have only one name, which is provisioned along with the PSK. It is this name that is used in the ID payload when initiating the dragonfly exchange to the peer. Note: it may make sense in certain client/server deployments to have a proper client username assigned to the password, in which case the server proving possession of the client's password-- identified by username-- authenticates it to the client.

When PSK authentication is chosen for a particular run of the

protocol, the KE payload contains each peer's "commit" contribution to the dragonfly exchange and the AU payload contains a keyed message authentication code binding the secret key to both peers' Init messages concatenated together.

Prior to beginning the "dragonfly" exchange, both peers MUST agree upon a secret element in the chosen group. A secret seed is generated and that seed is used in a group-specific hunting-and-pecking process-- one process for MODP groups and another for ECP groups. First, an 8-bit counter is set to one (1) and a secret base is computed using the negotiated one-way function with the secret PSK, and the counter:

```
base = H(PSK | counter)
```

The base is then stretched using the key derivation function from Section 4.4 to the length of the prime from the group's domain parameter set:

```
seed = KDF(base, "IKE PSK Hunting and Pecking")
```

The seed is then passed to the group-specific hunting and pecking technique.

5.2.1. Hunting and Pecking with ECP Groups

The ECP specific hunting and pecking technique entails looping until a valid point on the elliptic curve has been found. The seed is used as the x-coordinate with the equation of the curve to solve for a y-coordinate. If there is no solution, the counter is incremented, a new base and new seed are generated and the hunting and pecking continues. If there is a solution an ambiguity exists because two values for the y-coordinate would be valid. The low-order bit of the base is used to unambiguously determine the y-coordinate and the resulting (x,y) pair becomes the secret generator for the dragonfly exchange, SKE.

Algorithmically, the process looks like this:

```
found = 0
counter = 1
do {
  base = H(psk | counter)
  seed = KDF(seed, "IKE PSK Hunting And Pecking")
  if (seed < p)
  then
    x = seed
    if ( (x^3 + ax + b) is a quadratic residue mod p )
    then
      y = sqrt(x^3 + ax + b)
      if (LSB(y) == LSB(base))
      then
        SKE = (x,y)
      else
        SKE = (x, p-y)
      fi
    fi
    found = 1
  fi
  counter = counter + 1
} while (found == 0)
```

Figure 1: Fixing SKE for ECP Groups

5.2.2. Hunting and Pecking with MODP Groups

The MODP specific hunting and pecking technique entails finding a random element which, when used as a generator, will create a group with the same order as the group created by the generator from the domain parameter set. The secret generator is found by exponentiating the seed to the value $((p-1)/q)$, where p is the prime and q is the order from the domain parameter set. If that value is greater than one (1) it becomes SKE, otherwise the counter is incremented, a new base and seed are generated, and the hunting and pecking continues.

Algorithmically, the process looks like this:

```
found = 0
counter = 1
do {
  base = H(psk | counter)
  seed = KDF(base, "IKE SKE Hunting And Pecking")
  if (seed < p)
  then
    SKE = seed ^ ((p-1)/r) mod p
    if (SKE > 1)
    then
      found = 1
    fi
  fi
  counter = counter + 1
} while (found == 0)
```

Figure 2: Fixing SKE for MODP Groups

5.2.3. KE Payload with PSK Authentication

Once SKE has been determined, the peer randomly chooses two numbers between one and the order of the group, q , exclusively. These represent a private value and a mask. The peer then generates a scalar and an element using private, mask, and SKE:

```
scalar = (private + mask) mod q

Element = inverse(scalar-op(mask, SKE))
```

The scalar and element, respectively, are encoded into the KE payload by using the integer to octet string conversion technique from [RFC6090]. Octet strings are pre-pended with zero (0), if necessary, to achieve the required resulting length. Since the length of each component of the KE payload is implicitly known, the scalar and element can be extracted from the KE payload for processing.

The scalar is the same length as the order of the group. It is converted into an octet string and then the octet string is inserted into the body of the KE Payload.

If the selected group is MODP, the element can be treated directly as an integer and converted into an octet string. Its length is the same as the length of the prime of the group. The converted octet string is appended to the octet string representation of the scalar.

If the selected group is ECP, the element is an (x,y) pair and each coordinate is separately converted into an octet string, each of which is the same length as the prime of the group. The octet string

representation of the x-coordinate SHALL be appended to the scalar and the y-coordinate SHALL be appended to the x-coordinate.

The dragonfly key handshake is completed when both sides have finished sending and receiving an Init message. Each side generates the same shared secret, secret, by performing the following computation:

```
secret = F(scalar-op(private,
                      element-op(PeerElement,
                                scalar-op(peerscalar, SKE))))
```

where peerscalar and PeerElement are scalar and element from the peer's KE payload taken out of a received Init message. The secret is then used to generate three additional keys, the authenticated encryption key, the confirmation key, and a key-derivation key. (see Section 5.3).

5.2.4. AU Payload with PSK Authentication

The AU payload contains a keyed message authentication code which proves knowledge of the derived secret, and therefore knowledge of the PSK, and binds the two Init messages to the authenticated state.

Each side produces an authenticating message authentication code, mac, by invoking the HMAC version of the negotiated hash function and passing the confirmation key, cKEY, as the key and the concatenation of both peers' Init messages concatenated together, transmitter's Init message first. For example, assuming Alice sent the message InitA and Bob sent the message InitB, Alice's message authentication code would be "mac" where:

```
mac = HMAC-Hash(cKEY, InitA | InitB)
```

where "|" signifies concatenation and HMAC-Hash is the [RFC2104] instantiation of the negotiated hash algorithm, Hash. The portions of the Init messages passed HMAC-Hash consist of the IKE header (inclusive) to the end of the payload.

Bob would similarly send:

```
mac = HMAC-Hash(cKEY, InitB | InitA)
```

since Bob was the transmitter of InitB.

5.3. Deriving Shared Secrets

Upon successful completion of key establishment, IKEv3 produces three keys, an authenticated encryption key, aeKEY, to protect the Auth Messages, a confirmation key, cKEY, and a derivation key, dKEY, used to derive (a) shared secret(s) when constructing IPsec Security Associations (see Section 6.2).

The length of aeKEY depends on the authenticated encryption mode used and the length of cKEY and dKEY SHALL be the length of the digest of negotiated hash function. The keys are derived by passing the two nonces, appended to each other with the lexicographically larger nonce being first, as the key and secret from the authenticated key exchange concatenated with the label "IKEv3 Key Derivation" as the data:

$$\text{aeKEY} \mid \text{cKEY} \mid \text{dKEY} = \text{KDF}(\text{max}(\text{Na}, \text{Nb}) \mid \text{min}(\text{Na}, \text{Nb}), \\ \text{secret} \mid \text{"IKEv3 Key Derivation"})$$

where Na and Nb are the two nonces from the exchange (the transmitter is irrelevant in this peer-to-peer protocol), max() returns the lexicographically larger of the two parameters passed, and min() returns the lexicographically smaller of the two parameters passed.

The key aeKEY SHALL be used to protect the exchange of Auth Messages, the same key is used in both directions.

6. The Internet Key Exchange Protocol

The Internet Key Exchange (IKE) authenticates two peers to each other and derives security associations for use by IPsec. The credentials supported by IKE are PSKs and certificates.

IKE supports varying degrees of security by supporting various domain parameter groups, encryption algorithms, and hash algorithms.

IKEv3 supports detection of NATs between two peers through the exchange of source and destination indicators. When (a) NAT(s) is (are) present between the peers the source and/or destination addresses and/or ports will be modified and differ from those in the indicators. When (a) NATS(s) is (are) detected, UDP encapsulation of ESP traffic as defined by [RFC3948] is required. Note that IKEv3 is not required to use port 4500 in the presense of (a) NAT(s).

6.1. Message Flow

In the IKEv3 protocol each peer sends and receives an Init message and an Auth message. The Init message negotiates the type of authentication to be used between the peers, identifies the peers to each other, exchanges random nonces, and exchanges the components of a cryptographic key exchange. The Auth message authenticates the peer to the other peer and establishes IPsec security associations.

Messages are comprised of an IKE header followed by one or more payloads. The on-the-wire format of the IKE header and all payloads defined for use in IKE are in Section 6.4.

As is typical in these sorts of memos, the participants in the protocol are Alice and Bob. The exchange of Init and Auth messages between Alice and Bob look like this:

Alice		Bob
-----		----
Init: hdr, IAa, IDa, NOa,		hdr, IAb, IDb, NOb,
KEa [, CRa]	----> <-----	KEb [, CRb]
Auth: hdr, { [CEa,] AUa, AIs,		hdr, { [CEb,] Aub, AIs,
AId, SAa, TSs, TSd }	----> <-----	AId, SAb, TSs, TSd }

Where { x } indicates the authenticated encryption of payload x using the mode agreed upon in the exchange of IA payloads.

6.1.1. Init Messages

6.1.1.1. Construction of Init Messages

The IKEv3 header contains two message identifiers called SPIs, one chosen by the transmitter of the message and one chosen by the (intended) recipient of the message. The local SPI from the IKE security association is copied into the transmitter SPI field. If a peer SPI exists in the IKE security association, it is copied into the recipient SPI field. If there is no peer SPI in the IKE security association, the recipient SPI field remains all zero.

The first payload in an Init message MUST be an IA payload which indicates the terms by which the IKE protocol will be run (see Section 4). If this Init message is being constructed in response to receipt of an accepted Init message then the attributes from the received, and accepted, Init message MUST be copied into the Init message being constructed. If the Init message is being constructed due to an indication to the IKEv3 protocol to establish IPsec SAs with a remote peer (see Section 6.3) then the attributes MUST reflect the policy that accompanied that indication.

The next payload after the IA payload MUST be the ID payload which indicates the identity of the peer sending the Init message (see Section 3.1 for a discussion on identity confidentiality). The next payload MUST be a NO payload which contributes additional randomness to the exchange. The next payload MUST be a KE payload. The particular construction of the body of the KE payload depends on the authentication method being used for this run of the protocol (see Section 5). Finally, a CR payload MAY be added if the authentication method is public key authentication and the sender of the Init message believes that it needs the peer's public key.

Vendor specific payloads MAY be appended to an Init message to convey some additional semantics governing the Init message.

6.1.1.2. Processing of Init Messages

The first step of processing an Init Message is to record the peer's SPI by taking it out of the transmitter SPI field of the IKEv3 header and storing it in the IKE security association as the peer's SPI.

Next, the attributes that govern the IKEv3 protocol are checked. If the recipient SPI field is not all zeros (0) then the attributes in the received Init message MUST be identical to the attributes that have already been sent to the peer. If they are not, processing indicates a failure and stops. If the recipient SPI field is all zeros (0) and a message has not been sent to the peer then the attributes are checked for acceptability. If they are not acceptable processing SHALL indicate a failure and stop. If they are acceptable, then processing continues. Otherwise, if the recipient SPI field is all zeros (0) and a message has already been sent to the peer then there are three possible cases:

1. The attributes are identical to the attributes sent to the peer, so processing continues.
2. The attributes are not acceptable in which case the message is discarded and processing stops.
3. The attributes are acceptable but differ from those sent. In this case, a test is made to see which side drops its offer. Each side has sent its SPI to the other as the transmitter SPI in its Init message. If the low-order bit of those SPIs are identical then the transmitter of the larger SPI wins, if the low-order bit of those SPIs differ then the transmitter of the smaller SPI wins. The winner SHALL discard the message and the loser SHALL indicate a failure. In both cases, processing stops. Note: the loser will destroy all state associated with this conversation and the winner will retransmit, allowing the two to

synch up on the new, mutually acceptable attributes.

Finally, the nonce and key exchange data are extracted from the received Init message and processing finishes successfully. If the peer requested a certificate, that fact is noted to ensure that a certificate is included in the subsequent Auth message.

6.1.2. Auth Messages

6.1.2.1. Construction of Auth Messages

The Auth message MAY optionally contain a certificate payload (CE) with the public key of the transmitter and MUST contain, in the following order, an Auth payload (AU), a source Address Indication payload (AIs), a destination Address Indication payload (AId), a Security Association payload (SA), and two Traffic Selector payloads (TS). Optional vendor specific payload(s) MAY be appended to the message but MUST be the last payload(s) in the message.

The contents of AU payload are determined by the authentication method agreed-upon during the exchange of Init messages (see Section 5). The value of the Auth payload, from the point of view of the transmitter, SHALL be determined and copied into the data portion of the payload.

The AIs and AId payloads are constructed from the point of view of the transmitting peer. The source Address Indication payload, AIs, is the address and port being used as the source of the Auth message and the destination Address Indication payload, AId, is the address and port of the destination of the Auth message. The source Address Indication payload MUST precede the destination Address Indication payload.

The contents of the SA payload describe the transforms and options that will be represented in the IPsec SA after successful authentication. If this Auth message is being constructed in response to the receipt of an Auth message from the peer, the transforms in the Auth payload MUST be identical to those accepted when processing the peer's Auth message. If a locally-unique SPI with which to identify the security association for received IPsec packets has not yet been chosen, the transmitter chooses a SPI.

The TS payloads contains a description of the flows to protect using IPsec. There are two (2) TS payloads in each Auth message. The first describes the source of the flow and the second describes the sink of the flow, both from the perspective of the transmitter of the Auth message. If local Traffic Selector policy has been narrowed due to the processing of the peer's Auth message, then the narrowed

policy SHALL be reflected in the TS payloads.

Auth messages are sent after each side has both sent and received an Init message and completed the key establishment phase of the IKEv3 protocol. While the peers have not yet authenticated each other, they share a secret which can be used to secure the Auth messages. This is accomplished by using the authenticated encryption mode that was agreed-upon during the exchange of Init messages.

All the payloads of the Auth message are encrypted-- that is, everything after the IKEv3 header to the end of the message. The IKEv3 header, and the encrypted payloads are all authenticated.

When [RFC5297] is used for authenticated encryption the IKEv3 header from the transmitter's SPI (inclusive) to the Length (inclusive) is passed as associated data, and the data immediately following the IKEv3 header, from the generic IKEv3 header of the first payload (inclusive) to the end of the message is the data to encrypt. The ICV/SIV field of the IKEv3 header is not included in the associated data passed to AES-SIV.

The output of the [RFC5297] mode is ciphertext and a Synthetic Initialization Vector (SIV). The SIV SHALL be copied into the IKEv3 header and the ciphertext is appended to the IKEv3 header to form the complete Auth message.

6.1.2.2. Processing of Auth Messages

Auth messages are encrypted and authenticated so the first step in processing is to verify their integrity and to decrypt them. When [RFC5297] is used, the Synthetic Initialization Vector (SIV) is copied from the ICV/SIV field in the IKEv3 header. The IKEv3 header from the transmitter's SPI (inclusive) to the length (inclusive) is passed, along with the SIV to AES-SIV. If AES-SIV outputs FAIL the message is discarded and processing stops. If AES-SIV outputs plaintext, the plaintext will be the sequence of payloads that comprise the Auth message.

The first payload will be the Auth payload (AU). The contents of the AU payload are determined by the authentication method agreed-upon during the exchange of Init messages (see Section 5). The value of the the Auth message from the point of view of the transmitter (i.e. the peer) is calculated and compared to the value in the data portion of the AU payload. If they differ, the peer fails authentication, processing stops and a failure MUST be returned. If they are identical processing continues.

Next the Address Indication payloads are checked. If the source

address or port of the received message differ from the address or port in the source Address Indication payload, or if the destination address or port of the received message differ from the address or port in the destination Address Indication payload then a NAT is detected. Otherwise, a NAT is not detected.

The SA payload is next. The transforms in the SA payload are checked to determine whether they are acceptable according to local policy. If they are not the message is discarded and processing stops. If they are, then there are three possibilities:

- o An Auth message has not yet been sent to the peer, in which case processing continues; or,
- o An Auth message has been sent to the peer and the transforms are identical to those sent, in which case processing continues; or,
- o An Auth message has been sent to the peer and the transforms differ from those sent. In this case, a test is made to see which side's offer prevails. Each side has sent its IPsec SPI to the other in the SA payload. If the low-order bit of those SPIs are identical then the transmitter of the larger SPI prevails. If the low-order bit of those SPIs differ then the transmitter of the smaller SPI prevails. The transforms offered by the prevailing party SHALL be adopted by the party which does not prevail. Processing continues.

The Traffic Selectors in the TS payloads are next checked to determine whether they are acceptable according to local policy. If they are not acceptable, and no narrowing of the scope of the traffic flows is possible-- i.e. no intersection between the TS payloads and local policy-- the Auth message SHALL be discarded, processing stops.

If the Traffic Selectors are completely satisfactory and require no narrowing, then the Traffic Selectors are retained for creation of IPsec SAs and construction of an Auth message (if one has not already been sent).

If the Traffic Selectors are partially acceptable, and require narrowing, then the union of the local policy describing the flow and the Traffic Selectors describing the flow SHALL be retained for creation of IPsec SAs and construction of an Auth message (if one has not yet already been sent).

If an Auth message has not yet been sent, a locally-unique SPI SHALL be created to identify the IPsec SA for received IPsec-protected packets. This SPI MUST be retained for use when constructing the

Auth message response.

Upon completion of processing an Auth message, Two IPsec SAs MUST be instantiated (e.g. plumbed into the kernel) with the indicated transforms for the flow described in the (possibly narrowed) Traffic Selectors, one in each direction. The locally-unique SPI becomes the identifier to look up the SA for inbound IPsec packets and the peer's SPI (from its SA payload) becomes the identifier to look up the SA for outbound IPsec packets.

If a NAT was detected, the IPsec SAs MUST use UDP encapsulation for IPsec (see [RFC3948]). Since both sides know the original addresses and ports and the NATted addresses and ports, it is possible to obtain the required information to perform the necessary decapsulation procedures on received UDP-encapsulated IPsec packets.

6.2. IPsec Security Associations

The goal of the Internet Key Exchange is the creation of Security Associations (SAs) for [RFC4301]. SAs are established using Security Association (Section 6.4.12) and Traffic Selector (Section 6.4.11) payloads to negotiate the flow(s) to protect and the means to go about protecting it (them).

IKEv3 derives keys for IPsec SAs using the KDF (Section 4.4). The key derivation key, dKEY, established in Section 5.3 is used as the key and the label "IPsec Key Derivation" is used as the data:

```
key = KDF(dKEY, "IPsec Key Derivation")
```

The length of the key derived by KDF depends on the parameters of the IPsec SA and the key lengths used by the underlying primitives. If multiple, distinct, keying material is used-- for example, an ESP SA that performs encryption and integrity protection separately-- the key used for encryption MUST be taken from first and the key used for integrity protection MUST be taken from the remaining bits.

6.3. State Machine

The IKEv3 protocol is managed by a parent process that receives protocol events and IKEv3 packets and passes them on to instances of the IKEv3 state machine.

The state machine for IKE defines the behavior of a single run of the protocol. Each peer maintains a "protocol instance" for each remote peer that it is actively performing the protocol with that defines the current state of the protocol for that peer. The state machine guarantees that both sides will complete the protocol with each side

installing IPsec Security Associations or each side will fail to complete the protocol.

The state machine addresses the potential of dropped messages with a retransmission timer. This memo does not specify a period that state machines use when setting its retransmission timer.

6.3.1. Parent Process

The parent process of the IKEv3 state machine handles events from the IPsec SADB (e.g. an "acquire" message to create an IPsec security association) as well as receives incoming IKEv3 messages that it dispatches to state machine instances.

The parent process is also responsible for creation of state machine processes. The state of a state machine is stored in an IKEv3 security association so creation of a state machine process entails creation of a nascent IKEv3 security association, generating a unique and unpredictable local SPI, setting the peer address, and putting the state machine in NOTHING state.

When the parent process receives an event from the IPsec SADB to create an IPsec security association it first checks whether there is an existing IKEv3 state machine process with the indicated peer. If so, the parent process drops the event and waits for the process to complete. If there is no existing state machine process, the parent process creates a new state machine (see above) and sends the newly created state machine process a START event.

When the parent process receives an IKEv3 packet from a remote peer it first checks the recipient SPI field in the received packet.

If the recipient SPI field is all zeros, it indicates a peer that is initiating. If the IKEv3 message is an Auth frame, it SHALL be dropped as being meaningless (it is not possible to initiate IKEv3 with an Auth message). If the IKEv3 message is an Init message, the parent process checks whether there is an existing IKEv3 state machine process for the remote peer (the transmitter of the packet) that is in Initiation State. If so, the received message is passed to the state machine process with an INIT event. Otherwise, if there is no existing IKEv3 state machine process in Initiation State, the parent process creates an IKEv3 state machine process (see above) and passes the received message and an INIT event to it.

If the recipient SPI field is not all zeros, the parent process uses the recipient SPI to look up an existing IKEv3 process. If none exists, the packet SHALL be dropped. If the parent process succeeds in looking up an existing IKEv3 state machine process using the

recipient SPI, the message is passed to that state machine process with the appropriate event-- an INIT event for an Init message and an AUTH event for an Auth message.

6.3.2. Components of State Machine

The following states are part of the state machine:

- Nothing: a quiescent state in which nothing has happened
- Initiation: an Initiator has sent an Initiate message to a peer
- Reception: a Responder has sent an Initiate message to a peer
- Done: an Authenticate message has been sent to a peer

The following variables are used in the state machine:

- retrans: the number of retransmissions made (unsigned)
- thresh: the maximum number of retransmissions allowed (unsigned)
- committed: a counter on transmitted Auth messages (signed)
- reauth: a counter on received Auth messages (signed)

The following events are delivered to the state machine:

- START: an instruction to initiate IKE to a peer
- INIT: receipt of an Initiate message from a peer
- AUTH: receipt of an Authenticate message from a peer
- TM: expiry of the retransmission timer

The following actions are taken by the state machine:

- init: send an Initiate message to a peer
- auth: send an Authenticate message to a peer

The following timers are used by the state machine:

- tm: the retransmission timer
- fin: a deletion timer

6.3.3. States

The state machine for an IKEv3 process is show in Figure 3.

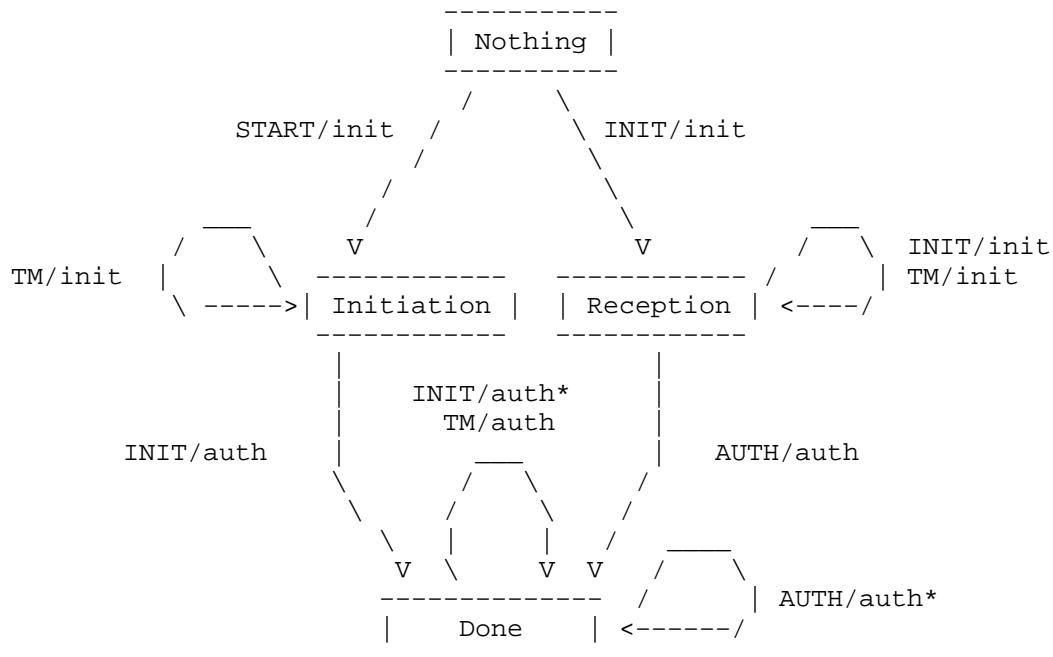


Figure 3: Protocol State Machine

6.3.3.1. Nothing State

Nothing state is the state in which an instance of the IKE state machine has just been created and has not received any events or performed any actions yet. Two events cause the state machine to exit Nothing state: a START event, and an INIT event.

When a state machine instance in Nothing state receives a START event the IKE peer initiates a connection to another peer. The information the IKE peer obtains as part of the START event is implementation specific but MUST indicate at a minimum the following:

- o IP address of peer
- o SPD information regarding the type of IPsec security association to form.

The method in which policy information regarding the type of authentication to propose, what group to use, etc., is out of scope of this memo. This information **MUST** be obtained but whether it is part of the START event indication or obtained as part of separate IKE configuration is irrelevant to the protocol.

The peer derives a session identifier, or SPI, to use as its transmitting SPI.

The peer retains the SPD information and SPI and constructs an Init message according Section 6.1.1.1 and transmits the message to the IP address of the peer. The state machine assigns the value zero (0) to the retrans counter, to the committed counter, and to the reauth counter. It sets the retransmission timer, and transitions to state Initiation.

When a state machine instance in Nothing state receives an INIT event, it signifies the reception of an Init message from a remote peer. The instance retains the IP address of the peer, extracts the transmitter's SPI from the message, and assigns the value zero (0) to the retrans counter, the committed counter and the reauth counter. It then processes the Init message according to Section 6.1.1.2. If processing of the Init message is successful, the instance generates an Init message for the peer according to Section 6.1.1.1, transmits the message to the peer, sets the retransmission timer and transitions to state Reception. If processing of the Init message is unsuccessful, the protocol instance remains in Nothing state and all state created as a result of receipt of the Init message **MUST** be deleted.

Note: a protocol instance that transition from Nothing state to Reception state has both received and sent an Initiate message. It **MAY** choose to finish the key exchange protocol and generate shared secret state according to the negotiated authentication method, or it may choose to delay such computation until it receives an AUTH event in Reception state.

6.3.3.2. Initiation State

In Initiation state a protocol instance has initiated the IKE protocol to a peer. An INIT event causes the instance to leave Initiation state, and a TM event causes it to remain in Initiation state.

When a protocol instance in Initiation state receives an INIT event, it signifies receipt of an Initiate message from the peer. The protocol instance first cancels the retransmission timer and then processes the Init message according to Section 6.1.1.2. If

processing indicates that the message was discarded, the protocol instance sets the TM timer and remains in Initiation state. If processing indicates a failure, the protocol instance deletes all state it has created or retained and transitions back to Nothing state. Otherwise, processing is successful and the protocol instance shall finish the key exchange protocol and generate shared secret state according to the negotiated authentication method. It then increments the committed counter and generates an Auth message for the peer according to Section 6.1.2.1, transmits the message to the peer, sets the retransmission timer and transitions to state Done.

When a protocol instance in Initiation state receives a TM event it indicates that the retransmission timer has expired. If the retrans counter is higher than the retransmission threshold it indicates failure of the protocol. In this case the protocol instance deletes all state it has created or retained and transitions back to Nothing state. If the retrans counter is not greater than the retransmission threshold, the Init message that was transmitted to the peer as part of transitioning into Initiation state is sent again to the peer, the retrans counter is incremented and the protocol instance remains in Initiation state.

6.3.3.3. Reception State

In Reception state a protocol instance is acting as the traditional "responder" in the IKE protocol. It has both sent and received an Init message. An AUTH event causes the instance to leave Reception state, and both an INIT and a TM event cause it to remain in Reception state.

When a protocol instance in Reception state receives an INIT event it signifies that the Init message it sent in order to transition into Reception state was not received by the peer. When it receives a TM event it indicates that its retransmission timer has expired. For an INIT event, the protocol instance first cancels the retransmission timer, after that the behavior a protocol instance takes is the same for an INIT or TM event. The instance retransmits the Init message it sent to the peer in order to transition in to Reception state, it sets its retransmission timer, and it remains in Reception state.

When a protocol instance in Reception state receives an AUTH event it signifies reception of an Auth message from its peer. First the protocol instance cancels its retransmission timer. Next, if the protocol instance has not finished the key exchange protocol (see the note in Section 6.3.3.1) and generated a shared secret it does so here. It then sets the reauth counter to the value of the "committed" field in the processed Auth messages, sets its committed counter to negative one (-1), generates an Auth message for the peer

according to Section 6.1.2.1, transmits the message to the peer, installs the IPsec SAs (per Section 6.2) and transitions to Done state. Note that the retransmission timer is not set.

6.3.3.4. Done State

Done state is the final state of the state machine. An initiator arrives in Done state after it has sent both of its messages to the peer and awaits a final Auth message. A responder arrives in Done state after it has sent and received both messages. To address the possibility of dropped packets and retransmission there are several events that can happen in Done state. Regardless of the event, though, after a state machine enters Done state it never leaves Done state.

When a protocol in Done state receives an INIT event, it signifies the receipt of a retransmitted Init message. Since the protocol has entered Done state it has already received and processed an Init message. If its committed counter is less than zero the protocol instance drops the Init message and remains in Done state. If its committed counter is not less than zero it cancels its retransmission timer, increments the committed counter, generates an Auth message, transmits the Auth message to the peer, sets its retransmission timer, and remains in Done state.

When a protocol in Done state receives a TM event, it signifies that a previously sent Auth message has not been replied to in a timely manner. In this case, the protocol instance, checks the retransmission counter. If it is greater than the thresh counter the protocol instance destroys all state associated with the current run of the protocol (including any IPsec SAs that it might have installed) and transitions back to Nothing state. If the retransmission counter is not greater than the thresh counter, the protocol instance increments the retransmission counter, increments the committed counter, generates an Auth message, transmits the Auth message to the peer, sets the retransmission counter, and remains in Done state.

When a protocol instance in Done state receives an AUTH event it signifies the receipt of an Auth message from the peer. This could be in response to a message from the protocol instance or it could be a retransmission or the replay of an old Auth message. To prevent a storm of Auth messages going back and forth between protocol instances in Done state, the response to an AUTH message is conditional. If the committed counter is less than zero the protocol instance drops the received Auth message and remains in Done state. If the committed counter is not less than zero, the protocol instance cancels its retransmission timer and processes the Auth message. If

processing of the Auth message fails the protocol instance sets the retransmission timer and remains in Done state. If processing of the Auth message succeeds, the value of the "committed" field in the received Auth message is checked. If it is not numerically greater than the reauth counter the message is dropped, the retransmission timer is set, and the protocol instance remains in Done state. If the "committed" field in the received Auth message is numerically greater than the reauth counter, the reauth counter is set to the value in the "committed" field, the committed counter is set to negative one (-1), and an Auth message is generated. The protocol instance then sends the Auth message to the peer, and installs the IPsec SAs (per Section 6.2) and remains in Done state. Note that the retransmission timer is not set.

6.3.4. Cleaning Up Protocol Instances

The state machine ensures that once each side has sent and received an Auth message it installs IPsec SAs. To handle potential lost messages and retransmissions of its final Auth message a protocol instance remains in for a period of time after it has installed its IPsec SAs. These stale protocol instances can have their state deleted and transitioned back to Nothing state after a sufficient period of time. This memo does not define what "sufficient" means but suggests that after installing IPsec SAs, a protocol instance waits at least the amount of time it would spend before retransmissions would cause it to expire. That is:

$$\text{wait} = (\text{thresh} - \text{retrans}) * \text{retrans-period}$$

where "retrans-period" is the amount of time that the retransmission timer is set for. This will ensure that either the protocol instance expires because it retransmitted too many times or it will expire because the protocol has naturally completed.

6.4. IKEv3 Payload Formats

All messages in the IKEv3 protocol consist of an IKE header followed by additional payloads which define the semantics of the message. All payloads contain the same generic header. The IKE header indicates the first payload that follows and the generic header of each payload indicates the payload, if any, that follows. In this fashion payloads are chained together to form messages.

6.4.1. IKE header

The IKE header contains two message identifiers, called Security Parameter Indices (SPIs), one for the transmitter and one for the receiver, an indicator of the first payload of the message,

versioning information, and the type of message, either an Init or an Auth. The format of the IKE header is shown in Figure 4.

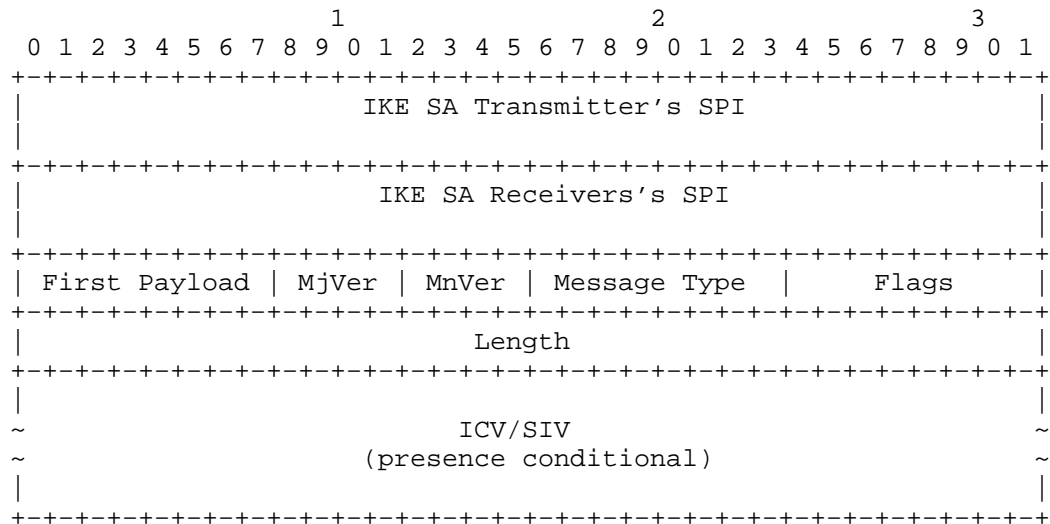


Figure 4: IKE Header Format

- o Transmitter's SPI (8 octets) - A session identifier chosen by the transmitter of the message.
- o Receiver's SPI (8 octets) - A session identifier chosen by the receiver of the message.
- o First Payload (1 octet) - The type of payload that follows the header. See Figure 6.
- o Major Version (4 bits) - The major version of this version of IKE. Implementations based on this memo MUST set the major version to three (3). Receipt of an IKE message with a different Major Version is governed by the memo which defines the version. An implementation that is only compliant with this version of IKE MUST drop any message with a Major version other than three (3).
- o Minor Version (4 bits) - The minor version of this version of IKE. Implementations based on this memo MUST set the minor version to zero (0) on transmitted messages and ignore the Minor Version on received messages.
- o Message Type (1 octet) - The type of message being transmitted: an Init message is type one (1) and an Auth message is type (2).

- o Flags (1 octet) - A bitmask that indicates specific options for the message. The bits in this bitmask are as follows:

```

+---+---+---+---+---+
|X|X|X|V|X|X|X|S|
+---+---+---+---+---+

```

Bits indicated as 'X' MUST be cleared on transmission and ignored on reception. Setting a bit to one (1) indicates that the option applies and clearing the bit to zero (0) indicates the option does not apply.

- * V (Version) - This bit indicates that the transmitter is capable of speaking a higher major version number of the IKE protocol than the one indicated in the major version field of this header.
- * S (Secured) - This bit indicates that the message following this header is authenticated and encrypted. When this bit is set the ICV/SIV field in the header is present.
- o Length (4 octets) - an unsigned integer that indicates the length of the total IKE message (IKE header + all payloads) in octets. Note: if the 'E' bit in the Flags is set this length includes the conditional field to hold the Synthetic Initialization Vector/MAC.
- o ICV/SIV (variable) - a conditional field that is present when the message following the header is secured. This field is the byproduct of authenticated encryption and is required for verified decryption. The exact format of the ICV/SIV field depends on the type of authenticated encryption used by the peers.

6.4.2. Generic IKE payload header

The Generic IKE payload header is used to demark and chain all payloads in a message. Each payload used in a message contains this header. The Generic IKE payload header is defined in Figure 5.

```

          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 |   RESERVED   |           Payload Length           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 5: IKE Header Format

- o Next Payload (1 octet) - Indicates the payload, if any, that follows the current payload. See Figure 6.
- o RESERVED (1 octet) - Unused by this version of IKE. It MUST be set to zero on all payloads in a transmitted message and ignored on all payloads in a received message.
- o Payload Length (2 octets) - An unsigned integer indicating the entire length of the current payload, including this generic header.

Subsequently defined payloads are all shown with the generic header for completeness. Payload types listed here are current as of publication of this memo. Readers are encouraged to see [IKEV3IANA] for the latest values.

Payload Type	Notation	Value
No Next Payload		0
IKE Attributes Payload	IA	1
Identity Payload	ID	2
Nonce Payload	NO	3
Key Exchange Payload	KE	4
Certificate Request Payload	CR	5
Certificate Payload	CE	6
Authentication Payload	AU	7
Address Indication	AI	8
Traffic Selector	TS	9
Security Association Payload	SA	10
Vendor Indication	VE	11

Figure 6: IKEv3 Payload Assignment

The value "No Next Payload" SHALL only be used in the last payload of a message.

6.4.3. IKE Attributes payload

The IKE attributes payload lists a number of attributes that define the manner in which a run of the IKEv3 protocol occurs. Attributes consist of type-value tuples to identify the type of attribute and its particular value. These attributes are offered, not negotiated. See Section 6.1.1.2 for a description of the processing and potential rejection of an IA offer. The IA payload is defined in Figure 7.

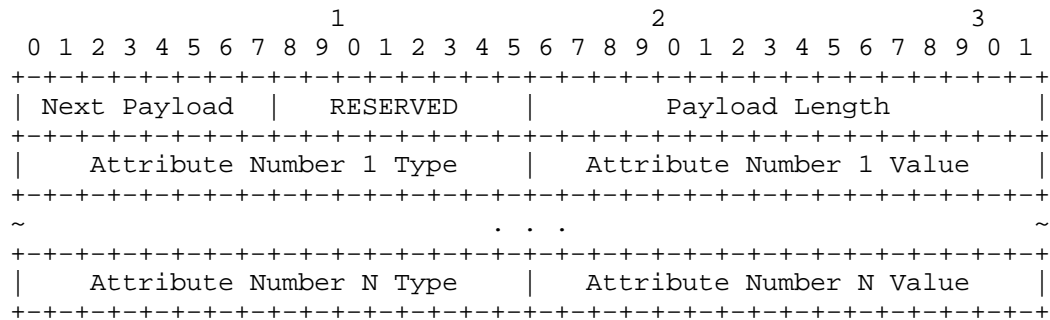


Figure 7: IA Payload

The following attributes types are defined and are indicated by assigning the indicated number to the Attributes Number <X> Type field:

1. Authentication Method
2. Authenticated Encryption Mode
3. Hash Algorithm
4. Diffie-Hellman Group

All other values are reserved to IANA.

When the Attribute Type indicates "Authentication Method", the following values are defined:

1. Digital Signatures
2. Pre-shared Key

All other values are reserved to IANA.

When the Attribute Type indicates "Authentication Encryption Mode", the following values are defined:

1. AES in Synthetic Initialization Mode ([RFC5297]) with a 256-bit key
2. AES in Synthetic Initialization Mode ([RFC5297]) with a 512-bit key

All other values are reserved to IANA.

When the Attribute Type indicates "Hash Algorithm, the following values are defined:

1. SHA-256 ([RFC4634])
2. SHA-512 ([RFC4634])

All other values are reserved to IANA.

When the Attribute Type indicates "Diffie-Hellman Group", the attribute values are taken from the "Diffie-Hellman Group Transform IDs" from [IKEV2IANA].

6.4.4. Identity Payload

The ID payload is used to convey the identity that is to be authenticated by the remote peer.

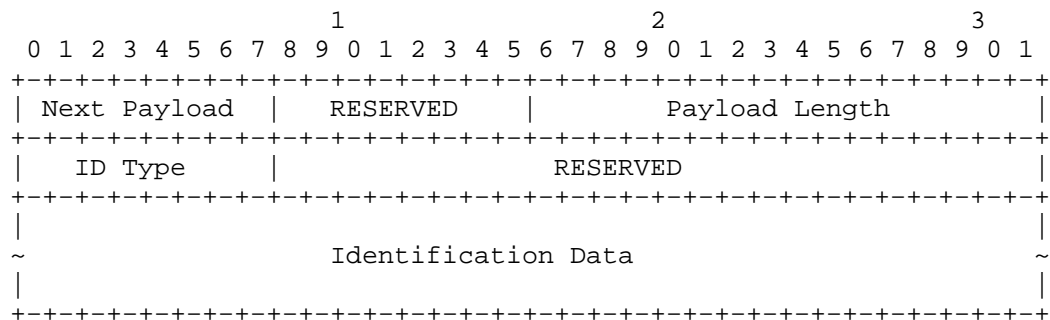


Figure 8: ID Payload

The following ID Types are defined:

ID Type	Value	Description
1	ID_IPV4_ADDR	A single four (4) octet IPv4 address
2	ID_FQDN	A fully-qualified domain name string. An ID_FQDN string MUST NOT contain any terminators (e.g. NULL, CR, etc.). All characters in the ID_FQDN are ASCII.
3	ID_RFC822_ADDR	A fully-qualified RFC 822 email address string. An ID_RFC822_ADDR string MUST NOT contain any terminators (e.g. NULL, CR, etc.). This field SHOULD be treated as UTF-8 encoded text.
4	ID_IPV6_ADDR	A single sixteen (16) octet IPv6 address

- 5 ID_DER_ASN1_DN The binary Distinguished Encoding Rules (DER) encoding of an ASN.1 X.500 Distinguished Name. See [RFC5280].
- 6 ID_DER_ASN1_GN The binary DER encoding of an ASN.1 X.500 GeneralName. See [RFC5280].
- 7 ID_BLOB_ID An opaque octet stream used for identity obfuscation. The two parties to the exchange MUST agree in an out-of-band fashion on how to map a Blob ID to an unobfuscated identity.

Table 1

Identification Data is a variable-length field that contains the identity of the specified type.

6.4.5. Nonce Payload

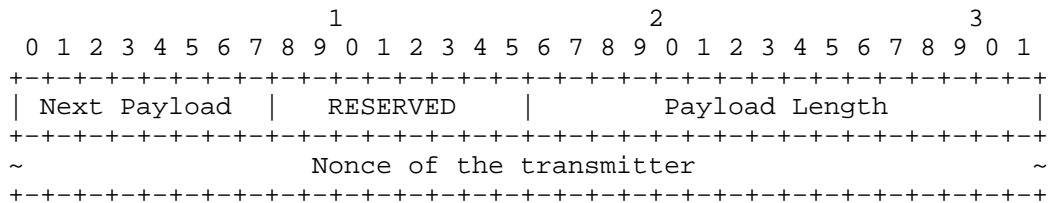


Figure 9: NO Payload

6.4.6. Key Exchange Payload

The KE payload is used to pass data used to perform the key exchange portion of the IKEv3 protocol (see Section 5). The body of the KE payload is authentication method specific. When doing The KE payload is authentication method specific. authentication using digital signatures, the body of the KE payload is a Diffie-Hellman public value. When doing PSK authentication, the body of the KE payload is a concatenation of the Commit and Confirm portions of the Dragonfly key exchange.

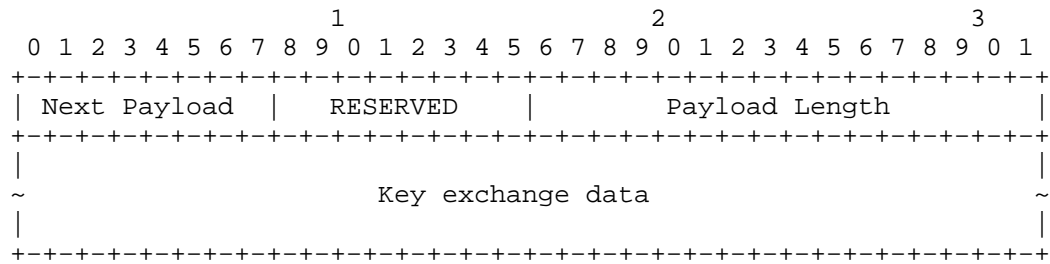


Figure 10: KE Payload

The key exchange data is a variable-length field that contains data to be transmitted to the remote peer to perform a cryptographic key exchange.

6.4.7. Certificate Payload

The CE payload is used to convey a public key which will be used for authentication to a remote peer. The public key can be certified or raw.

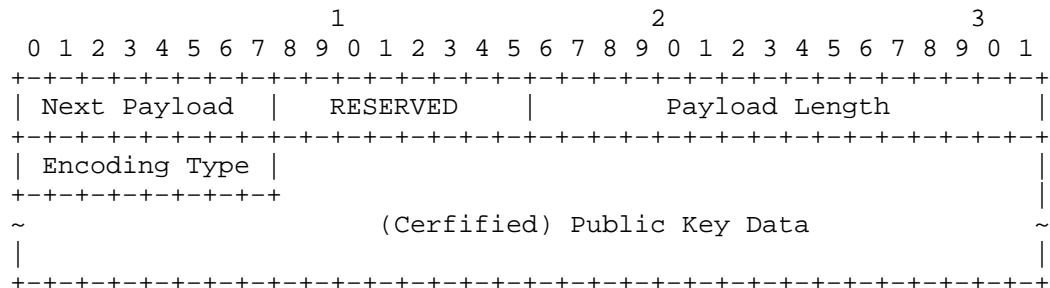


Figure 11: CE Payload

Encoding Description

1	A DER-encoded X.509 certificate
2	Subject public key info for a raw key encoded according to [RFC5480]
3	Subject public key info for a raw key encoded according to [RFC3279]

Table 2

The Public key data is a variable-length field that contains the public key of the specified encoding.

6.4.8. Certificate Request Payload

The CR payload is used to request a certificate from a remote peer. The transmitter indicates a preference for the type of certificate using by setting the Encoding type according to Table 2. The transmitter **SHOULD** indicate a trusted Certification Authority for certified public keys.

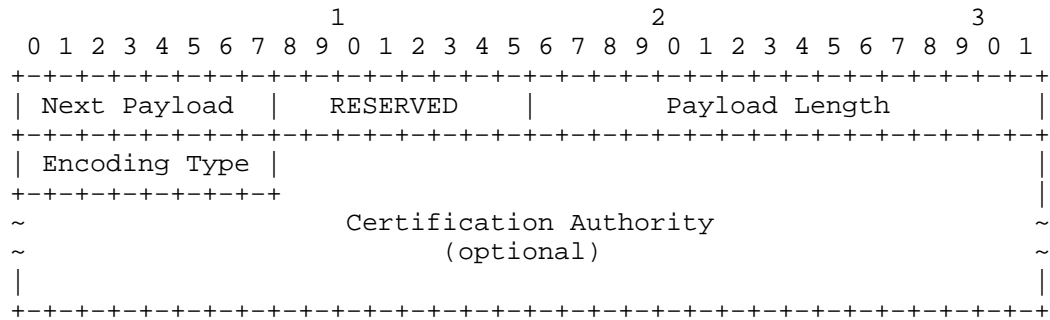


Figure 12: CR Payload

The Certificate Authority field, when present, is a variable-length field that contains the DER encoding of an ASN.1 X.509 IssuerName.

6.4.9. Authentication Payload

The AU payload contains data that authenticates a remote peer. The content of the body of an AU payload is either a digital signature (see Section 5.1.2) when authenticating with digital signatures, or a keyed message authentication function using the negotiated hash algorithm (see Section 5.2.4) when authenticating with a PSK.

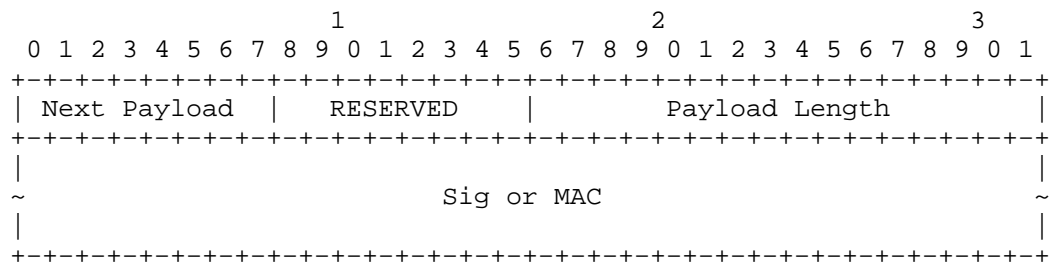


Figure 13: AU Payload

Sig or MAC is a variable-length field that contains either a digital signature of a message authentication code.

6.4.10. Address Indication Payload

The AI payload is used to convey the local view of addressing and port selection to the remote peer for the purposes of NAT detection.

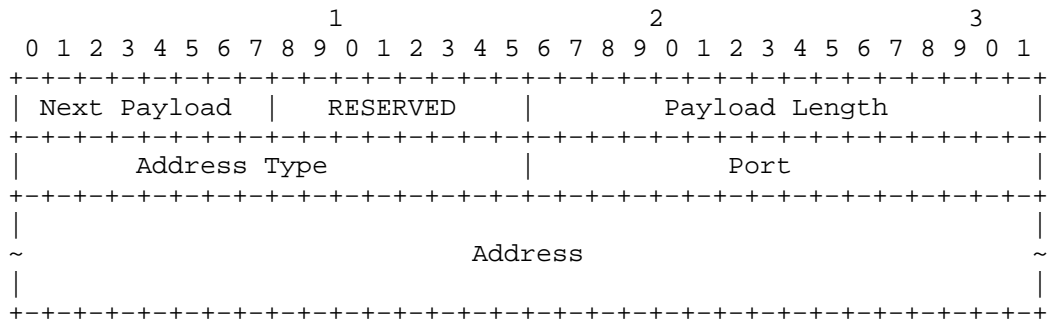


Figure 14: AI Payload

Address Types have the following meaning:

Address Type	Description
1	The Address field is a single four (4) octet IPv4 address
2	The Address field is a single sixteen (16) octet IPv6 address

Table 3

All other Address Types are reserved and MUST NOT be used.

6.4.11. Traffic Selector Payload

The TS payload is used to convey a set of traffic selectors used to identify traffic flows for processing by IPsec security services.

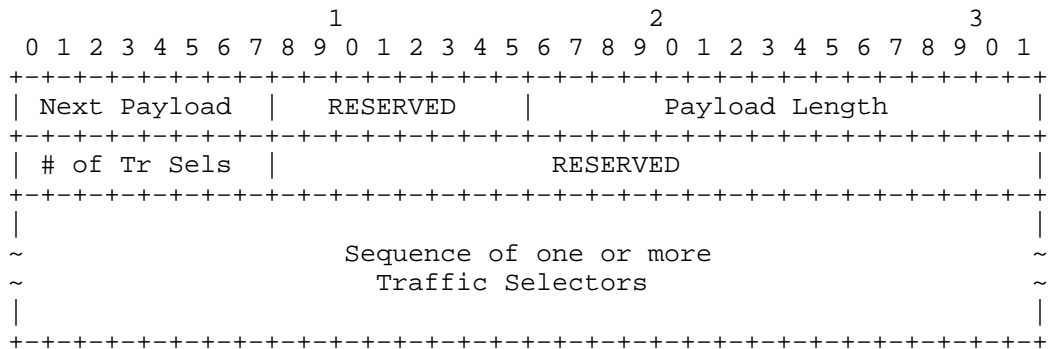


Figure 15: TS Payload

The traffic selectors conveyed to a peer are determined by the local Security Policy Database (see [RFC4301]). They define the data that MUST be protected by IPsec by individual flow. Each Traffic Selector in the set is defined by the following structure:

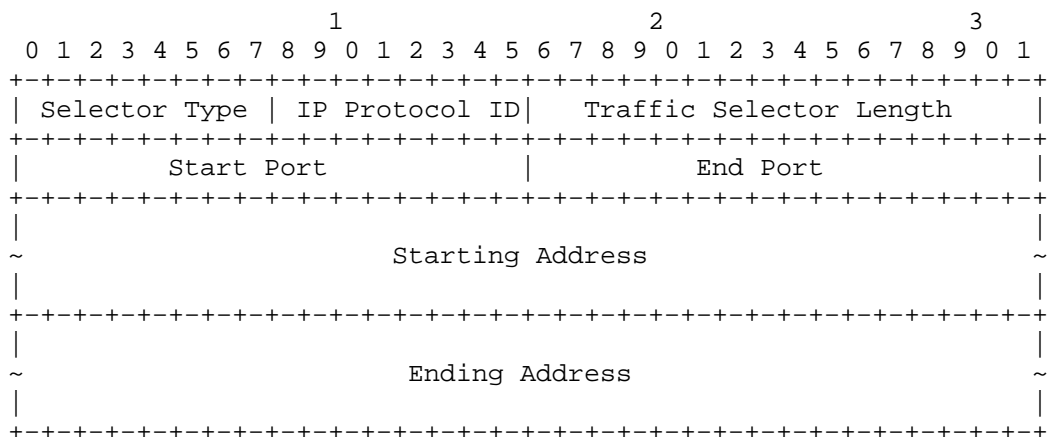


Figure 16: Traffic Selector

- o Select Type (one octet) - either a one (1) to indicate an IPV4_ADDR_RANGE or two (2) to indicate an IPV6_ADDR_RANGE. All other types are invalid and MUST be rejected.
- o IP Protocol ID (one octet) - indicates the associated IP protocol (such as UDP, TCP, and ICMP). A value of zero (0) means that the traffic selector covers all protocols.
- o Traffic Selector Length (two octets) - the total length of the selector, including this sub header.

- o Start Port (two octets) - the lowest port in a range of ports that are covered by this traffic selector. A value of zero (0) means that the traffic selector covers all ports. ICMP and ICMPv6 Type and Code values, as well as Mobile IP version 6 (MIPv6) mobility header (MH) Type values, are represented according to Section 4.4.1.1 of [RFC4301]. ICMP Type and Code values are treated as a single 16-bit integer port number, with Type in the most significant 8 bits and Code in the least significant 8 bits. MIPv6 MH Type and Code values are treated as a single 16-bit integer port number with Type in the most significant 8 bits and the least significant 8 bits set to zero.
- o End Port (two octets) - the highest port in a range of ports that are covered by this traffic selector. For protocols for which port is undefined (including protocol 0), or if all ports are allowed, this field MUST be 65535. ICMP and ICMPv6 Type and Code values, as well as MIPv6 MH Type values, are represented in this field as specified in Section 4.4.1.1 of [RFC4301]. ICMP Type and Code values are treated as a single 16-bit port number with Type in the most significant 8 bits and Code in the least significant 8 bits. MIPv6 MH Type values are treated as a single 16-bit integer port number with Type in the most significant 8 bit and the least significant 8 bits set to zero.
- o Starting Address (variable) - the lowest address in a range of addresses that are covered by this traffic selector. This field will be either sixteen (16) octets or four (4) octets depending on whether the Selector Type was IPV6_ADDR_RANGE or IPV4_ADDR_RANGE, respectively.
- o Ending Address (variable) - the highest address in a range of addresses that are covered by this traffic selector. This field will be either sixteen (16) octets or four (4) octets depending on whether the Selector Type was IPV6_ADDR_RANGE or IPV4_ADDR_RANGE, respectively.

6.4.12. Security Association Payload

The SA payload indicates the type of protection that IPsec will apply to the data that is covered by the Traffic Selector(s) in the TS payload. Protection is described as a number of attributes with each attribute consisting of a type-value tuple to identify the type of attribute and its particular value.

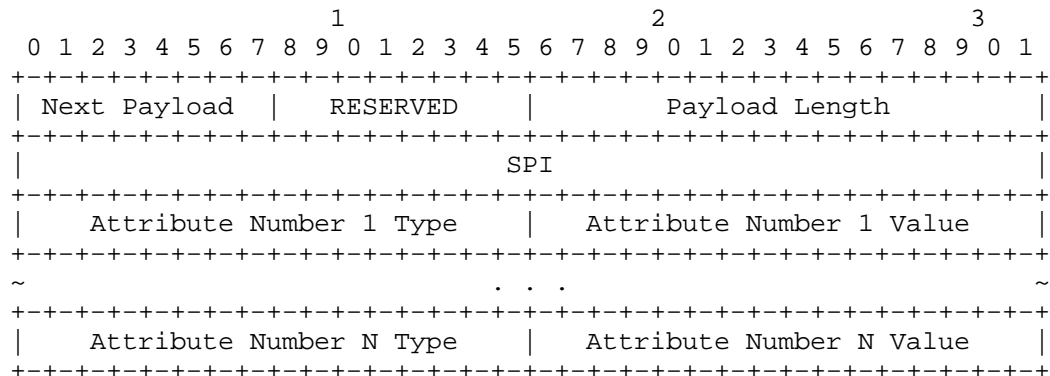


Figure 17: SA Payload

SPI (4 octets) - the Security Parameter Index that the transmitter of the SA payload will use to identify the resulting security association for received IPsec-protected packets.

The following attribute types are defined:

1. Encryption Algorithm
2. Integrity Algorithm
3. Extended Sequence Numbers

If the Encryption Algorithm attribute is present in an SA payload the SA SHALL be for ESP. If it is absent the SA SHALL be for AH. The Integrity Algorithm attribute is OPTIONAL for ESP and MANDATORY for AH. The Extended Sequence Number attribute is OPTIONAL.

The attribute values for the Encryption Algorithm attribute are defined in [IKEV2IANA] in the "Encryption Algorithm Transform IDs" table. The attribute values for the Integrity Algorithm attribute are defined in [IKEV2IANA] in the "Integrity Algorithm Transform IDs" table. The attribute values for the Extended Sequence Numbers attribute are defined in [IKEV2IANA] in the "Extended Sequence Numbers Transform IDs".

6.4.13. Vendor Indication Payload

The VI payload allows vendors of IKEv3 products to identify each other during the protocol.

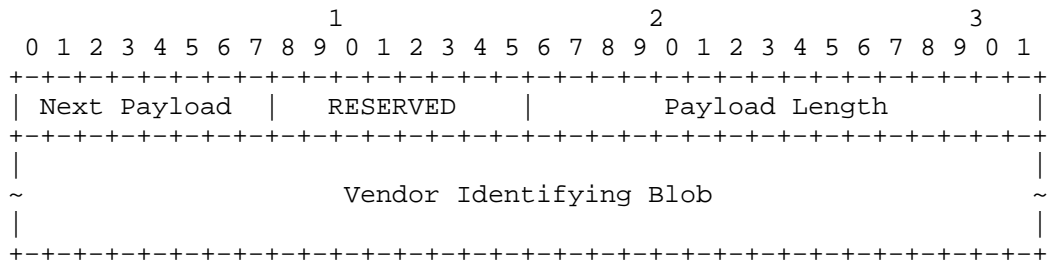


Figure 18: VI Payload

The Vendor Identifying Blob is a variable length field that contains information to identify the vendor of the transmitter of the payload. No registry for vendor identification is used and it is advised that vendors produce an opaque blob that will be different for each run of the protocol to identify themselves. This can be accomplished, for instance, by hashing the transmitter and receiver SPIs and/or the IP addresses of the peers with a vendor-specific constant.

7. Acknowledgements

Portions of the payload descriptions (e.g. Traffic Selector payload) were lifted from [RFC5996]. The author thanks the editors of that document and the IPsecME Working Group that produced that document.

8. IANA Considerations

This section is incomplete.

9. Security Considerations

This section is incomplete.

10. References

10.1. Normative References

- [IKEV2IANA]
 "Internet Key Exchange Version 2 (IKEv2) Parameters",
 <<http://www.iana.org>>.
- [IKEV3IANA]
 "Internet Key Exchange Version 3 (IKEv3) Parameters",

<<http://www.iana.org>>.

- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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, April 2002.
- [RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, January 2005.
- [RFC4634] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RFC5297] Harkins, D., "Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)", RFC 5297, October 2008.
- [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, March 2009.
- [RFC6090] McGrew, D., Igoe, K., and M. Salter, "Fundamental Elliptic Curve Cryptography Algorithms", RFC 6090, February 2011.

10.2. Informative References

- [RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

- [RFC4718] Eronen, P. and P. Hoffman, "IKEv2 Clarifications and Implementation Guidelines", RFC 4718, October 2006.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

Author's Address

Dan Harkins (editor)
Aruba Networks
1322 Crossman avenue
Sunnyvale, California 94089
United States of America

Phone: +1 408 227 4500
Email: dharkins@arubanetworks.com

IPsecME Working Group
Internet-Draft
Intended status: Informational
Expires: January 17, 2014

V. Manral
HP
S. Hanna
Juniper
July 16, 2013

Auto Discovery VPN Problem Statement and Requirements
draft-ietf-ipsecme-ad-vpn-problem-09

Abstract

This document describes the problem of enabling a large number of systems to communicate directly using IPsec to protect the traffic between them. It then expands on the requirements, for such a solution.

Manual configuration of all possible tunnels is too cumbersome in many such cases. In other cases the IP address of endpoints change or the endpoints may be behind NAT gateways, making static configuration impossible. The Auto Discovery VPN solution will address these requirements.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 17, 2014.

Copyright Notice

Copyright (c) 2013 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. Terminology	3
1.2. Conventions Used in This Document	4
2. Use Cases	4
2.1. Endpoint-to-Endpoint VPN Use Case	4
2.2. Gateway-to-Gateway VPN Use Case	5
2.3. Endpoint-to-Gateway VPN Use Case	5
3. Inadequacy of Existing Solutions	6
3.1. Exhaustive Configuration	6
3.2. Star Topology	6
3.3. Proprietary Approaches	7
4. Requirements	7
4.1. Gateway and Endpoint Requirements	7
5. Security Considerations	10
6. IANA Considerations	11
7. Acknowledgements	11
8. Normative References	11
Authors' Addresses	11

1. Introduction

IPsec [RFC4301] is used in several different cases, including tunnel-mode site-to-site VPNs and Remote Access VPNs. Both tunneling modes for IPsec gateways and host-to-host transport mode are supported in this document.

The subject of this document is the problem presented by large scale deployments of IPsec and the requirements on a solution to address the problem. These may be a large collection of VPN gateways connecting various sites, a large number of remote endpoints connecting to a number of gateways or to each other, or a mix of the two. The gateways and endpoints may belong to a single administrative domain or several domains with a trust relationship.

Section 4.4 of RFC 4301 describes the major IPsec databases needed for IPsec processing. It requires an extensive configuration for each tunnel, so manually configuring a system of many gateways and endpoints becomes infeasible and inflexible.

The difficulty is that a lot of configuration mentioned in RFC 4301 is required to set up a Security Association. IKE implementations need to know the identity and credentials of all possible peer systems, as well as the addresses of hosts and/or networks behind them. A simplified mechanism for dynamically establishing point-to-point tunnels is needed. Section 2 contains several use cases that motivate this effort.

1.1. Terminology

ADVPN - Auto Discovery Virtual Private Network (ADVPN) is VPN solution that enables a large number of systems to communicate directly, with minimal configuration and operator intervention using IPsec to protect communication between them.

Endpoint - A device that implements IPsec for its own traffic but does not act as a gateway.

Gateway - A network device that implements IPsec to protect traffic flowing through the device.

Point-to-Point - Communication between two parties without active participation (e.g. encryption or decryption) by any other parties.

Hub - The central point in a star topology/ dynamic full mesh topology, or one of the central points in the full mesh style VPN, i.e. gateway where multiple other hubs or spokes connect to. The hubs usually forward traffic coming from encrypted links to other encrypted links, i.e. there are no devices connected to it in clear.

Spoke - The endpoint in a star topology/ dynamic full mesh topology, or gateway which forwards traffic from multiple cleartext devices to other hubs or spokes, and some of those other devices are connected to it in clear (i.e. it encrypts data coming from cleartext devices and forwards it to the ADVPN).

ADVPN Peer - any member of an ADVPN including gateways, endpoints, hub or spoke.

Star topology - This is the topology where there is direct connectivity only between the hub and spoke and communication between the 2 spokes happens through the hub.

Allied and Federated Environments - Environments where we have multiple different organizations that have close association and need to connect to each other.

Full Mesh topology - This is the topology where there is a direct connectivity between every Spoke to every other Spoke directly, without the traffic between the spokes having to be redirected through an intermediate hub device.

Dynamic Full Mesh topology - This is the topology where direct connections exist in a hub and spoke manner, but dynamic connections are created/ removed between the spokes on a need basis.

Security Association (SA) - Defined in [RFC4301].

1.2. Conventions Used in This Document

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

2. Use Cases

This section presents the key use cases for large-scale point-to-point VPN.

In all of these use cases, the participants (endpoints and gateways) may be from a single organization (administrative domain) or from multiple organizations with an established trust relationship. When multiple organizations are involved, products from multiple vendors are employed so open standards are needed to provide interoperability. Establishing communications between participants with no established trust relationship is out of scope for this effort.

2.1. Endpoint-to-Endpoint VPN Use Case

Two endpoints wish to communicate securely via a point-to-point Security Association (SA).

The need for secure endpoint-to-endpoint communications is often driven by a need to employ high-bandwidth, low -latency local connectivity instead of using slow, expensive links to remote gateways. For example, two users in close proximity may wish to place a direct, secure video or voice call without needing to send the call through remote gateways, which would add latency to the call, consume precious remote bandwidth, and increase overall costs. Such a use case also enables connectivity when both users are behind NAT gateways. Such a use case ought to allow for seamless connectivity even as endpoints roam, even if they are moving out from behind a NAT gateway, from behind one NAT gateway to behind another, or from a standalone position to behind a NAT gateway.

In a star topology, when two endpoints communicate they need a mechanism for authentication, such that they do not expose themselves to impersonation by the other spoke endpoint.

2.2. Gateway-to-Gateway VPN Use Case

A typical Enterprise traffic model follows a star topology, with the gateways connecting to each other using IPsec tunnels.

However for voice and other rich media traffic that requires a lot of bandwidth or is performance sensitive, the traffic tromboning (taking a suboptimal path) to the hub can create traffic bottlenecks on the hub and can lead to an increase in cost. A fully meshed solution would make best use of the available network capacity and performance but the deployment of a fully meshed solution involves considerable configuration, especially when a large number of nodes are involved. It is for this purpose spoke-to-spoke tunnels are dynamically created and torn-down. For the reasons of cost and manual error reduction, it is desired that there be minimal configuration on each gateway.

The solution ought to work in cases where the endpoints are in different administrative domains, albeit, ones that have an existing trust relationship (for example two organisations who are collaborating on a project, they may wish to join their networks, whilst retaining independent control over configuration). It is highly desirable that the solution works for the star, full mesh as well as dynamic full mesh topology.

The solution ought to also address the case where gateways use dynamic IP addresses.

Additionally, the routing implications of gateway-to-gateway communication need to be addressed. In the simple case, selectors provide sufficient information for a gateway to forward traffic appropriately. In other cases, additional tunneling (e.g., Generic Routing Encapsulation - GRE) and routing (e.g., Open Shortest Path First - OSPF) protocols are run over IPsec tunnels, and the configuration impact on those protocols needs to be considered. There is also the case when Layer-3 Virtual Private Networks (L3VPNs) operate over IPsec Tunnels.

When two gateways communicate, they need to use a mechanism for authentication, such that they do not expose themselves to the risk of impersonation by the other entities.

2.3. Endpoint-to-Gateway VPN Use Case

A mobile endpoint ought to be able to use the most efficient gateway as it roams in the internet.

A mobile user roaming on the Internet may connect to a gateway, which because of roaming is no longer the most efficient gateway to use (reasons could be cost/ efficiency/ latency or some other factor). The mobile user ought to be able to discover and then connect to the current most efficient gateway in a seamless way without having to bring down the connection.

3. Inadequacy of Existing Solutions

Several solutions exist for the problems described above. However, none of these solutions is adequate, as described here.

3.1. Exhaustive Configuration

One simple solution is to configure all gateways and endpoints in advance with all the information needed to determine which gateway or endpoint is optimal and to establish an SA with that gateway or endpoint. However, this solution does not scale in a large network with hundreds of thousands of gateways and endpoints, especially when multiple administrative domains are involved and things are rapidly changing (e.g. mobile endpoints). Such a solution is also limited by the smallest endpoint/ gateway, as the same exhaustive configuration is to be applied on all endpoints/ gateways. A more dynamic, secure and scalable system for establishing SAs between gateways is needed.

3.2. Star Topology

The most common way to address a part of this this problem today is to use what has been termed a "star topology". In this case one or a few gateways are defined as "hub gateways", while the rest of the systems (whether endpoints or gateways) are defined as "spokes". The spokes never connect to other spokes. They only open tunnels with the hub gateways. Also for a large number of gateways in one administrative domain, one gateway may be defined as the hub, and the rest of the gateways and remote access clients connect only to that gateway.

This solution however is complicated by the case when the spokes use dynamic IP addresses and DNS with dynamic updates needs to be used. It is also desired that there is minimal to no configuration on the hub as the number of spokes increases and new spokes are added and deleted randomly.

Another problem with the star topology is that it creates a high load on the hub gateways as well as on the connection between the spokes

and the hub. This load is both in processing power and in network bandwidth. A single packet in the hub-and-spoke scenario can be encrypted and decrypted multiple times. It would be much preferable if these gateways and clients could initiate tunnels between them, bypassing the hub gateways. Additionally, the path bandwidth to these hub gateways may be lower than that of the path between the spokes. For example, two remote access users may be in the same building with high-speed wifi (for example, at an IETF meeting). Channeling their conversation through the hub gateways of their respective employers seems extremely wasteful, as well as having lower bandwidth.

The challenge is to build large scale, IPsec-protected networks that can dynamically change with minimal administrative overhead.

3.3. Proprietary Approaches

Several vendors offer proprietary solutions to these problems. However, these solutions offer no interoperability between equipment from one vendor and another. This means that they are generally restricted to use within one organization, and it is harder to move off such solutions as the features are not standardized. Besides multiple organizations cannot be expected to all choose the same equipment vendor.

4. Requirements

This section defines the requirements, on which the solution will be based.

4.1. Gateway and Endpoint Requirements

1. For any network topology (star, full mesh and dynamic full mesh), when a new gateway or endpoint is added, removed, or changed, configuration changes are minimized as follows. Adding or removing a spoke in the topology **MUST NOT** require configuration changes to the hubs other than where the spoke was connected to and **SHOULD NOT** require configuration changes to the hub the spoke was connected to. The changes also **MUST NOT** require configuration changes in other spokes.

Specifically, when evaluating potential proposals, we will compare them by looking at how many endpoints or gateways must be reconfigured when a new gateway or endpoint is added, removed, or changed and how substantial this reconfiguration is, besides the amount of static configuration required.

This requirement is driven by use cases 2.1 and 2.2 and by the scaling limitations pointed out in section 3.1.

2. ADVPN peers MUST allow IPsec Tunnels to be setup with other members of the ADVPN without any configuration changes, even when peer addresses get updated every time the device comes up. This implies that SPD entries or other configuration based on peer IP address will need to be automatically updated, avoided, or handled in some manner to avoid a need to manually update policy whenever an address changes.

3. In many cases additional tunneling protocols (e.g. GRE) or Routing protocols (e.g. OSPF) are run over the IPsec tunnels. Gateways MUST allow for the operation of tunneling and Routing protocols operating over spoke-to-spoke IPsec Tunnels with minimal or no, configuration impact. The ADVPN solution SHOULD NOT increase the amount of information required to configure protocols running over IPsec tunnels.

4. In the full mesh and dynamic full mesh topology, Spokes MUST allow for direct communication with other spoke gateways and endpoints. In the star topology mode, direct communication between spokes MUST be disallowed.

This requirement is driven by use cases 2.1 and 2.2 and by the limitations of a star topology pointed out in section 3.2.

5. Any of the ADVPN Peers MUST NOT have a way to get the long term authentication credentials for any other ADVPN Peers. The compromise of an Endpoint MUST NOT affect the security of communications between other ADVPN Peers. The compromise of a Gateway SHOULD NOT affect the security of the communications between ADVPN Peers not associated with that Gateway.

This requirement is driven by use case 2.1. ADVPN Peers (especially Spokes) become compromised fairly often. The compromise of one ADVPN Peer SHOULD NOT affect the security of other unrelated ADVPN Peers.

6. Gateways SHOULD allow for seamless handoff of sessions in case endpoints are roaming, even if they cross policy boundaries. This would mean the data traffic is minimally affected even as the handoff happens. External factors like firewall, NAT boxes that will be part of the overall solution when DVPN is deployed will not be considered part of this solution.

Such endpoint roaming may affect not only the endpoint-to-endpoint SA but also the relationship between the endpoints and gateways (such as when an endpoint roams to a new network that is handled by a different gateway).

This requirement is driven by use case 2.1. Today's endpoints are mobile and transition often between different networks (from 4G to WiFi and among various WiFi networks).

7. Gateways SHOULD allow for easy handoff of a session to another gateway, to optimize latency, bandwidth, load balancing, availability, or other factors, based on policy.

This ability to migrate traffic from one gateway to another applies regardless of whether the gateways in question are hubs or spokes. It even applies in the case where a gateway (hub or spoke) moves in the network, as may happen with a vehicle-based network.

This requirement is driven by use case 2.3.

8. Gateways and endpoints MUST have the capability to participate in an ADVPN even when they are located behind NAT boxes. However, in some cases they may be deployed in such a way that they will not be fully reachable behind a NAT box. It is especially difficult to handle cases where the Hub is behind a NAT box. Where the two endpoints are both behind separate NATs, communication between these spokes SHOULD be supported using workarounds such as port forwarding by the NAT or detecting when two spokes are behind uncooperative NATs and using a hub in that case.

This requirement is driven by use cases 2.1 and 2.2. Endpoints are often behind NATs and gateways sometimes are. IPsec SHOULD continue to work seamlessly regardless, using ADVPN techniques whenever possible and providing graceful fallback to hub and spoke techniques as needed.

9. Changes such as establishing a new IPsec SA SHOULD be reportable and manageable. However, creating a MIB or other management technique is not within scope for this effort.

This requirement is driven by manageability concerns for all the use cases, especially use case 2.2. As IPsec networks become more dynamic, management tools become more essential.

10. To support allied and federated environments, endpoints and gateways from different organizations SHOULD be able to connect to each other.

This requirement is driven by demand for all the use cases in federated and allied environments.

11. The administrator of the ADVPN SHOULD allow for the configuration of a Star, Full mesh or a partial full mesh topology, based on which tunnels are allowed to be setup.

This requirement is driven by demand for all the use cases in federated and allied environments.

12. The ADVPN solution SHOULD be able to scale for multicast traffic.

This requirement is driven by the use case 2.2, where the amount of rich media multicast traffic is increasing.

13. The ADVPN solution SHOULD allow for easy monitoring, logging and reporting of the dynamic changes, to help for trouble shooting such environments.

This requirement is driven by demand for all the use cases in federated and allied environments.

14. There is also the case when L3VPNs operate over IPsec Tunnels, for example Provider Edge (PE) based VPN's. An ADVPN MUST support L3VPN as an application protected by the IPsec Tunnels.

This requirement is driven by demand for all the use cases in federated and allied environments.

15. There ADVPN solution SHOULD allow the enforcement of per peer QoS in both the Star as well as the Full Mesh topology.

This requirement is driven by demand for all the use cases in federated and allied environments.

16. There ADVPN solution SHOULD take care of not letting the Hub to be a single point of failure.

This requirement is driven by demand for all the use cases in federated and allied environments.

5. Security Considerations

This is a Problem statement and requirement document for the ADVPN solution, and in itself does not introduce any new security concerns. The solution to the problems presented in this draft may involve dynamic updates to databases defined by RFC 4301, such as the

Security Policy Database (SPD) or the Peer Authorization Database (PAD).

RFC 4301 is silent about the way these databases are populated, and it is implied that these databases are static and pre-configured by a human. Allowing dynamic updates to these databases must be thought out carefully, because it allows the protocol to alter the security policy that the IPsec endpoints implement.

One obvious attack to watch out for is stealing traffic to a particular site. The IP address for `www.example.com` is `192.0.2.10`. If we add an entry to an IPsec endpoint's SPD that says that traffic to `192.0.2.10` is protected through peer Gw-Mallory, then this allows Gw-Mallory to either pretend to be `www.example.com` or to proxy and read all traffic to that site. Updates to this database requires a clear trust model.

Hubs can be a single point of failure which can cause loss of connectivity of the entire system, that can be a big security issue. Any ADVPN solution design should take care of these concerns.

6. IANA Considerations

No actions are required from IANA for this informational document.

7. Acknowledgements

Many people have contributed to the development of this problem statement and many more will probably do so before we are done with it. While we cannot thank all contributors, some have played an especially prominent role. Yoav Nir, Yaron Sheffer, Jorge Coronel Mendoza, Chris Ulliot, and John Veizades wrote the document upon which this draft was based. Geoffrey Huang, Toby Mao, Suresh Melam, Praveen Sathyanarayan, Andreas Steffen, Brian Weis, and Lou Berger provided essential input.

8. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

Authors' Addresses

Vishwas Manral
Hewlett-Packard Co.
19111 Pruneridge Ave.
Cupertino, CA 95113
USA

Email: vishwas.manral@hp.com

Steve Hanna
Juniper Networks, Inc.
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
USA

Email: shanna@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 7, 2013

Y. Nir
Check Point
December 4, 2012

A TCP transport for the Internet Key Exchange
draft-ietf-ipsecme-ike-tcp-01

Abstract

This document describes using TCP for IKE messages. This facilitates the transport of large messages over paths where fragments are either dropped, or where packet loss makes the use of large UDP packets unreliable.

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 June 7, 2013.

Copyright Notice

Copyright (c) 2012 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.

1. Introduction

The Internet Key Exchange version 2 (IKEv2), specified in [RFC5996] uses UDP to transport the exchange messages. Some of those messages may be fairly large. Specifically, the messages of the IKE_AUTH exchange can become quite large, as they may contain a chain of certificates, an "Auth" payload (that may contain a public key signature), CRLs, and some configuration information that is carried in the CFG payload.

When such UDP packets exceed the path MTU, they get fragmented. This increases the probability of packets being dropped. The retransmission mechanisms in IKE (as described in section 2.1 of RFC 5996) takes care of that as long as packet loss is at a reasonable level. More recently we have seen a number of service providers dropping fragmented packets. Firewalls and NAT devices need to keep state for each packet where some (but not all) of the fragments have passed through. This creates a burden in terms of memory, especially for high capacity devices such as Carrier-Grade NAT (CGN) or high capacity firewalls.

The BEHAVE working group has an Internet Draft describing required behavior of CGNs ([I-D.ietf-behave-lsn-requirements]). It requires CGNs to comply with [RFC4787], which in section 11 requires NAT devices to support fragments. However, some people deploying IKE have found that some ISPs have begun to drop fragments in preparation for deploying CGNs. While we all hope for a future where all devices comply with the emerging standards, or even a future where CGNs are not required, we have to make IKE work today.

The solution described in this document is to transport the IKE messages over a TCP ([RFC0793]) connection rather than over UDP. IKE packets describe their own length, so they are well-suited for transport over a stream-based connection such as TCP. The Initiator opens a TCP connection to the Responder's port 500, sends the requests and receives the responses, and then closes the connection. TCP can handle arbitrary-length messages, works well with any sized data, and is well supported by all ISP infrastructure.

1.1. Non-Goals of this Specification

Firewall traversal is not a goal of this specification. If a firewall has a policy to block IKE and/or IPsec, hiding the IKE exchange in TCP is not expected to help. Some implementations hide both IKE and IPsec in a TCP connection, usually pretending to be HTTPS by using port 443. This has a significant impact on bandwidth and gateway capacity, and even this is defeated by better firewalls. SSL VPNs tunnel IP packets over TLS, but the latest firewalls are

also TLS proxies, and are able to defeat this as well.

This document is not part of that arms race. It is only meant to allow IKE to work when faced with broken infrastructure that drops large IP packets.

1.2. Conventions Used in This Document

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

2. The Protocol

2.1. Initiator

An Initiator MAY try IKE using TCP for any request. It opens a TCP connection from an arbitrary port to port 500 of the Responder. When the three-way handshake completes, the Initiator MUST send the request. If the Initiator knows that this request is the last request needed at this time, it MAY half-close the TCP connection, or it MAY wait until the last response has been received. When all responses have been received, the Initiator MUST close the connection. If the peer has closed the connection before all requests have been transmitted or responded to, the Initiator SHOULD either open a new TCP connection or transmit them over UDP again.

An initiator MUST accept responses sent over IKE within the same connection, but MUST also accept responses over other transports, if the request had been sent over them as well.

An initiator that is configured to respond to IKE over TCP on some port, and is not prevented from receiving TCP connections by network address translation (see Section 3.2), MUST send an IKE_TCP_SUPPORTED notification (Section 2.5) in the Initial request.

Note that stateless cookies may be dependent on some of the parameters of the connection, so retransmitting the IKE_INITIAL request with a stateless cookie over a different transport may cause the cookie to be invalid. For this reason, retransmissions with a cookie SHOULD be sent over the same transport.

2.2. Responder

A Responder MAY accept TCP connections to port 500, and if it does, it MUST accept IKE requests over this connection. Responses to requests received over this connection MUST also go over this

connection. If the connection has closed before the Responder has had a chance to respond, it MUST NOT respond over UDP, but MUST instead wait for a retransmission over UDP or over another TCP connection.

The responder MUST accept different requests on different transports. Specifically, the Responder MUST NOT rely on subsequent requests coming over the same transport. For example, it is entirely acceptable to have the IKE_INITIAL exchange come over UDP port 500, while the IKE_AUTH request comes over TCP, and some following requests might come over UDP port 4500 (because NAT has been detected).

A responder that is configured to support IKE over TCP and receives an IKEv2 Initial request over any other transport MUST send an IKE_TCP_SUPPORTED notification (Section 2.5) in the Initial response. the responder MAY send this notification even if the Initial request was received over TCP.

If the responder has some requests of its own to send, it MUST NOT use a connection that has been opened by a peer. Instead, it MUST either use UDP or else open a new TCP connection to the original Initiator's TCP port, specified in the IKE_TCP_SUPPORTED notification in the Initial request. If the Initial request did not include this notification, the original Responder MUST NOT initiate IKE over TCP to the original Initiator.

The normal flow of things is that the Initiator opens a connection and closes its side first. The responder closes after sending the last response where the initiator has already half-closed the connection. If, however, a significant amount of time has passed, and neither new requests arrive nor the connection is closed by the initiator, the Responder MAY close or even reset the connection.

This specification makes no recommendation as to how long such a timeout should be, but a few seconds should be enough.

The stateless cookie mechanism in IKEv2 only assures that the initiator is able to respond to the address and port of the request. TCP already provides this with the three-way handshake. If the IKE_INITIAL exchange is transmitted over TCP, the stateless cookie mechanism SHOULD NOT be used.

2.3. Transmitter

The transmitter, whether an initiator transmitting a request or a responder transmitting a response MUST NOT retransmit over the same connection. TCP takes care of that. It SHOULD send the IKE header

and the IKE payloads with a single command or in rapid succession, because the receiver might block on reading from the socket.

2.4. Receiver

The IKE header is copied from RFC 5996 below for reference:

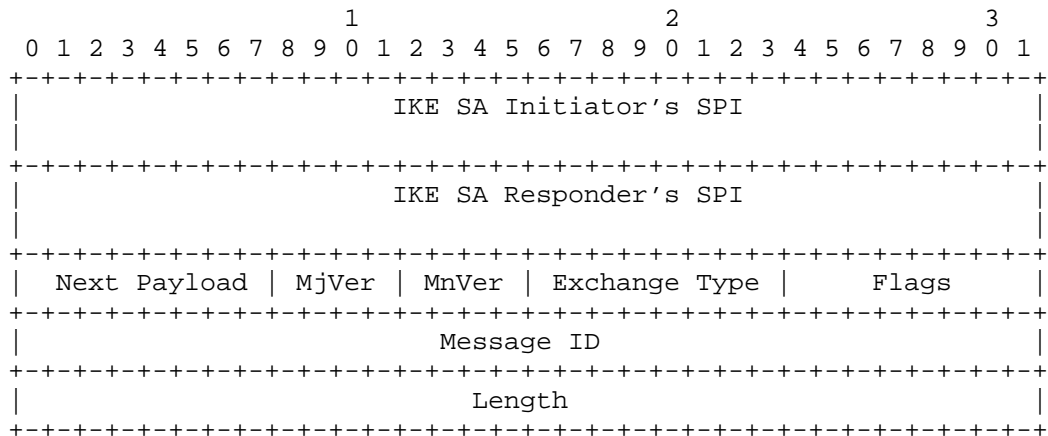


Figure 1: IKE Header Format

The receiver MUST first read in the 28 bytes that make up the IKE header. The Responder then subtracts 28 from the length field, and reads the resulting number of bytes. The combined message, comprised on 28 header bytes and whatever number of payload bytes is processed the same way as regular UDP messages. That includes retransmission detection, with one slight difference: if a retransmitted request is detected, the response is retransmitted as well, but using the current TCP connection rather than whatever other transport had been used for the original transmission of the request.

2.5. IKE_TCP_SUPPORTED Notification

This notification is sent by a responder over non-TCP transports to inform the initiator that this specification is supported and configured.

The Notify payload is formatted 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
+-----+-----+-----+-----+-----+-----+-----+-----+
! Next Payload !C!  RESERVED   !               Payload Length   !
+-----+-----+-----+-----+-----+-----+-----+-----+
!  Protocol ID !   SPI Size    !IKE_TCP_SUPPORTED Message Type !
+-----+-----+-----+-----+-----+-----+-----+-----+
|               TCP Port               |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

- o Protocol ID (1 octet) MUST be 0.
- o SPI Size (1 octet) MUST be zero, in conformance with section 3.10 of RFC 5996.
- o IKE_TCP_SUPPORTED Notify Message Type (2 octets) - MUST be xxxxxx, the value assigned for IKE_TCP_SUPPORTED. TBA by IANA.
- o TCP port (2 octets) - The TCP port to which the recipient should open TCP connections. This is not necessarily the same port that the IKE gateway is listening to. See Section 3.2. If the sender is not subject to network address translation, the port SHOULD be 500.

3. Operational Considerations

Most IKE messages are relatively short. All but the IKE_AUTH exchange in IKEv2 are comprised of short messages that fit in a single packet on most networks. The Informational exchange could be an exception, as it may contain arbitrary-length CFG payloads, but in practice this is not done. It is only the IKE_AUTH exchange that has long messages. UDP has advantages in lower latency and lower resource consumption, so it makes sense to use UDP whenever TCP is not required.

The requirements in Section 2.2 were written so that different requests may be sent over different transports. The initiator can choose the transport on a per-request basis. So one obvious policy would be to do everything over UDP except the specific requests that tend to become too big. This way the first messages use UDP, and the Initiator can set up the TCP connection at the same time, eliminating the latency penalty of using TCP. This may not always be the most efficient policy, though. It means that the first messages sent over TCP are relatively large ones, and TCP slow start may cause an extra roundtrip, because the message has exceeded the transmission window. An initiator using this policy MUST NOT go to TCP if the responder has not indicated support by sending the IKE_TCP_SUPPORTED notification (Section 2.5) in the Initial response.

An alternative method, that is probably easier for the Initiator to

implement, is to do an entire "mission" using the same transport. So if TCP is needed for long messages and an IKE SA has not yet been created, the Initiator will open a TCP connection, and perform all 2-4 requests needed to set up a child SA over the same connection.

Yet another policy would be to begin by using UDP, and at the same time set up the TCP connection. If at any point the TCP handshake completes, the next requests go over that connection. This method can be used to auto-discover support of TCP on the responder. This is easier for the user than configuring which peers support TCP, but has the potential of wasting resources, as TCP connections may finish the three-way handshake just when IKE over UDP has finished. The requirements from the responder ensure that all these policies will work.

3.1. Liveness Check

The TCP connections described in this document are short-lived. We do not expect them to stay for the lifetime of the SA, but to get torn down by either side within seconds of the SA being set up. Because of this, they are not well-suited for the transport of short requests such as those for liveness check.

Although liveness checks MAY be sent over TCP, this is not recommended.

On the other hand, see Section 3.2 for when liveness check should be used.

3.2. Network Address Translation

If the IKE gateway is subject to network address translation (NAT), TCP ports may be translated, so that one port on the NAT device gets translated to some other port on the gateway. In this case, the gateway MUST advertise the NAT device port in the IKE_TCP_SUPPORTED notification.

In some cases, the NAT or some other box prevents incoming TCP connections to the IKE peer behind it. In these cases, the IKE peer MUST NOT advertise support using the IKE_TCP_SUPPORTED notification.

When IKE peers detect the presence of a NAT device during the IKE exchange, they typically switch to working over UDP port 4500. Sending the IKE_AUTH messages over this UDP port creates a port mapping entry on the NAT device, and this mapping can then be used for bidirectional traffic between the peers. When using IKE over TCP, this mapping is not created, so traffic can only flow from the initiator to the responder. To make a bidirectional mapping, it is

RECOMMENDED that when NAT is detected, initiators initiate a liveness check using UDP 4500 to the responders immediately following the successful IKE_AUTH exchange.

4. Security Considerations

Most of the security considerations for IKE over TCP are the same as those for UDP as in RFC 5996.

For the Responder, listening to TCP port 500 involves all the risks of maintaining any TCP server. Precautions against DoS attacks, such as SYN cookies are RECOMMENDED. see [RFC4987] for details.

5. IANA Considerations

IANA is requested to assign a notify message type from the status types range (16418-40959) of the "IKEv2 Notify Message Types" registry with name "IKE_TCP_SUPPORTED"

No IANA action is required for the TCP port, as TCP port 500 is already allocated to "ISAKMP".

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

6.2. Informative References

- [I-D.ietf-behave-lsn-requirements]
Perreault, S., Yamagata, I., Miyakawa, S., Nakagawa, A., and H. Ashida, "Common requirements for Carrier Grade NATs (CGNs)", draft-ietf-behave-lsn-requirements-09 (work in progress), August 2012.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC4787] Audet, F. and C. Jennings, "Network Address Translation

(NAT) Behavioral Requirements for Unicast UDP", BCP 127,
RFC 4787, January 2007.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common
Mitigations", RFC 4987, August 2007.

Author's Address

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 67897
Israel

Email: ynir@checkpoint.com

IP Security Maintenance and Extensions
(ipsecme)
Internet-Draft
Intended status: Informational
Expires: April 4, 2013

T. Kivinen
AuthenTec
October 1, 2012

Minimal IKEv2
draft-kivinen-ipsecme-ikev2-minimal-01.txt

Abstract

This document describes minimal version of the Internet Key Exchange version 2 (IKEv2) protocol. IKEv2 is a component of IPsec used for performing mutual authentication and establishing and maintaining Security Associations (SAs). IKEv2 includes several optional features, which are not needed in minimal implementations. This document describes what is required from the minimal implementation, and also describes various optimizations which can be done. The protocol described here is compliant with full IKEv2 with exception that this document only describes shared secret authentication (IKEv2 requires support for certificate authentication in addition to shared secret authentication).

This document does not update or modify RFC 5996, but provides more compact description of the minimal version of the protocol. If this document and RFC 5996 conflicts then RFC 5996 is the authoritative description.

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 April 4, 2013.

Copyright Notice

Copyright (c) 2012 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
1.1. Use Cases	4
2. Exchanges	6
2.1. Initial Exchange	6
2.2. Other Exchanges	11
2.3. Generating Keying Material	12
3. Conformance Requirements	14
4. Security Considerations	15
5. IANA Considerations	16
6. Acknowledgements	17
7. References	18
7.1. Normative References	18
7.2. Informative References	18
Appendix A. Header and Payload Formats	19
A.1. The IKE Header	19
A.2. Generic Payload Header	21
A.3. Security Association Payload	22
A.3.1. Proposal Substructure	24
A.3.2. Transform Substructure	25
A.3.3. Valid Transform Types by Protocol	27
A.3.4. Transform Attributes	28
A.4. Key Exchange Payload	28
A.5. Identification Payloads	29
A.6. Certificate Payload	30
A.7. Certificate Request Payload	31
A.8. Authentication Payload	32
A.9. Nonce Payload	33
A.10. Notify Payload	33
A.10.1. Notify Message Types	34
A.11. Traffic Selector Payload	35
A.11.1. Traffic Selector	37
A.12. Encrypted Payload	38
Appendix B. Useful Optional Features	41
B.1. IKE SA Delete Notification	41
B.2. Raw RSA keys	42
Author's Address	44

1. Introduction

This document tells what minimal IKEv2 implementation could look like. Minimal IKEv2 implementation only supports initiator end of the protocol. It only supports the initial IKE_SA_INIT and IKE_AUTH exchanges and does not initiate any other exchanges. It also replies with empty (or error) message to all incoming requests.

This means that most of the optional features of IKEv2 are left out: NAT Traversal, IKE SA rekey, Child SA Rekey, Multiple Child SAs, Deleting Child / IKE SAs, Configuration payloads, EAP authentication, COOKIES etc.

Some optimizations can be done because of limited set of supported features, and this text should not be considered for generic IKEv2 implementations (for example Message IDs can be done as specified as implementation is only sending out IKE_SA_INIT and IKE_AUTH request, and do not ever send any other request).

This document should be stand-alone, meaning everything needed to implement IKEv2 is copied here except the description of the cryptographic algorithms. The IKEv2 specification has lots of background information and rationale which has been omitted from this document.

Numerous additional numeric values from IANA registries have been omitted from this document, only those which are of interest for minimal implementation are listed in this document.

For more information check the full IKEv2 specification in RFC 5996 [RFC5996] and [IKEV2IANA].

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].

1.1. Use Cases

One use case for this kind of minimal implementation is in small devices doing machine to machine communication. In such environments the node initiating connections is usually very small and the other end of the communication channel is some kind of larger device.

An example of the small initiating node could be an remote garage door opener device. I.e. device having buttons which open and close garage door, and which connects to the home area network server over wireless link.

Another example of the such device is some kind of sensor device, for example room temperature sensor, which sends periodic temperature data to some centralized node.

Those devices are usually sleeping long times, and only wakes up because of user interaction or periodically. The data transfer is always initiated from the sleeping node and after they send packets there might be ACKs or other packets coming back before they go back to sleep. If some data needs to be transferred from server node to the small device, it can be implemented by polling, i.e. small node periodically polls for the server to see if it for example have some configuration changes or similar.

2. Exchanges

2.1. Initial Exchange

All IKEv2 communications consist of pairs of messages: a request and a response. The pair is called an "exchange", and is sometimes called a "request/response pair". Every request requires a response.

For every pair of IKEv2 messages, the initiator is responsible for retransmission in the event of a timeout. The responder **MUST** never retransmit a response unless it receives a retransmission of the request.

IKEv2 is a reliable protocol: the initiator **MUST** retransmit a request until it either receives a corresponding response or deems the IKE SA to have failed. A retransmission from the initiator **MUST** be bitwise identical to the original request. Retransmission times **MUST** increase exponentially.

IKEv2 is run over UDP port 500. All IKEv2 implementations **MUST** be able to send, receive, and process IKEv2 messages that are up to 1280 octets long. An implementation **MUST** accept incoming requests even if the source port is not 500, and **MUST** respond to the address and port from which the request was received.

The minimal implementation of IKEv2 only uses first two exchanges called `IKE_SA_INIT` and `IKE_AUTH`. Those are used to create the IKE SA and the first child SA. In addition to those messages minimal IKEv2 implementation need to understand `CREATE_CHILD_SA` request so it can reply with `CREATE_CHILD_SA` error response saying `NO_ADDITIONAL_SAS` to it, and understand `INFORMATIONAL` request so much, it can reply with empty `INFORMATIONAL` response to it. There is no requirement to be able to respond to any other requests.

All messages following the `IKE_SA_INIT` exchange are cryptographically protected using the cryptographic algorithms and keys negotiated in the `IKE_SA_INIT` exchange.

Every IKEv2 message contains a Message ID as part of its fixed header. This Message ID is used to match up requests and responses, and to identify retransmissions of messages.

Minimal implementation need only support of being initiator, so it does not ever need to send any other request as one `IKE_SA_INIT`, and one `IKE_AUTH` message. As those messages have fixed Message IDs (0 and 1) it does not need to keep track of its own Message IDs for outgoing requests after that.

Minimal implementations can also optimize Message ID handling of the incoming requests, as they do not need to protect incoming requests against replays. This is possible because minimal implementation will only return error or empty notifications replies to incoming requests. This means that any of those incoming requests do not have any effect on the minimal implementation, thus processing them again does not cause any harm. Because of this the minimal implementation can always answer to request coming in, with the same Message ID than what the request had and then forget the request/response pair immediately. This means there is no need to keep any kind of track of Message IDs of the incoming requests.

In the following descriptions, the payloads contained in the message are indicated by names as listed below.

Notation	Payload

AUTH	Authentication
CERTREQ	Certificate Request
D	Delete
HDR	IKE header (not a payload)
IDi	Identification - Initiator
IDr	Identification - Responder
KE	Key Exchange
Ni, Nr	Nonce
N	Notify
SA	Security Association
SK	Encrypted and Authenticated
TSi	Traffic Selector - Initiator
TSr	Traffic Selector - Responder

The initial exchanges are as follows:

Initiator	Responder

HDR(SPIi=xxx, SPIr=0, IKE_SA_INIT, Flags: Initiator, Message ID=0), SAil, KEi, Ni -->	<-- HDR(SPIi=xxx, SPIr=yyy, IKE_SA_INIT, Flags: Response, Message ID=0), SArl, KEr, Nr, [CERTREQ]

HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. Each endpoint chooses one of the two SPIs and MUST choose them so as to be unique identifiers of an IKE SA. An SPI value of zero is special: it indicates that the remote SPI value is not yet known by the sender.

Incoming IKEv2 packets are mapped to an IKE SA only using the packet's SPI, not using (for example) the source IP address of the packet.

The SAil payload states the cryptographic algorithms the initiator supports for the IKE SA. The KEi and KEr payload contain Diffie-Hellman values and Ni and Nr are the nonces. The SARl contains chosen cryptographic suite from initiator's offered choices. Minimal implementation using shared secrets will ignore the CERTREQ payload.

Minimal implementation will most likely support exactly one set of cryptographic algorithms, meaning the SAil payload will be static. It needs to check that the SARl received matches the proposal it sent.

At this point in the negotiation, each party can generate SKEYSEED, from which all keys are derived for that IKE SA.

$$\text{SKEYSEED} = \text{prf}(\text{Ni} \parallel \text{Nr}, g^{\text{ir}})$$

$$\begin{aligned} \{ \text{SK}_d \parallel \text{SK}_{ai} \parallel \text{SK}_{ar} \parallel \text{SK}_{ei} \parallel \text{SK}_{er} \parallel \text{SK}_{pi} \parallel \text{SK}_{pr} \} \\ = \text{prf}^+ (\text{SKEYSEED}, \text{Ni} \parallel \text{Nr} \parallel \text{SPIi} \parallel \text{SPIr}) \end{aligned}$$

$$\text{prf}^+ (K, S) = T1 \parallel T2 \parallel T3 \parallel T4 \parallel \dots$$

where:

$$\begin{aligned} T1 &= \text{prf} (K, S \parallel 0x01) \\ T2 &= \text{prf} (K, T1 \parallel S \parallel 0x02) \\ T3 &= \text{prf} (K, T2 \parallel S \parallel 0x03) \\ T4 &= \text{prf} (K, T3 \parallel S \parallel 0x04) \\ &\dots \end{aligned}$$

(indicating that the quantities SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi, and SK_pr are taken in order from the generated bits of the prf+). g^{ir} is the shared secret from the ephemeral Diffie-Hellman exchange. g^{ir} is represented as a string of octets in big endian order padded with zeros if necessary to make it the length of the modulus. Ni and Nr are the nonces, stripped of any headers.

The SK_d is used for deriving new keys for the Child SAs. The SK_ai and SK_ar are used as a key to the integrity protection algorithm for authenticating the component messages of subsequent exchanges. The SK_ei and SK_er are used for encrypting (and of course decrypting) all subsequent exchanges. The SK_pi and SK_pr are used when generating an AUTH payload. The lengths of SK_d, SK_pi, and SK_pr MUST be the preferred key length of the PRF agreed upon.

A separate SK_e and SK_a is computed for each direction. The keys

used to protect messages from the original initiator are SK_ai and SK_ei. The keys used to protect messages in the other direction are SK_ar and SK_er. The notation SK { ... } indicates that these payloads are encrypted and integrity protected using that direction's SK_e and SK_a.

Initiator

Responder

```
-----
HDR(SPIi=xxx, SPIr=yyy, IKE_AUTH,
  Flags: Initiator, Message ID=1),
SK {IDi, AUTH, SAi2, TSi, TSr,
  N(INITIAL_CONTACT)} -->
```

```
<-- HDR(SPIi=xxx, SPIr=yyy, IKE_AUTH, Flags:
      Response, Message ID=1),
      SK {IDr, AUTH, SAr2, TSi, TSr}
```

The initiator asserts its identity with the IDi payload, proves knowledge of the secret corresponding to IDi and integrity protects the contents of the first message using the AUTH payload. The responder asserts its identity with the IDr payload, authenticates its identity and protects the integrity of the second message with the AUTH payload.

As minimal implementation usually has only one host where it connects, and that means it has only one shared secret. This means it does not need to care about IDr payload that much. If the other end sends AUTH payload which initiator can verify using the shared secret it has, then it knows the other end is the peer it was configured to talk to.

In the IKE_AUTH initiator sends SA offer(s) in the SAi2 payload, and the proposed Traffic Selectors for the proposed Child SA in the TSi and TSr payloads. The responder replies with the accepted offer in an SAr2 payload, and selected Traffic Selectors. The selected Traffic Selectors may be a subset of what the initiator proposed.

In the minimal implementation both SA payloads and TS payloads are going to be mostly static. The SA payload will have the SPI value used in the ESP, but the algorithms are most likely going to be the one and only supported set. The TS payloads on the initiator end will most likely say from any to any, i.e. full wildcard ranges, or from the local IP to the remote IP. In the wildcard case the server quite often narrow the range down to the one IP address pair. Using single IP address pair as a traffic selectors when sending IKE_AUTH will simplify processing as then server will either accept that pair or return error. If wildcard ranges are used, there is possibility that server narrows the range to some other range than what was

intended.

The IKE_AUTH (and IKE_SA_INIT) responses may contain multiple status notification payloads which can be ignored by minimal implementation. There can also be Vendor ID, Certificate, Certificate Request or Configuration payloads, but any payload unknown to minimal implementation can simply be skipped over (response messages cannot have critical unsupported payloads).

The exchange above includes N(INITIAL_CONTACT) notification in the request as that is quite commonly sent by the minimal implementation. It indicates to the other end that the initiator does not have any other IKE SAs between them, and if there is any left from previous runs they can be deleted. As minimal implementation does not delete IKE SAs by sending IKE SA delete, this will help server to clean up leftover state.

When using shared secret authentication, the peers are authenticated by having each calculating a MAC over a block of data:

For the initiator:

```
AUTH = prf( prf(Shared Secret, "Key Pad for IKEv2"),  
            <InitiatorSignedOctets>)
```

For the responder:

```
AUTH = prf( prf(Shared Secret, "Key Pad for IKEv2"),  
            <ResponderSignedOctets>)
```

The string "Key Pad for IKEv2" is 17 ASCII characters without null termination. The implementation can precalculate the inner prf and only store the output of it. This is possible because minimal IKEv2 implementation usually only supports one PRF.

The initiator signs the first message (IKE_SA_INIT request), starting with the first octet of the first SPI in the header and ending with the last octet of the last payload in that first message. Appended to this (for purposes of computing the signature) are the responder's nonce Nr, and the value prf(SK_pi, IDi').

For the responder, the octets to be signed start with the first octet of the first SPI in the header of the second message (IKE_SA_INIT response) and end with the last octet of the last payload in that second message. Appended to this are the initiator's nonce Ni, and the value prf(SK_pr, IDr').

In these calculations, IDi' and IDr' are the entire ID payloads excluding the fixed header and the Ni, and Nr are only the value, not the payload containing it. Note that neither the nonce Ni/Nr nor the value prf(SK_pr, IDr')/prf(SK_pi, IDi') are transmitted.

The initiator's signed octets can be described as:

```
InitiatorSignedOctets = RealMessage1 | NonceRData | MACedIDForI
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR = SPIi | SPIr | . . . | Length
RealMessage1 = RealIKEHDR | RestOfMessage1
NonceRPayload = PayloadHeader | NonceRData
InitiatorIDPayload = PayloadHeader | RestOfInitIDPayload
RestOfInitIDPayload = IDType | RESERVED | InitIDData
MACedIDForI = prf(SK_pi, RestOfInitIDPayload)
```

The responder's signed octets can be described as:

```
ResponderSignedOctets = RealMessage2 | NonceIData | MACedIDForR
GenIKEHDR = [ four octets 0 if using port 4500 ] | RealIKEHDR
RealIKEHDR = SPIi | SPIr | . . . | Length
RealMessage2 = RealIKEHDR | RestOfMessage2
NonceIPayload = PayloadHeader | NonceIData
ResponderIDPayload = PayloadHeader | RestOfRespIDPayload
RestOfRespIDPayload = IDType | RESERVED | RespIDData
MACedIDForR = prf(SK_pr, RestOfRespIDPayload)
```

Note that all of the payloads inside the RestOfMessageX are included under the signature, including any payload types not listed in this document.

The initiator might also get unauthenticated response back having notification payload with error code inside. As that error code will be unauthenticated and may be faked, there is no need to do anything for those. Minimal implementation can simply ignore those errors, and retransmit its request until it times out and if that happens then the IKE SA (and Child SA) creation failed.

Responder might also reply with IKE_AUTH response packet which do not contain payloads needed to set up Child SA (SAr2, TSi and TSr), but contains AUTH payload and an error. As minimal implementation probably do not support multiple SAs nor sending the CREATE_CHILD_SA exchanges the IKE SA is useless for initiator. It can delete the IKE SA and start over from the beginning (which might fail again if this is configuration error, or it might succeed if this was temporal failure).

2.2. Other Exchanges

Minimal implementation MUST be able to reply to INFORMATIONAL request by sending empty response back:

Initiator	Responder

	<-- HDR(SPIi=xxx, SPIr=yyy, INFORMATIONAL, Flags: none, Message ID=m), SK {...}
HDR(SPIi=xxx, SPIr=yyy, INFORMATIONAL, Flags: Initiator Response, Message ID=m), SK {} -->	

Minimal implementation also MUST be able to reply to incoming CREATE_CHILD_SA requests. Typical implementation will reject the CREATE_CHILD_SA exchanges by sending NO_ADDITIONAL_SAS error notify back:

Initiator	Responder

	<-- HDR(SPIi=xxx, SPIr=yyy, CREATE_CHILD_SA, Flags: none, Message ID=m), SK {...}
HDR(SPIi=xxx, SPIr=yyy, CREATE_CHILD_SA, Flags: Initiator Response, Message ID=m), SK {N(NO_ADDITIONAL_SAS)}) -->	

Note, that INFORMATIONAL and CREATE_CHILD_SA requests might contain unsupported critical payloads, in which case compliant implementation MUST ignore the request, and send response message back having the UNSUPPORTED_CRITICAL_PAYLOAD notification. That notification payload data contains one-octet payload type of the unsupported critical payload.

2.3. Generating Keying Material

Keying material for Child SA created by the IKE_AUTH exchange is generated as follows:

KEYMAT = prf+(SK_d, Ni | Nr)

Where Ni and Nr are the nonces from the IKE_SA_INIT exchange.

A single CHILD_SA negotiation may result in multiple Security Associations. ESP and AH SAs exist in pairs (one in each direction), so two SAs are created in a single Child SA negotiation for them. The keying material for each Child SA MUST be taken from the expanded KEYMAT using the following rules:

- o All keys for SAs carrying data from the initiator to the responder are taken before SAs going from the responder to the initiator.
- o If an IPsec protocol requires multiple keys, the order in which they are taken from the SA's keying material needs to be described in the protocol's specification. For ESP and AH, [IPSECARCH] defines the order, namely: the encryption key (if any) MUST be taken from the first bits and the integrity key (if any) MUST be taken from the remaining bits.

Each cryptographic algorithm takes a fixed number of bits of keying material specified as part of the algorithm, or negotiated in SA payloads.

3. Conformance Requirements

For an implementation to be called conforming to RFC 5996 specification, it MUST be possible to configure it to accept the following:

- o Public Key Infrastructure using X.509 (PKIX) Certificates containing and signed by RSA keys of size 1024 or 2048 bits, where the ID passed is any of ID_KEY_ID, ID_FQDN, ID_RFC822_ADDR, or ID_DER_ASN1_DN.
- o Shared key authentication where the ID passed is any of ID_KEY_ID, ID_FQDN, or ID_RFC822_ADDR.
- o Authentication where the responder is authenticated using PKIX Certificates and the initiator is authenticated using shared key authentication.

This document only supports the second bullet, it does not support PKIX certificates at all. As full RFC5996 responders must also support that shared key authentication, this allows minimal implementation to be able to interoperate with all RFC 5996 compliant implementations.

PKIX certificates are left out from the minimal implementation as those would add quite a lot of complexity to the implementation. The actual code changes needed in the IKEv2 protocol are small, but the certificate validation code would be more complex than the whole minimal IKEv2 implementation itself. If public key based authentication is needed for scalability reasons, then raw RSA keys would probably be the best compromise (see Appendix B.2).

4. Security Considerations

As this implements same protocol as RFC 5996 this means all security considerations from it also apply to this document.

5. IANA Considerations

There is no new IANA considerations in this document.

6. Acknowledgements

Most of the contents of this document is copied from the RFC 5996.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

7.2. Informative References

- [IKEV2IANA] "Internet Key Exchange Version 2 (IKEv2) Parameters", <<http://www.iana.org>>.
- [MODES] National Institute of Standards and Technology, U.S. Department of Commerce, "Recommendation for Block Cipher Modes of Operation", SP 800-38A, 2001.
- [RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RSA] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", February 1978.

Appendix A. Header and Payload Formats

This appendix describes actual packet payload formats. This is required to make the document self contained. The descriptions are mostly copied from the RFC5996 and more information can be found from there.

Various payload contains RESERVED fields and those MUST be sent as zero and MUST be ignored on receipt.

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

A.1. The IKE Header

Each IKEv2 message begins with the IKE header, denoted HDR in this document. Following the header are one or more IKE payloads each identified by a "Next Payload" field in the preceding payload. Payloads are identified in the order in which they appear in an IKE message by looking in the "Next Payload" field in the IKE header, and subsequently according to the "Next Payload" field in the IKE payload itself until a "Next Payload" field of zero indicates that no payloads follow. If a payload of type "Encrypted" is found, that payload is decrypted and its contents parsed as additional payloads. An Encrypted payload MUST be the last payload in a packet and an Encrypted payload MUST NOT contain another Encrypted payload.

The format of the IKE header is shown in Figure 1.

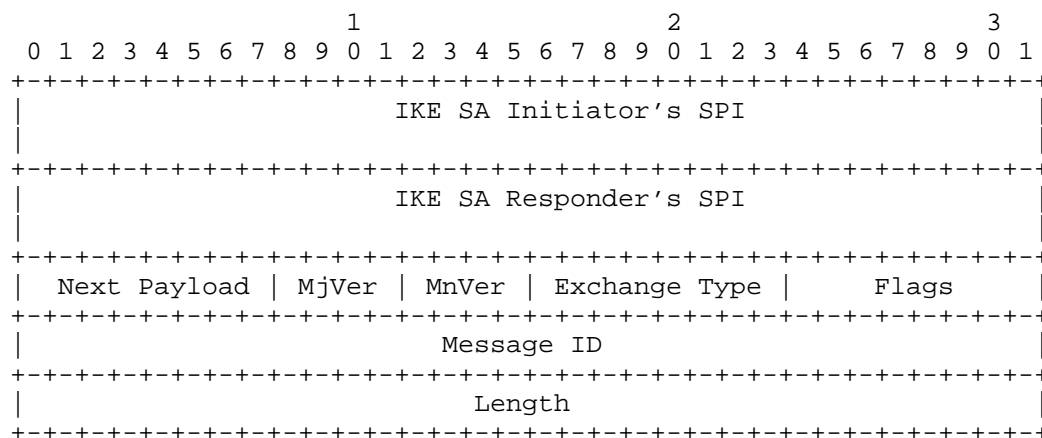


Figure 1: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE Security Association. This value MUST NOT be zero.
- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE Security Association. This value MUST be zero in the first message of an IKE initial exchange.
- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.
- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the major version to 2 and MUST drop the messages with a higher major version number.
- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the minor version to 0. They MUST ignore the minor version number of received messages.
- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message in an exchange.

Exchange Type	Value
-----	-----
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37

- o Flags (1 octet) - Indicates specific options that are set for the message. Presence of options is indicated by the appropriate bit in the flags field being set. The bits are as follows:

```

+---+---+---+---+---+---+
|X|X|R|V|I|X|X|X|
+---+---+---+---+---+---+

```

In the description below, a bit being 'set' means its value is '1', while 'cleared' means its value is '0'. 'X' bits MUST be cleared when sending and MUST be ignored on receipt.

- * R (Response) - This bit indicates that this message is a response to a message containing the same Message ID. This bit MUST be cleared in all request messages and MUST be set in all

responses. An IKEv2 endpoint MUST NOT generate a response to a message that is marked as being a response.

- * V (Version) - This bit indicates that the transmitter is capable of speaking a higher major version number of the protocol than the one indicated in the major version number field. Implementations of IKEv2 MUST clear this bit when sending and MUST ignore it in incoming messages.
- * I (Initiator) - This bit MUST be set in messages sent by the original initiator of the IKE SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient. This bit changes to reflect who initiated the last rekey of the IKE SA.
- o Message ID (4 octets, unsigned integer) - Message identifier used to control retransmission of lost packets and matching of requests and responses. It is essential to the security of the protocol because it is used to prevent message replay attacks.
- o Length (4 octets, unsigned integer) - Length of the total message (header + payloads) in octets.

A.2. Generic Payload Header

Each IKE payload begins with a generic payload header, shown in Figure 2. Figures for each payload below will include the generic payload header, but for brevity, the description of each field will be omitted.

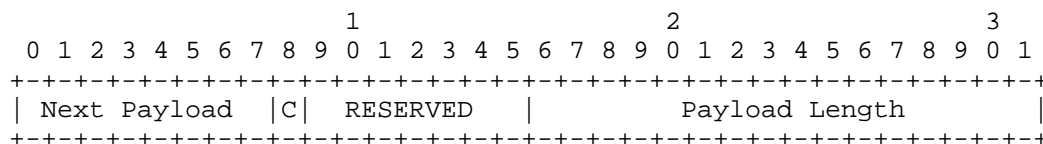


Figure 2: Generic Payload Header

The Generic Payload Header 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 0. This field provides a "chaining" capability whereby additional payloads can be added to a message by appending each one to the end of the message and setting the "Next Payload" field of the preceding payload to indicate the new payload's type. An Encrypted payload, which must

always be the last payload of a message, is an exception. It contains data structures in the format of additional payloads. In the header of an Encrypted payload, the Next Payload field is set to the payload type of the first contained payload (instead of 0); conversely, the Next Payload field of the last contained payload is set to zero). The payload type values needed for minimal implementations are listed here.

Next Payload Type	Notation	Value
No Next Payload		0
Security Association	SA	33
Key Exchange	KE	34
Identification - Initiator	IDi	35
Identification - Responder	IDr	36
Certificate	CERT	37
Certificate Request	CERTREQ	38
Authentication	AUTH	39
Nonce	Ni, Nr	40
Notify	N	41
Delete	D	42
Traffic Selector - Initiator	TSi	44
Traffic Selector - Responder	TSr	45
Encrypted and Authenticated	SK	46

- o Critical (1 bit) - MUST be set to zero if the sender wants the recipient to skip this payload if it does not understand the payload type code in the Next Payload field of the previous payload. MUST be set to one if the sender wants the recipient to reject this entire message if it does not understand the payload type. MUST be ignored by the recipient if the recipient understands the payload type code. MUST be set to zero for payload types defined in this document. Note that the critical bit applies to the current payload rather than the "next" payload whose type code appears in the first octet.
- o Payload Length (2 octets, unsigned integer) - Length in octets of the current payload, including the generic payload header.

A.3. Security Association Payload

The Security Association payload, denoted SA in this document, is used to negotiate attributes of a Security Association.

An SA payload consists of one or more proposals. Each proposal includes one protocol. Each protocol contains one or more transforms -- each specifying a cryptographic algorithm. Each transform contains zero or more attributes (attributes are needed only if the

Transform ID does not completely specify the cryptographic algorithm, currently only attribute is key length attribute for variable length ciphers, meaning there is exactly zero or one attribute).

The responder MUST choose a single suite, which may be any subset of the SA proposal following the rules below.

Each proposal contains one protocol. If a proposal is accepted, the SA response MUST contain the same protocol. Each IPsec protocol proposal contains one or more transforms. Each transform contains a Transform Type. The accepted cryptographic suite MUST contain exactly one transform of each type included in the proposal. For example: if an ESP proposal includes transforms ENCR_3DES, ENCR_AES w/keysize 128, ENCR_AES w/keysize 256, AUTH_HMAC_MD5, and AUTH_HMAC_SHA, the accepted suite MUST contain one of the ENCR_ transforms and one of the AUTH_ transforms. Thus, six combinations are acceptable.

Minimal implementation can create very simple SA proposal, i.e. include one proposal, which contains exactly one transform for each transform type. It is important to only include one Diffie-Hellman group in proposal, so there is no need to do INVALID_KEY_PAYLOAD processing in responses.

When parsing an SA, an implementation MUST check that the total Payload Length is consistent with the payload's internal lengths and counts. Proposals, Transforms, and Attributes each have their own variable-length encodings. They are nested such that the Payload Length of an SA includes the combined contents of the SA, Proposal, Transform, and Attribute information. The length of a Proposal includes the lengths of all Transforms and Attributes it contains. The length of a Transform includes the lengths of all Attributes it contains.

Each Proposal/Protocol structure is followed by one or more transform structures. The number of different transforms is generally determined by the Protocol. AH generally has two transforms: Extended Sequence Numbers (ESNs) and an integrity check algorithm. ESP generally has three: ESN, an encryption algorithm, and an integrity check algorithm. IKEv2 generally has four transforms: a Diffie-Hellman group, an integrity check algorithm, a PRF algorithm, and an encryption algorithm. For each Protocol, the set of permissible transforms is assigned Transform ID numbers, which appear in the header of each transform.

If there are multiple transforms with the same Transform Type, the proposal is an OR of those transforms. If there are multiple transforms with different Transform Types, the proposal is an AND of

the different groups.

A given transform MAY have one or more Attributes. Attributes are necessary when the transform can be used in more than one way, as when an encryption algorithm has a variable key size. The transform would specify the algorithm and the attribute would specify the key size. To propose alternate values for an attribute (for example, multiple key sizes for the AES encryption algorithm), an implementation MUST include multiple transforms with the same Transform Type each with a single Attribute.

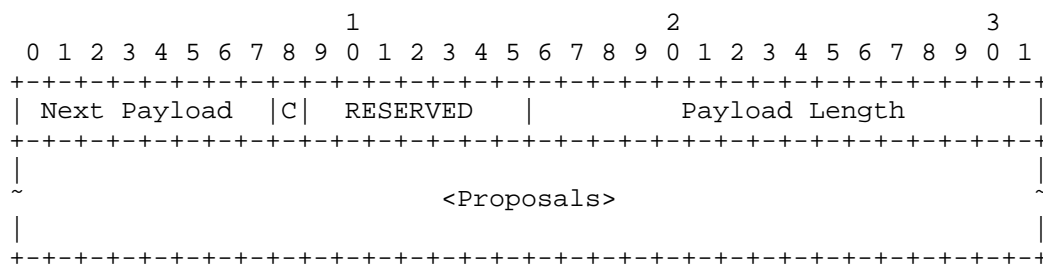


Figure 3: Security Association Payload

- o Proposals (variable) - One or more proposal substructures.

A.3.1. Proposal Substructure

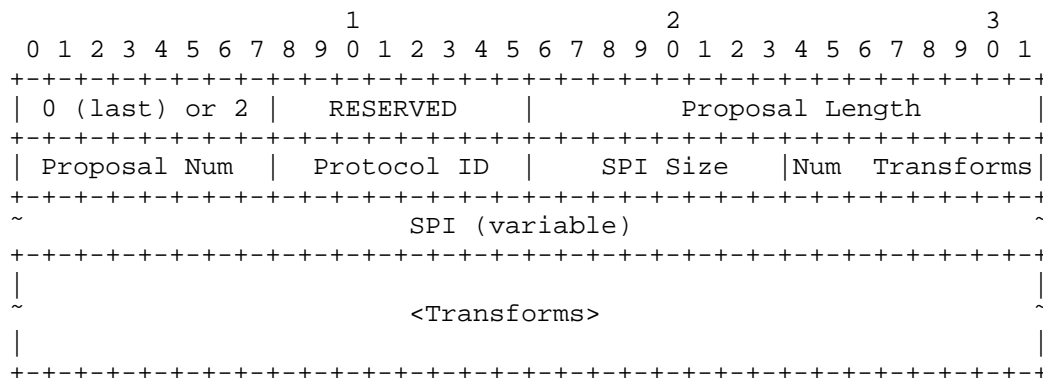


Figure 4: Proposal Substructure

- o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA.
- o Proposal Length (2 octets, unsigned integer) - Length of this proposal, including all transforms and attributes that follow.

- o Proposal Num (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be 1, and subsequent proposals MUST be one more than the previous proposal. When a proposal is accepted, the proposal number in the SA payload MUST match the number on the proposal sent that was accepted.
- o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation.

Protocol	Protocol ID
-----	-----
IKE	1
AH	2
ESP	3

- o SPI Size (1 octet) - For an initial IKE SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).
- o Num Transforms (1 octet) - Specifies the number of transforms in this proposal.
- o SPI (variable) - The sending entity's SPI. When the SPI Size field is zero, this field is not present in the Security Association payload.
- o Transforms (variable) - One or more transform substructures.

A.3.2. Transform Substructure

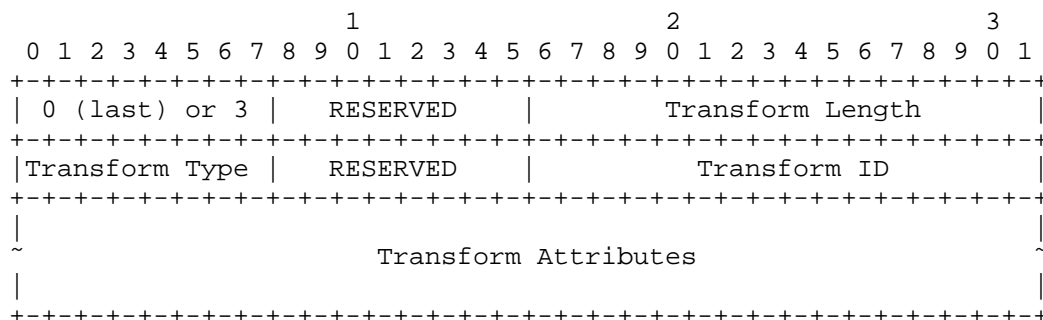


Figure 5: Transform Substructure

- o 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal.
- o Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.
- o Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different Transform Types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with Transform ID = 0 as one of the options.
- o Transform ID (2 octets) - The specific instance of the Transform Type being proposed.

The relevant Transform Type values are listed below.

Description	Trans. Type	Used In
Encryption Algorithm (ENCR)	1	IKE and ESP
Pseudorandom Function (PRF)	2	IKE
Integrity Algorithm (INTEG)	3	IKE, AH, optional in ESP
Diffie-Hellman group (D-H)	4	IKE, optional in AH & ESP
Extended Sequence Numbers (ESN)	5	AH and ESP

For Transform Type 1 (Encryption Algorithm), the relevant Transform IDs are listed below.

Name	Number	Defined In
ENCR_3DES	3	(RFC2451)
ENCR_AES_CBC	12	(RFC3602)

For Transform Type 2 (Pseudorandom Function), the relevant Transform IDs are listed below.

Name	Number	Defined In
PRF_HMAC_MD5	1	(RFC2104), [MD5]
PRF_HMAC_SHA1	2	(RFC2104), [SHA]

For Transform Type 3 (Integrity Algorithm), relevant Transform IDs are listed below.

Name	Number	Defined In
NONE	0	
AUTH_HMAC_MD5_96	1	(RFC2403)
AUTH_HMAC_SHA1_96	2	(RFC2404)
AUTH_AES_XCBC_96	5	(RFC3566)

For Transform Type 4 (Diffie-Hellman group), relevant Transform IDs are listed below.

Name	Number	Defined In
NONE	0	
768-bit MODP	1	Appendix B
1024-bit MODP	2	Appendix B
1536-bit MODP	5	[ADDGROUP]
2048-bit MODP	14	[ADDGROUP]

For Transform Type 5 (Extended Sequence Numbers), relevant Transform IDs are listed below.

Name	Number
No Extended Sequence Numbers	0
Extended Sequence Numbers	1

Note that an initiator who supports ESNs will usually include two ESN transforms, with values "0" and "1", in its proposals. A proposal containing a single ESN transform with value "1" means that using normal (non-extended) sequence numbers is not acceptable.

A.3.3. Valid Transform Types by Protocol

The number and type of transforms that accompany an SA payload are dependent on the protocol in the SA itself. An SA payload proposing the establishment of an SA has the following mandatory and optional Transform Types. A compliant implementation MUST understand all mandatory and optional types for each protocol it supports (though it need not accept proposals with unacceptable suites). A proposal MAY omit the optional types if the only value for them it will accept is NONE.

Protocol	Mandatory Types	Optional Types
IKE	ENCR, PRF, INTEG, D-H	
ESP	ENCR, ESN	INTEG, D-H
AH	INTEG, ESN	D-H

A.3.4. Transform Attributes

Transform type 1 (Encryption Algorithm) transforms might include one transform attribute: Key Length.

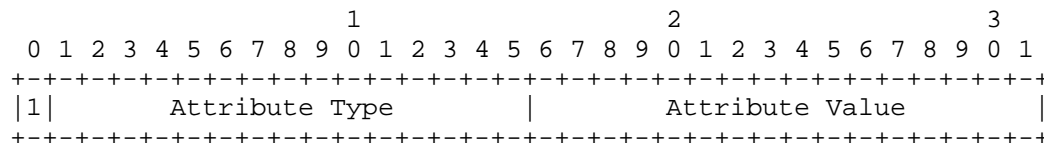


Figure 6: Data Attributes

- o Attribute Type (15 bits) - Unique identifier for each type of attribute (see below).
- o Attribute Value - Value of the attribute associated with the attribute type.

Attribute Type	Value
-----	-----
Key Length (in bits)	14

The Key Length attribute specifies the key length in bits (MUST use network byte order) for certain transforms as follows:

- o The Key Length attribute MUST NOT be used with transforms that use a fixed-length key.
- o Some transforms specify that the Key Length attribute MUST be always included. For example ENCR_AES_CBC.

A.4. Key Exchange Payload

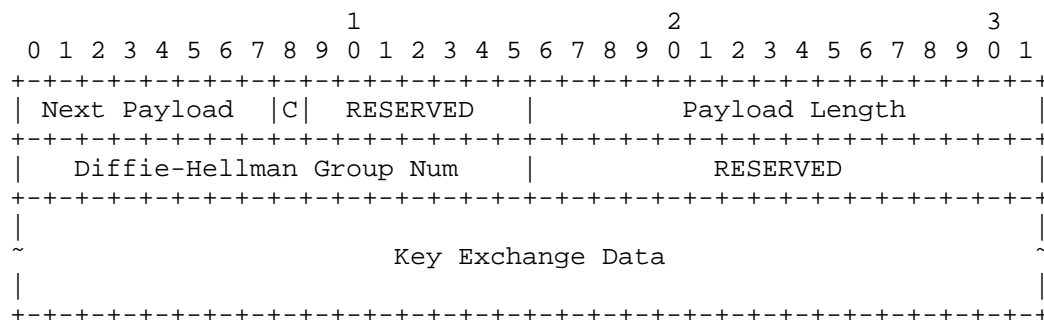


Figure 7: Key Exchange Payload Format

A Key Exchange payload is constructed by copying one's Diffie-Hellman public value into the "Key Exchange Data" portion of the payload. The length of the Diffie-Hellman public value for modular exponentiation group (MODP) groups MUST be equal to the length of the prime modulus over which the exponentiation was performed, prepending zero bits to the value if necessary.

The Diffie-Hellman Group Num identifies the Diffie-Hellman group in which the Key Exchange Data was computed. This Diffie-Hellman Group Num MUST match a Diffie-Hellman group specified in a proposal in the SA payload that is sent in the same message

A.5. Identification Payloads

The Identification payloads, denoted IDi and IDr in this document, allow peers to assert an identity to one another. When using the ID_IPV4_ADDR/ID_IPV6_ADDR identity types in IDi/IDr payloads, IKEv2 does not require this address to match the address in the IP header of IKEv2 packets, or anything in the TSi/TSr payloads. The contents of IDi/IDr are used purely to fetch the policy and authentication data related to the other party. In minimal implementation it might be easiest to always use KEY_ID type. This allows the ID payload to be static. Using IP address has problems in environments where IP addresses are dynamically allocated.

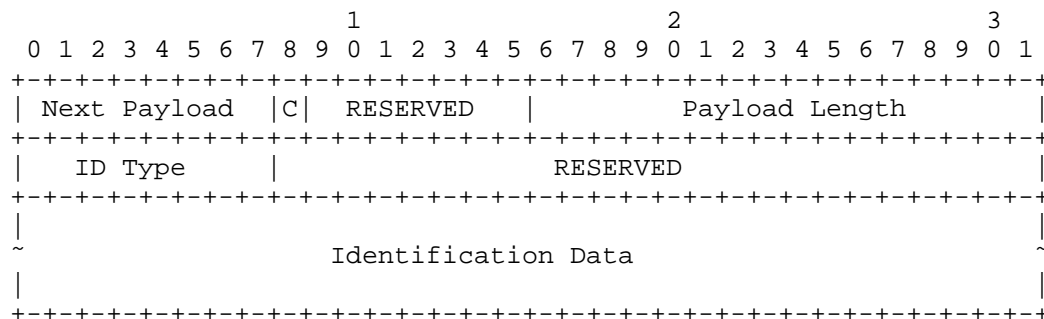


Figure 8: Identification Payload Format

- o ID Type (1 octet) - Specifies the type of Identification being used.
- o Identification Data (variable length) - Value, as indicated by the Identification Type. The length of the Identification Data is computed from the size in the ID payload header.

The following table lists the assigned semantics for the Identification Type field.

ID Type	Value

ID_IPV4_ADDR	1
A single four (4) octet IPv4 address.	
ID_FQDN	2
A fully-qualified domain name string. An example of an ID_FQDN is "example.com". The string MUST NOT contain any terminators (e.g., NULL, CR, etc.). All characters in the ID_FQDN are ASCII; for an "internationalized domain name", the syntax is as defined in [IDNA], for example "xn--tmonesimerkki-bfbb.example.net".	
ID_RFC822_ADDR	3
A fully-qualified RFC 822 email address string. An example of a ID_RFC822_ADDR is "jsmith@example.com". The string MUST NOT contain any terminators. Because of [EAI], implementations would be wise to treat this field as UTF-8 encoded text, not as pure ASCII.	
ID_IPV6_ADDR	5
A single sixteen (16) octet IPv6 address.	
ID_KEY_ID	11
An opaque octet stream that may be used to pass vendor-specific information necessary to do certain proprietary types of identification. Minimal implementation might use this type to send out serial number or similar device specific unique static identification data for the device.	

A.6. Certificate Payload

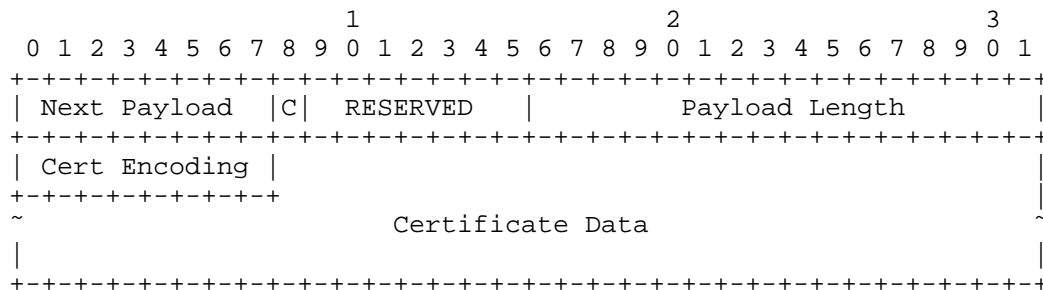


Figure 9: Certificate Payload Format

- o Certificate Encoding (1 octet) - This field indicates the type of certificate or certificate-related information contained in the Certificate Data field.

Certificate Encoding	Value
X.509 Certificate - Signature	4
Raw RSA Key	11

- o Certificate Data (variable length) - Actual encoding of certificate data. The type of certificate is indicated by the Certificate Encoding field.

The syntax of the types above are:

- o "X.509 Certificate - Signature" contains a DER-encoded X.509 certificate whose public key is used to validate the sender's AUTH payload. Note that with this encoding, if a chain of certificates needs to be sent, multiple CERT payloads are used, only the first of which holds the public key used to validate the sender's AUTH payload.
- o "Raw RSA Key" contains a PKCS #1 encoded RSA key, that is, a DER-encoded RSAPublicKey structure (see [RSA] and [RFC3447]).

A.7. Certificate Request Payload

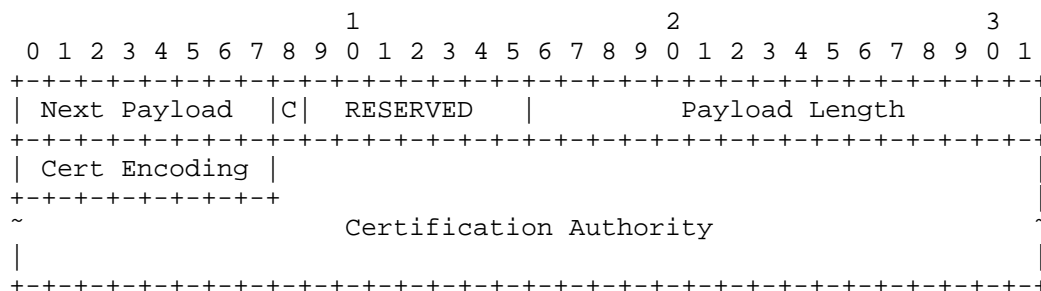


Figure 10: Certificate Request Payload Format

- o Certificate Encoding (1 octet) - Contains an encoding of the type or format of certificate requested.
- o Certification Authority (variable length) - Contains an encoding of an acceptable certification authority for the type of certificate requested.

The Certificate Encoding field has the same values as those defined certificate payload. The Certification Authority field contains an indicator of trusted authorities for this certificate type. The Certification Authority value is a concatenated list of SHA-1 hashes of the public keys of trusted Certification Authorities (CAs). Each

is encoded as the SHA-1 hash of the Subject Public Key Info element (see section 4.1.2.7 of [RFC5280]) from each Trust Anchor certificate. The 20-octet hashes are concatenated and included with no other formatting.

A.8. Authentication Payload

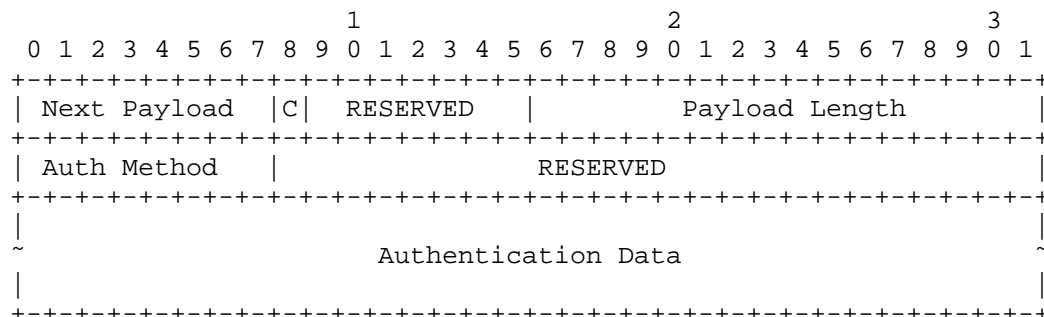


Figure 11: Authentication Payload Format

- o Auth Method (1 octet) - Specifies the method of authentication used.

Mechanism	Value
RSA Digital Signature	1
Using an RSA private key with RSASSA-PKCS1-v1_5 signature scheme specified in [PKCS1], see [RFC5996] section 2.15 for details.	
Shared Key Message Integrity Code	2
Computed as specified earlier using the shared key associated with the identity in the ID payload and the negotiated PRF.	
DSS Digital Signature	3
Using a DSS private key (see [DSS]) over a SHA-1 hash, see [RFC5996] Section 2.15 for details.	

- o Authentication Data (variable length) - see Section 2.1.

A.9. Nonce Payload

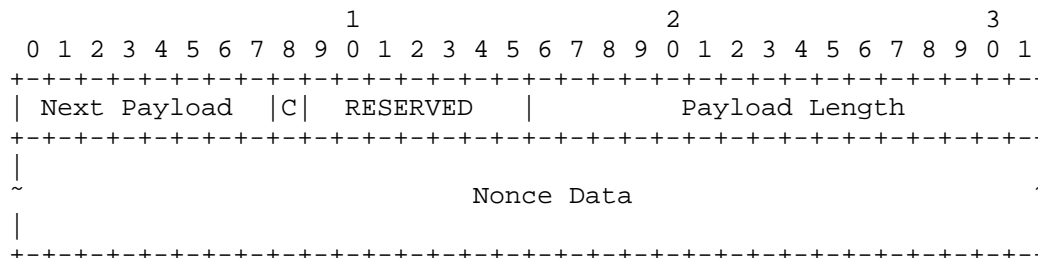


Figure 12: Nonce Payload Format

- o Nonce Data (variable length) - Contains the random data generated by the transmitting entity.

The size of the Nonce Data MUST be between 16 and 256 octets, inclusive. Nonce values MUST NOT be reused.

A.10. Notify Payload

The Notify payload, denoted N in this document, is used to transmit informational data, such as error conditions and state transitions, to an IKE peer. A Notify payload may appear in a response message (usually specifying why a request was rejected), in an INFORMATIONAL Exchange (to report an error not in an IKE request), or in any other message to indicate sender capabilities or to modify the meaning of the request.

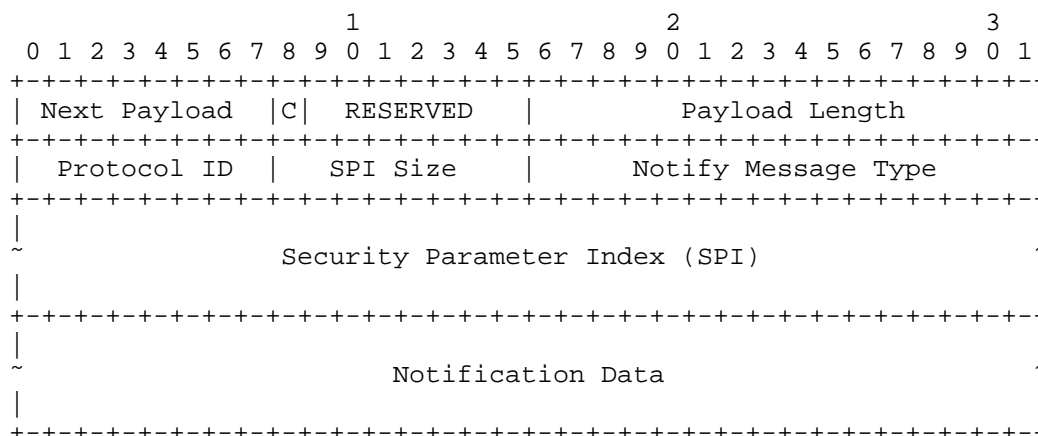


Figure 13: Notify Payload Format

- o Protocol ID (1 octet) - If this notification concerns an existing SA whose SPI is given in the SPI field, this field indicates the type of that SA. If the SPI field is empty, this field MUST be sent as zero and MUST be ignored on receipt.
- o SPI Size (1 octet) - Length in octets of the SPI as defined by the IPsec protocol ID or zero if no SPI is applicable. For a notification concerning the IKE SA, the SPI Size MUST be zero and the field must be empty.
- o Notify Message Type (2 octets) - Specifies the type of notification message.
- o SPI (variable length) - Security Parameter Index.
- o Notification Data (variable length) - Status or error data transmitted in addition to the Notify Message Type. Values for this field are type specific.

A.10.1. Notify Message Types

Notification information can be error messages specifying why an SA could not be established. It can also be status data that a process managing an SA database wishes to communicate with a peer process.

Types in the range 0 - 16383 are intended for reporting errors. An implementation receiving a Notify payload with one of these types that it does not recognize in a response MUST assume that the corresponding request has failed entirely. Unrecognized error types in a request and status types in a request or response MUST be ignored, and they should be logged.

Notify payloads with status types MAY be added to any message and MUST be ignored if not recognized. They are intended to indicate capabilities, and as part of SA negotiation, are used to negotiate non-cryptographic parameters.

NOTIFY messages: error types	Value
UNSUPPORTED_CRITICAL_PAYLOAD	1
Indicates that the one-octet payload type included in the Notification Data field is unknown.	
INVALID_SYNTAX	7
Indicates the IKE message that was received was invalid because some type, length, or value was out of range or because the request was rejected for policy reasons. To avoid a DoS attack using forged messages, this status may only be returned for and in an encrypted packet if the Message ID and cryptographic checksum were valid. To avoid leaking information to someone probing a node, this status MUST be sent in response to any error not covered by one of the other status types. To aid debugging, more detailed error information should be written to a console or log.	
NO_PROPOSAL_CHOSEN	14
None of the proposed crypto suites was acceptable. This can be sent in any case where the offered proposals are not acceptable for the responder.	
NO_ADDITIONAL_SAS	35
Specifies that the node is unwilling to accept any more Child SAs.	

NOTIFY messages: status types	Value
INITIAL_CONTACT	16384
Asserts that this IKE SA is the only IKE SA currently active between the authenticated identities.	

A.11. Traffic Selector Payload

Traffic Selector (TS) payloads allow endpoints to communicate some of the information from their SPD to their peers. TS payloads specify the selection criteria for packets that will be forwarded over the newly set up SA.

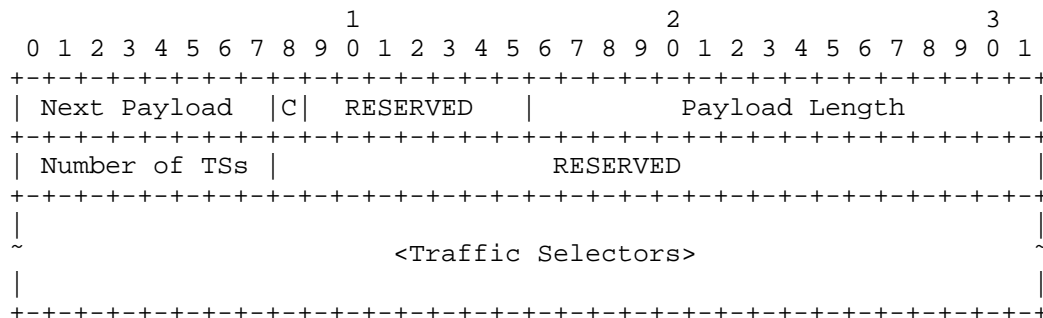


Figure 14: Traffic Selectors Payload Format

- o Number of TSs (1 octet) - Number of Traffic Selectors being provided.
- o Traffic Selectors (variable length) - One or more individual Traffic Selectors.

The length of the Traffic Selector payload includes the TS header and all the Traffic Selectors.

There is no requirement that TSi and TSr contain the same number of individual Traffic Selectors. Thus, they are interpreted as follows: a packet matches a given TSi/TSr if it matches at least one of the individual selectors in TSi, and at least one of the individual selectors in TSr.

Two TS payloads appear in each of the messages in the exchange that creates a Child SA pair. Each TS payload contains one or more Traffic Selectors. Each Traffic Selector consists of an address range (IPv4 or IPv6), a port range, and an IP protocol ID.

The first of the two TS payloads is known as TSi (Traffic Selector-initiator). The second is known as TSr (Traffic Selector-responder). TSi specifies the source address of traffic forwarded from (or the destination address of traffic forwarded to) the initiator of the Child SA pair. TSr specifies the destination address of the traffic forwarded to (or the source address of the traffic forwarded from) the responder of the Child SA pair.

IKEv2 allows the responder to choose a subset of the traffic proposed by the initiator.

When the responder chooses a subset of the traffic proposed by the initiator, it narrows the Traffic Selectors to some subset of the initiator's proposal (provided the set does not become the null set).

If the type of Traffic Selector proposed is unknown, the responder ignores that Traffic Selector, so that the unknown type is not returned in the narrowed set.

To enable the responder to choose the appropriate range, if the initiator has requested the SA due to a data packet, the initiator SHOULD include as the first Traffic Selector in each of TSi and TSr a very specific Traffic Selector including the addresses in the packet triggering the request. If the initiator creates the Child SA pair not in response to an arriving packet, but rather, say, upon startup, then there may be no specific addresses the initiator prefers for the initial tunnel over any other. In that case, the first values in TSi and TSr can be ranges rather than specific values.

As minimal implementations might only support one SA, the traffic selectors will usually be from initiator's IP address to responders IP address (i.e. no port or protocol selectors and only one range).

A.11.1. Traffic Selector

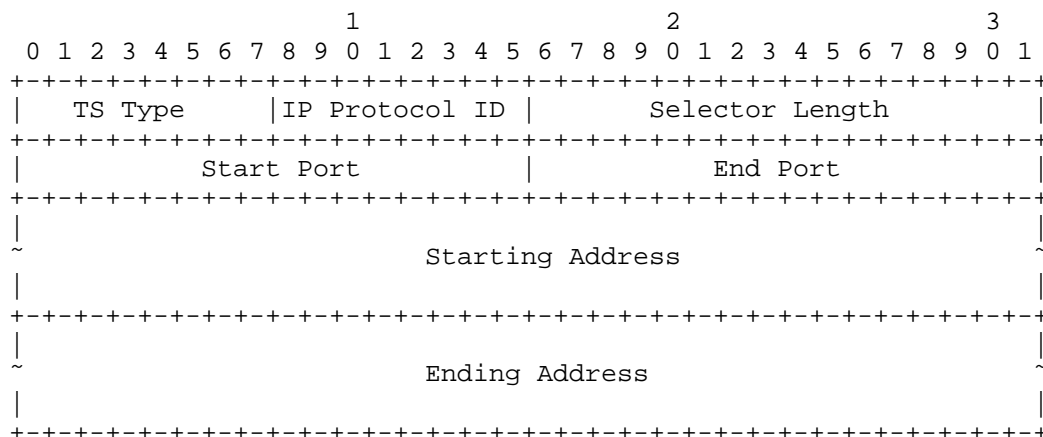


Figure 15: Traffic Selector

- o TS Type (one octet) - Specifies the type of Traffic Selector.
- o IP protocol ID (1 octet) - Value specifying an associated IP protocol ID (such as UDP, TCP, and ICMP). A value of zero means that the protocol ID is not relevant to this Traffic Selector -- the SA can carry all protocols.
- o Selector Length - Specifies the length of this Traffic Selector substructure including the header.

- o Start Port (2 octets, unsigned integer) - Value specifying the smallest port number allowed by this Traffic Selector. For protocols for which port is undefined (including protocol 0), or if all ports are allowed, this field MUST be zero.
- o End Port (2 octets, unsigned integer) - Value specifying the largest port number allowed by this Traffic Selector. For protocols for which port is undefined (including protocol 0), or if all ports are allowed, this field MUST be 65535.
- o Starting Address - The smallest address included in this Traffic Selector (length determined by TS Type).
- o Ending Address - The largest address included in this Traffic Selector (length determined by TS Type).

The following table lists values for the Traffic Selector Type field and the corresponding Address Selector Data.

TS Type	Value

TS_IPV4_ADDR_RANGE	7
A range of IPv4 addresses, represented by two four-octet values. The first value is the beginning IPv4 address (inclusive) and the second value is the ending IPv4 address (inclusive). All addresses falling between the two specified addresses are considered to be within the list.	
TS_IPV6_ADDR_RANGE	8
A range of IPv6 addresses, represented by two sixteen-octet values. The first value is the beginning IPv6 address (inclusive) and the second value is the ending IPv6 address (inclusive). All addresses falling between the two specified addresses are considered to be within the list.	

A.12. Encrypted Payload

The Encrypted payload, denoted SK{...} in this document, contains other payloads in encrypted form.

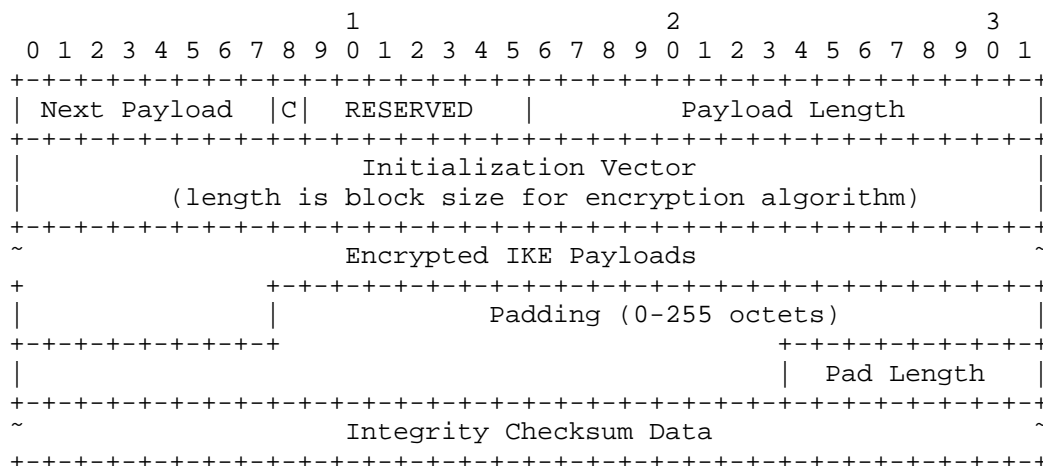


Figure 16: Encrypted Payload Format

- o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.
- o Payload Length - Includes the lengths of the header, initialization vector (IV), Encrypted IKE payloads, Padding, Pad Length, and Integrity Checksum Data.
- o Initialization Vector - For CBC mode ciphers, the length of the initialization vector (IV) is equal to the block length of the underlying encryption algorithm. Senders MUST select a new unpredictable IV for every message; recipients MUST accept any value. The reader is encouraged to consult [MODES] for advice on IV generation. In particular, using the final ciphertext block of the previous message is not considered unpredictable. For modes other than CBC, the IV format and processing is specified in the document specifying the encryption algorithm and mode.
- o IKE payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.
- o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the payloads, the Padding, and the Pad Length to be a multiple of the encryption block size. This field is encrypted with the negotiated cipher.

- o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.
- o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Appendix B. Useful Optional Features

There are some optional features of IKEv2, which might be useful for minimal implementations in some scenarios. Such features include Raw RSA keys authentication, and sending IKE SA delete notification.

B.1. IKE SA Delete Notification

In some scenarios the minimal implementation device creates IKE SA, sends one or few packets, perhaps gets some packets back, and then device goes back to sleep forgetting the IKE SA. In such scenarios it would be nice for the minimal implementation to send the IKE SA delete notification to tell the other end that the IKE SA is going away, so it can free the resources.

Deleting the IKE SA can be done using by sending one packet with fixed Message ID, and with only one payload inside the encrypted payload. The other end will send back an empty response:

```

Initiator                                Responder
-----
HDR(SPIi=xxx, SPIr=yyy, INFORMATIONAL,
  Flags: Initiator, Message ID=2),
  SK {D}  -->

<-- HDR(SPIi=xxx, SPIr=yyy, INFORMATIONAL,
      Flags: Response, Message ID=2),
      SK {}

```

The delete payload format is:

```

          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          |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Protocol ID  |   SPI Size   |          Num of SPIs            |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                           |
|      Security Parameter Index(es) (SPI)      |
|                                     |                           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 17: Delete Payload Format

- o Protocol ID (1 octet) - Must be 1 for an IKE SA.

- o SPI Size (1 octet) - Length in octets of the SPI as defined by the protocol ID. It MUST be zero for IKE (SPI is in message header).
- o Num of SPIs (2 octets, unsigned integer) - The number of SPIs contained in the Delete payload. This MUST be zero for IKE.
- o Security Parameter Index(es) (variable length) - Identifies the specific Security Association(s) to delete. The length of this field is determined by the SPI Size and Num of SPIs fields. This field is empty for the IKE SA delete.

B.2. Raw RSA keys

In some scenarios the shared secret authentication is not safe enough, as anybody who knows the secret can impersonate himself of being the server. If the shared secret is printed on the side of the device, then anybody who gets physical access to the device can read it. In such environments public key authentication allows stronger authentication with minimal operational overhead. Certificate support is quite complex, and minimal implementations do not usually have need for them. Using Raw RSA keys is much simpler, and it allows similar scalability than certificates. The fingerprint of the Raw RSA key can still be distributed by for example printing it on the side of the device allowing similar setup than using shared secret.

Raw RSA keys can also be used in leap of faith or baby duck style initial setup, where the device imprints itself to the first device it sees when it first time boots up. After that initial connection it stores the fingerprint of the Raw RSA key of the server to its own configuration and verifies that it never changes (unless reset to factory setting or similar command is issued).

This changes the initial IKE_AUTH payloads as follows:

Initiator	Responder

HDR(SPIi=xxx, SPIr=yyy, IKE_AUTH,	
Flags: Initiator, Message ID=1),	
SK {IDi, CERT, AUTH, SAI2, TSi, TSr,	
N(INITIAL_CONTACT)} -->	
	<-- HDR(SPIi=xxx, SPIr=yyy, IKE_AUTH, Flags:
	Response, Message ID=1),
	SK {IDr, CERT, AUTH, SAR2, TSi, TSr}

The CERT payloads contains the Raw RSA keys used the sign the hash of the InitiatorSignedOctects/ResponderSignedOctects when generating

AUTH payload. Minimal implementations should use SHA-1 as the hash function as that is the SHOULD support algorithm specified in the RFC5996, so it is the most likely one that is supported by all devices.

Author's Address

Tero Kivinen
AuthenTec
Eerikinkatu 28
HELSINKI FI-00180
FI

Email: kivinen@iki.fi

Network Working Group
Internet-Draft
Updates: 7296 (if approved)
Intended status: Standards Track
Expires: April 18, 2016

T. Kivinen
INSIDE Secure
P. Wouters
Red Hat
H. Tschofenig
October 16, 2015

Generic Raw Public Key Support for IKEv2
draft-kivinen-ipsecme-oob-pubkey-14.txt

Abstract

The Internet Key Exchange Version 2 (IKEv2) protocol did have support for raw public keys, but it only supported RSA Raw public keys. In constrained environments it is useful to make use of other types of public keys, such as those based on Elliptic Curve Cryptography. This document updates RFC 7296, adding support for other types of raw public keys to IKEv2.

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 April 18, 2016.

Copyright Notice

Copyright (c) 2015 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	3
3. Certificate Encoding Payload	3
4. Security Considerations	4
5. IANA Considerations	5
6. Acknowledgements	5
7. References	5
7.1. Normative References	5
7.2. Informative References	6
Appendix A. Examples	7
A.1. ECDSA Example	7
A.2. RSA Example	8
Authors' Addresses	10

1. Introduction

This document replaces an algorithm-specific version of raw public keys of Internet Key Exchange Version 2 (IKEv2) [RFC7296] with a generic version of raw public keys that is algorithm agnostic.

In [RFC5996] IKEv2 had support for PKCS #1 encoded RSA keys, i.e., a DER-encoded RSAPublicKey structure (see [RSA] and [RFC3447]). Other raw public key types are, however, not supported. In [RFC7296] this feature was removed, and this document adds support for raw public keys back to IKEv2 in a more generic way.

DNSSEC allows public keys to be associated with domain names for usage with security protocols like IKEv2 and Transport Layer Security (TLS) [RFC5246] but it relies on extensions in those protocols to be specified.

The Raw Public Keys in Transport Layer Security specification ([RFC7250]) adds generic support for raw public keys to TLS by re-using the SubjectPublicKeyInfo format from the X.509 Public Key Infrastructure Certificate profile [RFC5280].

This document is similar to the Raw Public Keys in Transport Layer Security specification and applies the concept to IKEv2 to support all public key formats defined by PKIX. This approach also allows future public key extensions to be supported without the need to introduce further enhancements to IKEv2.

To support new types of public keys in IKEv2 the following changes are needed:

- o A new Certificate Encoding format needs to be defined for carrying the SubjectPublicKeyInfo structure. Section 3 specifies this new encoding format.
- o A new Certificate Encoding type needs to be allocated from the IANA registry. Section 5 contains this request to IANA.

The base IKEv2 specification includes support for RSA and DSA signatures, but the Signature Authentication in IKEv2 [RFC7427] extended IKEv2 so that signature methods over any key type can be used. Implementations using raw public keys SHOULD use the Digital Signature method described in the RFC7427.

When using raw public keys, the authenticated identity is not usually the identity from the ID payload, but instead the public key itself is used as identity for the other end. This means that ID payload contents might not be useful for authentication purposes. It might still be used for policy decisions, for example to simplify the policy lookup etc. Alternatively, the ID_NULL type [RFC7619] can be used to indicate that the ID payload is not relevant to this authentication.

2. Terminology

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

3. Certificate Encoding Payload

Section 3.6 of RFC 7296 defines the Certificate payload format as shown in Figure 1.

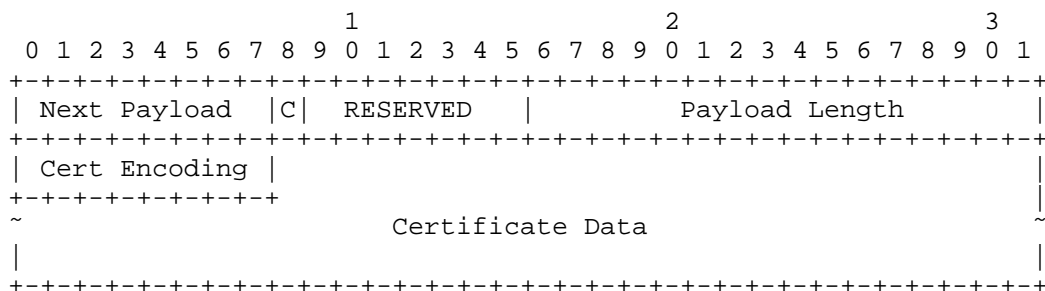


Figure 1: Certificate Payload Format.

To support raw public keys, the field values are as follows:

- o Certificate Encoding (1 octet) - This field indicates the type of certificate or certificate-related information contained in the Certificate Data field.

Certificate Encoding	Value
-----	-----
Raw Public Key	TBD

- o Certificate Data (variable length) - Actual encoding of the certificate data.

In order to provide a simple and standard way to indicate the key type when the encoding type is 'Raw Public Key', the SubjectPublicKeyInfo structure of the PKIX certificate is used. This is a very simple encoding, as most of the ASN.1 part can be included literally, and recognized by block comparison. See [RFC7250] Appendix A for a detailed breakdown. In addition, Appendix A has several examples.

In addition to the Certificate payload, the Cert Encoding for Raw Public Key can be used in the Certificate Request payload. In that case the Certification Authority field MUST be empty if the "Raw Public Key" certificate encoding is used.

For RSA keys, the implementations MUST follow the public key processing rules of section 1.2 of the Additional Algorithms and Identifiers for RSA Cryptography for PKIX ([RFC4055]) even when the SubjectPublicKeyInfo is not part of a certificate, but rather sent as a Certificate Data field. This means that RSASSA-PSS and RSASSA-PSS-params inside the SubjectPublicKeyInfo structure MUST be sent when applicable.

4. Security Considerations

An IKEv2 deployment using raw public keys needs to utilize an out-of-band public key validation procedure to be confident in the authenticity of the keys being used. One way to achieve this goal is to use a configuration mechanism for provisioning raw public keys into the IKEv2 software. "Smart object" deployments are likely to use such preconfigured public keys.

Another approach is to rely on secure DNS to associate public keys with domain names using the IPSECKEY DNS RRtype [RFC4025]. More information can be found in DNS-Based Authentication of Named Entities (DANE) [RFC6394].

This document does not change the security assumptions made by the IKEv2 specification since "Raw RSA Key" support was already available in IKEv2 in [RFC5996]. This document only generalizes raw public key support.

5. IANA Considerations

This document allocates a new value from the IKEv2 Certificate Encodings registry:

TBD	Raw Public Key
-----	----------------

6. Acknowledgements

This document reproduces some parts of the similar TLS document ([RFC7250]).

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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<http://www.rfc-editor.org/info/rfc5280>>.
- [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>>.
- [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>>.
- [RFC7619] Smyslov, V. and P. Wouters, "The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 7619, DOI 10.17487/RFC7619, August 2015, <<http://www.rfc-editor.org/info/rfc7619>>.

7.2. Informative References

- [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, <<http://www.rfc-editor.org/info/rfc3447>>.
- [RFC4025] Richardson, M., "A Method for Storing IPsec Keying Material in DNS", RFC 4025, DOI 10.17487/RFC4025, March 2005, <<http://www.rfc-editor.org/info/rfc4025>>.
- [RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, DOI 10.17487/RFC4055, June 2005, <<http://www.rfc-editor.org/info/rfc4055>>.
- [RFC4754] Fu, D. and J. Solinas, "IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 4754, DOI 10.17487/RFC4754, January 2007, <<http://www.rfc-editor.org/info/rfc4754>>.
- [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>>.
- [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, <<http://www.rfc-editor.org/info/rfc5480>>.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, DOI 10.17487/RFC5996, September 2010, <<http://www.rfc-editor.org/info/rfc5996>>.
- [RFC6394] Barnes, R., "Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE)", RFC 6394, DOI 10.17487/RFC6394, October 2011, <<http://www.rfc-editor.org/info/rfc6394>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <<http://www.rfc-editor.org/info/rfc7250>>.

[RSA] R. Rivest, , A. Shamir, , and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", February 1978.

Appendix A. Examples

This appendix provides examples of the actual payloads sent on the wire.

A.1. ECDSA Example

This first example uses the 256-bit ECDSA private/public key pair defined in section 8.1 of the IKEv2 ECDSA document [RFC4754].

The public key is as follows:

- o Algorithm : id-ecPublicKey (1.2.840.10045.2.1)
- o Fixed curve: secp256r1 (1.2.840.10045.3.1.7)
- o Public key x coordinate : cb28e099 9b9c7715 fd0a80d8 e47a7707 9716cbbf 917dd72e 97566eal c066957c
- o Public key y coordinate : 2b57c023 5fb74897 68d058ff 4911c20f dbe71e36 99d91339 afbb903e e17255dc

The SubjectPublicKeyInfo ASN.1 object is as follows:

```
0000 :        SEQUENCE
0002 :        SEQUENCE
0004 :            OBJECT IDENTIFIER   id-ecPublicKey (1.2.840.10045.2.1)
000d :            OBJECT IDENTIFIER   secp256r1 (1.2.840.10045.3.1.7)
0017 :            BIT STRING   (66 bytes)
00000000: 0004 cb28 e099 9b9c 7715 fd0a 80d8 e47a
00000010: 7707 9716 cbbf 917d d72e 9756 6eal c066
00000020: 957c 2b57 c023 5fb7 4897 68d0 58ff 4911
00000030: c20f dbe7 1e36 99d9 1339 afbb 903e e172
00000040: 55dc
```

The first byte (00) of the bit string indicates that there is no "number of unused bits", and the second byte (04) indicates uncompressed form ([RFC5480]). Those two octets are followed by the values of X and Y.

The final encoded SubjectPublicKeyInfo object is as follows:

```
00000000: 3059 3013 0607 2a86 48ce 3d02 0106 082a
00000010: 8648 ce3d 0301 0703 4200 04cb 28e0 999b
00000020: 9c77 15fd 0a80 d8e4 7a77 0797 16cb bf91
00000030: 7dd7 2e97 566e alc0 6695 7c2b 57c0 235f
00000040: b748 9768 d058 ff49 11c2 0fdb e71e 3699
00000050: d913 39af bb90 3ee1 7255 dc
```

This will result in the final IKEv2 Certificate Payload:

```
00000000: NN00 0060 XX30 5930 1306 072a 8648 ce3d
00000010: 0201 0608 2a86 48ce 3d03 0107 0342 0004
00000020: cb28 e099 9b9c 7715 fd0a 80d8 e47a 7707
00000030: 9716 cbbf 917d d72e 9756 6ea1 c066 957c
00000040: 2b57 c023 5fb7 4897 68d0 58ff 4911 c20f
00000050: dbe7 1e36 99d9 1339 afbb 903e e172 55dc
```

Where NN is the next payload type (i.e., the type of the payload that immediately follows this Certificate payload).

Note to the RFC editor / IANA, replace the XX above with the newly allocated Raw Public Key number (in hex notation), and remove this note.

A.2. RSA Example

This second example uses a random 1024-bit RSA key.

The public key is as follows:

- o Algorithm : rsaEncryption (1.2.840.113549.1.1.1)
- o Modulus n (1024 bits, decimal):


```
1323562071162740912417075551025599045700
3972512968992059076067098474693867078469
7654066339302927451756327389839253751712
9485277759962777278073526290329821841100
9721044682579432931952695408402169276996
5181887843758615443536914372816830537901
8976615344413864477626646564638249672329
04996914356093900776754835411
```
- o Modulus n (1024 bits, hexadecimal): bc7b4347 49c7b386 00bfa84b


```
44f88187 9a2dda08 d1f0145a f5806c2a ed6a6172 ff0dc3d4 cd601638
e8ca348e bdca5742 31cad9c7 12e209b1 fddba58a 8c62b369 038a3d1e
aa727c1f 39ae49ed 6ebc30f8 d9b52e23 385a4019 15858c59 be72f343
fb1eb87b 16ffc5ab 0f8f8fe9 f7cb3e66 3d8fe9f9 ecfa1230 66f36835
8ceaefd3
```

- o Exponent e (17 bits, decimal): 65537
- o Exponent e (17 bits, hexadecimal): 10001

The SubjectPublicKeyInfo ASN.1 object is as follows:

```

0000 :      SEQUENCE
0003 :      SEQUENCE
0005 :      OBJECT IDENTIFIER rsaEncryption (1.2.840.113549.1.1.1)
0016 :      NULL
0018 :      BIT STRING  (141 bytes)
00000000: 0030 8189 0281 8100 bc7b 4347 49c7 b386
00000010: 00bf a84b 44f8 8187 9a2d da08 d1f0 145a
00000020: f580 6c2a ed6a 6172 ff0d c3d4 cd60 1638
00000030: e8ca 348e bdca 5742 31ca dc97 12e2 09b1
00000040: fddb a58a 8c62 b369 038a 3d1e aa72 7c1f
00000050: 39ae 49ed 6ebc 30f8 d9b5 2e23 385a 4019
00000060: 1585 8c59 be72 f343 fb1e b87b 16ff c5ab
00000070: 0f8f 8fe9 f7cb 3e66 3d8f e9f9 ecfa 1230
00000080: 66f3 6835 8cea efd3 0203 0100 01

```

The first byte (00) of the bit string indicates that there is no "number of unused bits". Inside that bit string there is an ASN.1 sequence having 2 integers. The second byte (30) indicates that this is beginning of the sequence, and the next byte (81) indicates the length does not fit in 7 bits, but requires one byte, so the length is in the next byte (89). Then starts the first integer with tag (02) and length (81 81). After that we have the modulus (prefixed with 0 so it will not be a negative number). After the modulus there follows the tag (02) and length (03) of the exponent, and the last 3 bytes are the exponent.

The final encoded SubjectPublicKeyInfo object is as follows:

```

00000000: 3081 9f30 0d06 092a 8648 86f7 0d01 0101
00000010: 0500 0381 8d00 3081 8902 8181 00bc 7b43
00000020: 4749 c7b3 8600 bfa8 4b44 f881 879a 2dda
00000030: 08d1 f014 5af5 806c 2aed 6a61 72ff 0dc3
00000040: d4cd 6016 38e8 ca34 8ebd ca57 4231 cadc
00000050: 9712 e209 b1fd dba5 8a8c 62b3 6903 8a3d
00000060: 1eaa 727c 1f39 ae49 ed6e bc30 f8d9 b52e
00000070: 2338 5a40 1915 858c 59be 72f3 43fb 1eb8
00000080: 7b16 ffc5 ab0f 8f8f e9f7 cb3e 663d 8fe9
00000090: f9ec fa12 3066 f368 358c eaef d302 0301
000000a0: 0001

```

This will result in the final IKEv2 Certificate Payload:

```
00000000: NN00 00a7 XX30 819f 300d 0609 2a86 4886
00000010: f70d 0101 0105 0003 818d 0030 8189 0281
00000020: 8100 bc7b 4347 49c7 b386 00bf a84b 44f8
00000030: 8187 9a2d da08 d1f0 145a f580 6c2a ed6a
00000040: 6172 ff0d c3d4 cd60 1638 e8ca 348e bdca
00000050: 5742 31ca dc97 12e2 09b1 fddb a58a 8c62
00000060: b369 038a 3d1e aa72 7c1f 39ae 49ed 6ebc
00000070: 30f8 d9b5 2e23 385a 4019 1585 8c59 be72
00000080: f343 fb1e b87b 16ff c5ab 0f8f 8fe9 f7cb
00000090: 3e66 3d8f e9f9 ecfa 1230 66f3 6835 8cea
000000a0: efd3 0203 0100 01
```

Where NN is the next payload type (i.e., the type of the payload that immediately follows this Certificate payload).

Note to the RFC editor / IANA, replace the XX above with the newly allocated Raw Public Key number, and remove this note.

Authors' Addresses

Tero Kivinen
INSIDE Secure
Eerikinkatu 28
HELSINKI FI-00180
FI

Email: kivinen@iki.fi

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Hannes Tschofenig

Email: Hannes.Tschofenig@gmx.net
URI: <http://www.tschofenig.priv.at>

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2013

D. McGrew
Cisco Systems
W. Feghali
Intel Corp.
October 15, 2012

Cryptographic Algorithm Implementation Requirements and Usage Guidance
for Encapsulating Security Payload (ESP) and Authentication Header (AH)
draft-mcgrew-ipsec-me-esp-ah-reqts-00

Abstract

This Internet Draft is an individual submission that proposes an update to the Cryptographic Algorithm Implementation Requirements for ESP and AH; it also adds usage guidance to help in the selection of these algorithms.

The Encapsulating Security Payload (ESP) and Authentication Header (AH) protocols makes use of various cryptographic algorithms to provide confidentiality and/or data origin authentication to protected data communications in the IP Security (IPsec) architecture. To ensure interoperability between disparate implementations, the IPsec standard specifies a set of mandatory-to-implement algorithms. This document specifies the current set of mandatory-to-implement algorithms for ESP and AH, specifies algorithms that should be implemented because they may be promoted to mandatory at some future time, and also recommends against the implementation of some obsolete algorithms. Usage guidance is also provided to help the user of ESP and AH best achieve their security goals through appropriate choices of cryptographic algorithms.

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 April 18, 2013.

Copyright Notice

Copyright (c) 2012 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. Requirements Language	3
2. Implementation Requirements	5
2.1. ESP Authenticated Encryption (Combined Mode Algorithms) .	5
2.2. ESP Encryption Algorithms	5
2.3. ESP Authentication Algorithms	5
2.4. AH Authentication Algorithms	5
2.5. Summary of Changes	6
3. Usage Guidance	7
4. Rationale	8
4.1. Authenticated Encryption	8
4.2. Encryption Transforms	8
4.3. Authentication Transforms	8
5. Acknowledgements	10
6. IANA Considerations	11
7. Security Considerations	12
8. References	13
8.1. Normative References	13
8.2. Informative References	13
Authors' Addresses	15

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].

To ensure interoperability between disparate implementations, it is necessary to specify a set of mandatory-to-implement algorithms. This ensures that there is at least one algorithm that all implementations will have in common. This document specifies the current set of mandatory-to-implement algorithms for ESP and AH, specifies algorithms that should be implemented because they may be promoted to mandatory at some future time, and also recommends against the implementation of some obsolete algorithms. Usage guidance is also provided to help the user of ESP and AH best achieve their security goals through appropriate choices of mechanisms.

The nature of cryptography is that new algorithms surface continuously and existing algorithms are continuously attacked. An algorithm believed to be strong today may be demonstrated to be weak tomorrow. Given this, the choice of mandatory-to-implement algorithm should be conservative so as to minimize the likelihood of it being compromised quickly. Thought should also be given to performance considerations as many uses of IPsec will be in environments where performance is a concern.

The ESP and AH mandatory-to-implement algorithm(s) may need to change over time to adapt to new developments in cryptography. For this reason, the specification of the mandatory-to-implement algorithms is not included in the main IPsec, ESP, or AH specifications, but is instead placed in this document. Ideally, the mandatory-to-implement algorithm of tomorrow should already be available in most implementations of IPsec by the time it is made mandatory. To facilitate this, this document identifies such algorithms, as they are known today. There is no guarantee that the algorithms that we believe today may be mandatory in the future will in fact become so. All algorithms known today are subject to cryptographic attack and may be broken in the future.

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].

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

MUST- This term means the same as MUST. However, we expect that at some point in the future this algorithm will no longer be a MUST.

SHOULD+ This term means the same as SHOULD. However, it is likely that an algorithm marked as SHOULD+ will be promoted at some future time to be a MUST.

SHOULD- This term means the same as SHOULD. However, it is likely that an algorithm marked as SHOULD- will be deprecated to a MAY or worse in a future version of this document.

SHOULD NOT+ This term means the same as SHOULD NOT. However, it is likely that an algorithm marked as SHOULD NOT+ will be deprecated to a MUST NOT in a future version of this document.

2. Implementation Requirements

This section specifies the cryptographic algorithms that **MUST** be implemented, and provides guidance about ones that **SHOULD** or **SHOULD NOT** be implemented.

2.1. ESP Authenticated Encryption (Combined Mode Algorithms)

ESP combined mode algorithms provide both confidentiality and authentication services; in cryptographic terms, these are authenticated encryption algorithms [RFC5116]. Authenticated encryption transforms are listed in the ESP encryption transforms IANA registry.

Requirement	Authenticated Encryption Algorithm
-----	-----
SHOULD+	AES-GCM [RFC4106]
MAY	AES-CCM [RFC4309]

2.2. ESP Encryption Algorithms

Requirement	Encryption Algorithm
-----	-----
MUST	NULL [RFC2410]
MUST	AES-128-CBC [RFC3602]
MAY	AES-CTR [RFC3686]
SHOULD NOT	TripleDES-CBC [RFC2451]
SHOULD NOT+	DES-CBC [RFC2405]

2.3. ESP Authentication Algorithms

Requirement	Authentication Algorithm (notes)
-----	-----
MUST	HMAC-SHA1-96 [RFC2404]
SHOULD+	AES-GMAC [RFC4543]
SHOULD	AES-XCBC-MAC-96 [RFC3566]
MAY	NULL [RFC4303]
MAY	HMAC-SHA-256 [RFC4868]
MAY	HMAC-SHA-384 [RFC4868]
MAY	HMAC-SHA-512 [RFC4868]
SHOULD NOT	HMAC-MD5-96 [RFC2403]

2.4. AH Authentication Algorithms

The requirements for AH are the same as for ESP Authentication Algorithms, except that NULL authentication is inapplicable.

2.5. Summary of Changes

Old Requirement ----	New Requirement -----	Algorithm (notes) -----
MAY	SHOULD+	AES-GCM [RFC4106]
MAY	SHOULD+	AES-GMAC [RFC4543]
MUST-	SHOULD NOT	TripleDES-CBC [RFC2451]
SHOULD+	SHOULD	AES-XCBC-MAC-96 [RFC3566]
SHOULD	MAY	AES-CTR [RFC3686]
MAY	SHOULD NOT	HMAC-MD5-96 [RFC2403]

3. Usage Guidance

Since ESP and AH can be used in several different ways, this note provides guidance on the best way to utilize these mechanisms.

ESP can provide confidentiality, data origin authentication, or the combination of both of those security services. AH provides only data origin authentication. Background information on those security services is available [RFC4949]. In the following, we shorten 'data origin authentication' to 'authentication'.

Both confidentiality and authentication SHOULD be provided. If confidentiality is not needed, then authentication MAY be provided. Confidentiality without authentication is not effective [DP07] and SHOULD NOT be used. We describe each of these cases in more detail below.

To provide confidentiality and authentication, an authenticated encryption transform SHOULD be used in ESP, in conjunction with NULL authentication. Alternatively, an ESP encryption transform and ESP authentication transform MAY be used together (provided that neither transform is NULL). If authentication on the IP header is needed in conjunction with confidentiality of higher-layer data, then AH SHOULD be used in addition to the transforms recommended above. It is NOT RECOMMENDED to use ESP with NULL authentication in conjunction with AH; some configurations of this combination of services have been shown to be insecure [PD10].

To provide authentication without confidentiality, an authentication transform MUST be used in either ESP or AH. It is not possible to provide effective confidentiality without authentication, because the lack of authentication undermines the efficacy of encryption [B96][V02]. An encryption transform MUST NOT be used with a NULL authentication transform (unless the encryption transform is an authenticated encryption transform).

4. Rationale

This section explains the principles behind the implementation requirements described above.

4.1. Authenticated Encryption

This note encourages the use of authenticated encryption algorithms because they can provide significant efficiency and throughput advantages, and the tight binding between authentication and encryption can be a security advantage [RFC5116].

AES-GCM [RFC4106] brings significant performance benefits [KKGECD], has been incorporated into IPsec recommendations [RFC6379] and has emerged as the preferred authenticated encryption method in IPsec and other standards.

4.2. Encryption Transforms

Since ESP encryption is optional, support for the "NULL" algorithm is required to maintain consistency with the way services are negotiated. Note that while authentication and encryption can each be "NULL", they MUST NOT both be "NULL" [RFC4301] [H10].

AES Counter Mode (AES-CTR) is an efficient encryption method, but it provides no authentication capability. The AES-GCM authenticated encryption method has all of the advantages of AES-CTR, while also providing authentication. Thus this note moves AES-CTR from a SHOULD to a MAY.

The Triple Data Encryption Standard (TDES) is obsolete because of its small block size; as with all 64-bit block ciphers, it SHOULD NOT be used to encrypt more than one gigabyte of data with a single key. Its key size is smaller than that of the Advanced Encryption Standard (AES), while at the same time its performance and efficiency is worse. Thus, its use in new implementations is discouraged.

The Data Encryption Standard (DES) is obsolete because of its small key size and small block size. There have been publicly demonstrated and open-design special-purpose cracking hardware. Therefore, its use is discouraged.

4.3. Authentication Transforms

AES-GMAC provides good security along with performance advantages, even over HMAC-MD5. In addition, it uses the same internal components as AES-GCM and is easy to implement in a way that shares components with that authenticated encryption algorithm.

The MD5 hash function has been found to not meet its goal of collision resistance; it is so weak that its use in digital signatures is highly discouraged [RFC6151]. There have been theoretical results against HMAC-MD5, but that message authentication code does not seem to have a practical vulnerability. Thus, it may not be urgent to remove HMAC-MD5 from the existing protocols; however, since MD5 must not be used for digital signatures, HMAC-MD5 should not be included in new implementations.

SHA-1 has been found to not meet its goal of collision resistance. However, HMAC-SHA-1 does not rely on this property, and HMAC-SHA-1 is believed to be secure.

The HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 are believed to provide a good security margin, and they perform adequately on many platforms. However, these algorithms are listed only as MAY implement in this note, because HMAC-SHA-1 support is widespread and its security is good, AES-GMAC provides good security with better performance, and Authenticated Encryption algorithms do not need any authentication methods.

AES-XCBC has not seen widespread deployment, despite being previously being recommended as a SHOULD+ in RFC4305. Thus this draft lists it only as a SHOULD.

5. Acknowledgements

Much of the wording herein was adapted from [RFC4835], the parent document of this document. That RFC itself borrows from [RFC4305], which borrows in turn from [RFC4307]. RFC4835, RFC4305, and RFC4307 were authored by Vishwas Manral, Donald Eastlake, and Jeffrey Schiller respectively.

Thanks are due to Scott Fluhrer, Dan Harkins, Brian Weis, and Cheryl Madson for insightful feedback on this draft.

6. IANA Considerations

This memo includes no request to IANA.

7. 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 so far leads us to believe that they will likely remain secure into the foreseeable future. However, this is not necessarily forever. We would therefore expect that new revisions of this document will be issued from time to time that reflect the current best practice in this area.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2403] Madson, C. and R. Glenn, "The Use of HMAC-MD5-96 within ESP and AH", 1998.
- [RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", 1998.
- [RFC2405] Madson, C. and N. Doraswamy, "The ESP DES-CBC Cipher Algorithm With Explicit IV", 1998.
- [RFC2410] Glenn, R. and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", 1998.
- [RFC3566] Frankel, S. and H. Herbert, "The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec", 2003.
- [RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", 2003.
- [RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", 2004.
- [RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", 2005.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", 2005.
- [RFC4302] Kent, S., "IP Authentication Header", 2005.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload", 2005.

8.2. Informative References

- [B96] Bellovin, S., "Problem areas for the IP security protocols (Proceedings of the Sixth Usenix Unix Security Symposium)", 1996.
- [DP07] Degabriele, J. and K. Paterson, "Attacking the IPsec Standards in Encryption-only Configurations (IEEE

Symposium on Privacy and Security)", 2007.

- [H10] Hoban, A., "Using Intel AES New Instructions and PCLMULQDQ to Significantly Improve IPsec Performance on Linux", 2010.
- [KKGECD] Kounavis, M., Kang, X., Grewal, K., Eszenyi, M., Gueron, S., and D. Durham, "Encrypting the Internet (SIGCOMM)", 2010.
- [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.
- [RFC4305] Eastlake, D., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)".
- [RFC4307] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", 2005.
- [RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", 2005.
- [RFC4835] Manral, V., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)".
- [RFC4949] Shirley, R., "Internet Security Glossary, Version 2", 2007.
- [RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", 2008.
- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", 2011.
- [RFC6379] Law, L. and J. Solinas, "Suite B Cryptographic Suites for IPsec", 2011.
- [V02] Vaudenay, S., "Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ... (EUROCRYPT)", 2002.

Authors' Addresses

David McGrew
Cisco Systems
13600 Dulles Technology Drive
Herndon, Virginia 20171
USA

Phone: 408 525 8651
Email: mcgrew@cisco.com

Wajdi Feghali
Intel Corp.
75 Reed Road
Hudson, Massachusetts
USA

Phone:
Email: wajdi.k.feghali@intel.com

MIF Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2013

D. Migault
Francetelecom - Orange
C. Williams
MCSR Labs
November 3, 2012

IPsec Multiple Interfaces Problem Statement
draft-mglt-mif-security-requirements-03.txt

Abstract

IKEv2 is the protocol used to set up and negotiate Security Associations between nodes. IKEv2 has not been designed for nodes with multiple interfaces.

This document is focused on IKEv2 ability to set up IPsec protected communications between nodes with multiple interfaces. This document states the problems and provides requirements for IKEv2 to ease IPsec for multiple interface communication.

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 May 7, 2013.

Copyright Notice

Copyright (c) 2012 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	4
2. Introduction	4
3. Use Case 1: VPN with Multiple Interfaces	5
3.1. Initial MIF IPsec Configuration	6
3.1.1. Description	6
3.1.2. Problem Statement	6
3.1.3. Requirements	8
3.2. Mobility	8
3.2.1. Description	8
3.2.2. Problem Statement	9
3.2.3. Requirements	11
3.3. Multihoming	12
3.3.1. Description	12
3.3.2. Problem Statement	13
3.3.3. Requirements	13
3.4. Adding an Interface	14
3.4.1. Description	14
3.4.2. Problem Statement	17
3.4.3. Requirements	18
3.5. Deleting an Interface	18
3.5.1. Description	18
3.5.2. Problem Statement	18
3.5.3. Requirements	18
4. Use Case 2: MIF applications and IPsec Tunnel mode	19
5. Use Case 3: MIF aware applications with Transport mode	20
5.1. Initial MIF IPsec Configuration	21
5.1.1. Description	21
5.1.2. Problem Statement	21
5.1.3. Requirements	21
5.2. Mobility	22
5.2.1. Description	22
5.2.2. Problem Statement	23
5.2.3. Requirements	24
5.3. Multihoming	24
5.3.1. Description	24
5.3.2. Problem Statement	24
5.3.3. Requirements	25
5.4. Adding an Interface	25
5.5. Delete an Interface	25
6. Security Considerations	25

7. IANA Considerations	25
8. Acknowledgment	25
9. References	26
9.1. Normative References	26
9.2. Informational References	26
Authors' Addresses	26

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 protocol suite [RFC4301],[RFC5996] is mainly used to:

- Extend a trusted domain over an untrusted network: This typically corresponds to the Virtual Private Network (VPN) use case. A Security Gateway is a trusted entry point to a trusted network. The end user is connected to an untrusted network and tunnels its traffic to the Security Gateway in a encrypted tunnel using the IPsec tunnel mode. The Security Gateway decapsulates the traffic and forwards it on the trusted network. Once the traffic is in the trusted network it is usually not encrypted anymore. In other words, the traffic is protected from the end user terminal to the Security Gateway, that it to say over the untrusted network.
- Provide end-to-end security: With end-to-end security, the traffic is protected from the source - or the end user in our case - to the destination. The traffic does not require to be tunneled, and any segments of the network between the end user and the destination is considered as untrusted. With end-to-end security, one does not require encapsulation, and the IPsec transport mode can be used.

Currently most devices have multiple interfaces. Mobile phones have most of the time a Wireless LAN (WLAN) and a Radio Access Network (RAN) interface. Laptop can easily have Ethernet / WLAN / RAN with WiMAX interfaces. Furthermore, USB dongle can be plugged to provide additional RAN and WLAN interfaces. Regular PCs, Servers, or CPEs have multiple Ethernet interfaces, with additional WLAN interfaces.

Protocols like SCTP [RFC4960] or MOBIKE [RFC4555] have been designed to use these multiple interfaces for multihoming. Only a single interface is used at a time. The interface used to carry the IP datagrams is called the Primary interface and other interfaces are called Secondary or Alternate interfaces. Alternate interfaces are only expected to be used in case the Primary interface fails.

However, multihoming does not enable the simultaneous use of multiple interfaces which can provide a better use of the available bandwidth. MPTCP [RFC6182] has been designed for that purpose, and SCTP [RFC4960] can also be used for it. Raiciu and al. [Raiciu] showed how using multiple paths improve the performances and robustness of

data centers compared to TCP. Furthermore, a communication may be connected simultaneously to different networks with different technologies and takes advantage of their different characteristics. This is typically the case of Offload when ISPs are offloading their RAN communications to a WLAN network. Motivations for offloading is that RAN cannot support all mobile traffic [Cisco]. As a result, with a RAN and a WLAN interface, Mobile phones and ISPs may balance the communications between an unreliable WLAN with economical bandwidth and always connected RAN with expensive bandwidth.

The document focuses on how applications and services protected with IPsec can also take advantage of multiple interfaces. The traditional VPN application with multiple interfaces is the first use case we consider. However, with the offload usage, ISPs are offloading unprotected communications, services from a trusted network - like the RAN - to an untrusted and unreliable network - like the WLAN. This means that the ISP must protect the communications related to these services and applications while being offloaded. IPsec appears to be one way to secure communications transparently to the application.

They are two ways to secure the communications with IPsec. One way is to tunnel the communication to a Security Gateway. The other is to provide end to end security. The document will consider both ways.

Section 3 considers the specific case of VPN with multiple interfaces. Section 4 extends the previous use case by considering the general case of IPsec protected communications using the Tunnel mode. Finally Section 5 considers the case of IPsec protected communications with the Transport mode. For each case, the document details different scenarios that take advantage of multiple interfaces. Then it positions IKEv2 toward each of these scenarios and points out requirements

3. Use Case 1: VPN with Multiple Interfaces

This section describes the VPN scenario with connectivity described in figure 1, the End User (EU) has multiple interfaces and figure 1 represents 3 interfaces bound to 3 IP addresses EU @IP_outer(i), (i in {1, 2, 3}).

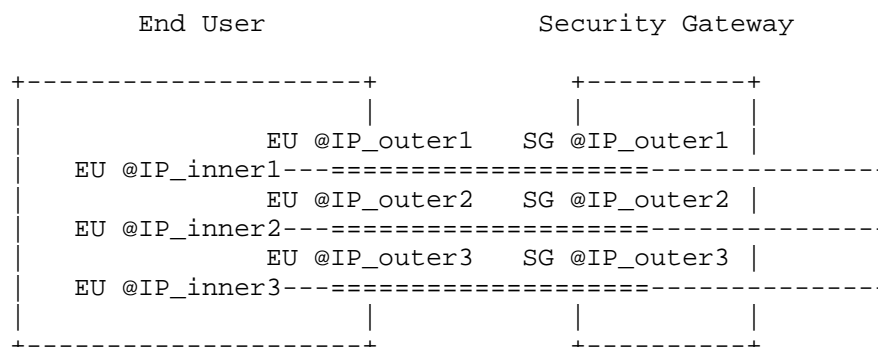


Figure 1: VPN with Multiple Interfaces

3.1. Initial MIF IPsec Configuration

3.1.1. Description

This section details how the End User with its three interfaces set (EU @IP_outer(*i*), *i* in {1, 2, 3}) can set an IPsec configuration as represented in figure 1. We consider the IPsec configuration is set using IKEv2, and that the End User uses only a single IKEv2 channel. In other words, each interface MUST NOT be considered independently from each other with its own IKEv2 channel and its own Security Associations.

One of these End User IP addresses is used to set the IKEv2 channel. This IP address is used to set the IKE_SA as well as for all IKEv2 exchanges. Suppose EU @IP_outer1 is used for the IKE_SA.

Using the IKEv2 channel, the End User requests the inner IP addresses EU @IP_inner(*i*), *i* in {1, 2, 3}. If the Security Gateway has multiple interfaces, it advertises the End User, what are the available interfaces.

Once the End User has inner and outer IP addresses, it starts negotiating via the IKEv2 channel the different Security Associations. For each Security Association, the End User and the Security Gateway SHOULD be able to agree on the Traffic Selectors (i.e. the inner IP addresses) as well as the outer IP addresses used for the Tunnel.

3.1.2. Problem Statement

This section positions the current IKEv2 specifications toward the scenario described in Section 3.1.1

To request multiple inner IP addresses, the End User can use the IKEv2 with multiple INTERNAL_IP*_ADDRESS Configuration Attributes in the CFG Payload (Section 3.15 [RFC5996]).

Currently IKEv2 does not provide ways for the Security Gateway to announce the End User the available outer IP addresses - SG @IP_outer1, SG @IP_outer2 and SG @IP_outer3. [I-D.arora-ipsecme-ikev2-alt-tunnel-addresses] details how this could be mitigated. Note that in the VPN use case, the initiator - that is to say the End User - is more likely to request the Security Gateway outer IP addresses, then the reverse. In other words, there seems very few interest for the responder to know the different outer IP addresses of the End User. However, as detailed in Section 5 the more general case SHOULD consider that both initiator and responder can advertise the available interface when the IKEv2 negotiation is initiated.

IKEv2 makes possible the negotiation of the Security Associations associated to each of the EU @IP_inner(i) IP addresses using a Traffic Selector Payload with one or multiple Traffic Selectors (section 3.13 [RFC5996]). IKEv2 even enables the simultaneous negotiation of Security Associations. However, currently the Security Association negotiation does not specify the outer IP addresses. The outer IP addresses are those used for the IKEv2 channel. In other words, current IKEv2 only considers a single working IP address for both the End User and the Security Gateway. Figure 2 illustrates current IKEv2 capabilities in the VPN use case with different Traffic Selectors associated to a single outer IP address. While negotiating a Security Association, IKEv2 SHOULD be able to specify the source and destination IP addresses.

Note that the benefits of specifying the outer IP addresses provides the End User or Initiator the ability to use simultaneously multiple interfaces. In the specific case of figure 1, the Security Gateway will most likely have a single IP outer IP address. We considered multiple IP addresses on the Security Gateway for the more general case.

Currently, IKEv2 does not provide the ability to negotiate the outer IP addresses of the Tunnel. By default, the outer IP addresses of the Child Security Associations are those used for the IKEv2 channel. This results in the configuration as represented in figure 2. The configuration of figure 1 does not result from an IKEv2 negotiation.

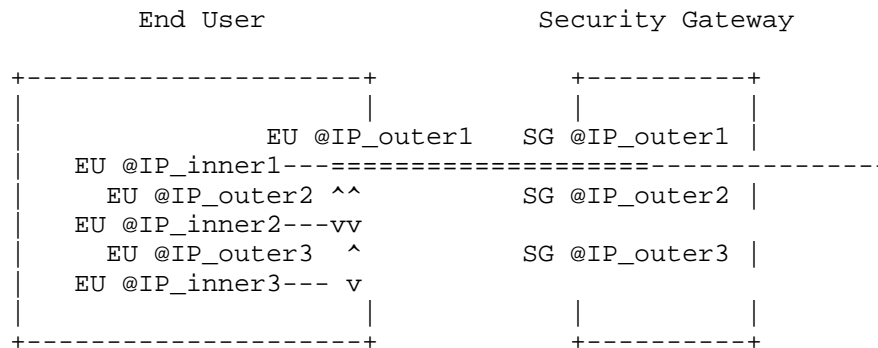


Figure 2: VPN with Multiple Interfaces
Current IKEv2 negotiation

3.1.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible:

- 1. To specify the different outer IP addresses for the tunnel mode in the Security Association negotiation.
- 2. Make possible the Responder and Initiator to announce its interfaces.

3.2. Mobility

3.2.1. Description

This section considers how a node with multiple interfaces can modify the value of the outer IP address. In the Tunnel mode, changing the outer IP address results in a mobility, however this should be seen as updating a parameter of the Security Association. Figure 3 illustrates a mobility where EU @IP_outer3 in Figure 1 is updated by EU @IP_outer4.

In fact, the Security Association associated to EU @IP_inner3 includes the outer IP address of the tunnel. The End User and the Security Gateway MUST change this outer IP address from EU @IP_outer3. The End User MUST modify its Security Association so that packets sent to the Security Gateway are using a valid IP address. Similarly, the Security Gateway MUST update its Security Association so that it can send packets to a reachable destination IP address. The notification of the update is performed using the IKEv2 channel, that is to say in our case EU @IP_outer1.

Figure 3 illustrates the case where EU @IP_outer4 is using the same network hardware interface as EU @IP_outer3. This corresponds to the

case where, for example, the End User decides to use EU @IP_outer4 instead of EU @IP_outer3 on the same hardware network interface. Other mobility use cases may also consider the EU @IP_outer4 may be associated to a different network hardware, including the one associated to EU @IP_outer(i), i in {1, 2}. Then, EU @IP_outer4 is different from EU @IP_outer3 but may be one of the EU @IP_outer(i), i in {1, 2}.

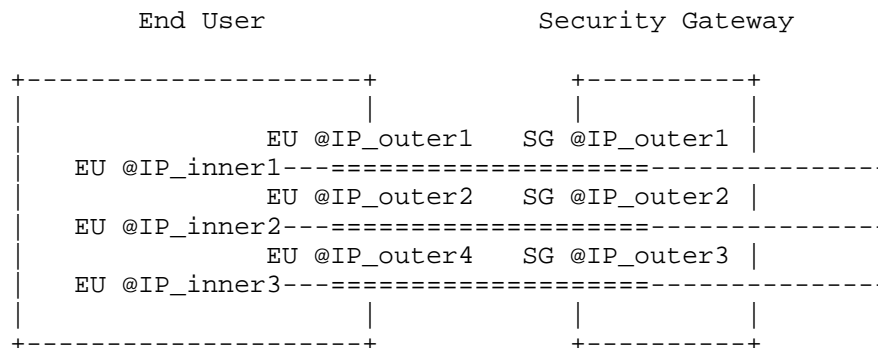


Figure 3: VPN Mobility

3.2.2. Problem Statement

Currently IKEv2 proposes different alternative to update a Security Association, and modify the outer IP address of the Tunnel. However none of them really address the description provided in Section 3.2.1

3.2.2.1. MOBIKE

MOBIKE [RFC4555] provides an UPDATE_SA_ADDRESSES exchange that updates the outer IP address of the tunnel. As explained in this section MOBIKE cannot be used in the general case described in figure 3 because the updated IP address is necessarily the one associated to the IKEv2 channel. This limitation is due to the fact that MOBIKE has been designed for a single interface.

MOBIKE does not explicitly specify in its message the IP address that has to be updated and the new value for this IP address. The IP address to be updated is the one used by the IKEv2 channel, and the new IP address to consider is the IP address used in the IP header of the UPDATE_SA_ADDRESSES message.

If EU @IP_outer1 is equal to EU @IP_outer3, then sending an UPDATE_SA_ADDRESSES would update the outer tunnel IP address of the Security Associations using the IP address of the IKEv2 channel, that is at least EU @IP_outer1 and EU @IP_outer3, with EU @IP_outer4.

This case is only a specific case and is not applicable when the outer IP address to update is different from the IP address used for the IKEv2 channel.

If EU @IP_outer3 is different from EU @IP_outer1, then, the only way to use MOBIKE is to move the IKEv2 channel to EU @IP_outer3, that is updating EU @IP_outer1 by EU @IP_outer3, and then updating EU @IP_outer3 by EU @IP_outer4. This is not convenient because all traffic on EU @IP_outer1 has been transferred to EU @IP_outer3, and then to EU @IP_outer4. Furthermore, it is only possible for managed mobility, because we need EU @IP_outer3 to be a valid interface until IKEv2 uses EU @IP_outer3. In other words, if EU @IP_outer3 fails suddenly, moving the IKEv2 channel to EU @IP_outer3 is not possible anymore.

As a result MOBIKE cannot be used to handle the mobility described in Section 3.2.1.

3.2.2.2. CREATE_CHILD_SA

A second alternative is to renegotiates a new Security Association between the End User and the Security Gateway. IKEv2 provides the CREATE_CHILD_SA Exchange (Section 1.3 [RFC5996]) to create a new Security Association. Similarly Section 3.1.2 this exchange does not specify the outer IP address of the Tunnel. By default, the outer IP address of the Tunnel is the IP address used for the IKEv2 channel. This does not address the use case described in Section 3.2.1.

If requirements of Section 3.1.3 were fulfilled, that is to say even if the CREATE_CHILD_SA would enable to negotiate the outer IP addresses of the Tunnel, then, using the CREATE_CHILD_SA exchange would be an alternative. However, this alternative would still suffer from several drawbacks:

- Not Mandatory: The CREATE_CHILD_SA is not a mandatory IKEv2 feature, especially for light implementations. For these implementation, an non reachable interface would require re-negotiating both the IKE_SA and the new Security Association. Furthermore, there is currently no way to advertise whether the implementation supports or not this exchange.
- Resource Consuming Exchange: The CREATE_CHILD exchange creates a Security Association from scratch and requires all parameters of the Security Association to be specified. This results in a quite complex exchange, which does not take advantage of the already negotiated parameters, like nonces, Keys, Traffic Selectors, Nonces, SPIs. Instead it requires all these parameters to be renegotiated, generation of nonces, keys, as well as multiple interactions with IPsec databases which requires more resources than updating a single parameter within

- a Security Association.
- Two-Successive Exchange: The CREATE_CHILD exchange creates a new Security Association, however, the previously used Security Association has not been removed from the IPsec databases. As a result, once the new Security Association has been created, a new exchange SHOULD be performed to delete the previous Security Association with the Delete Payload (Section 3.11 [RFC5996]). The Delete Payload specifies the Security Associations to Delete.
- Per Security Association Exchange: The CREATE_CHILD_SA exchange creates a specific Security Association, which means that there are as many CREATE_CHILD_SA exchanges as Security Association to update. In our case, multiple Security Associations may be bound to a single interface, so the Security Association granularity is not convenient for interface management. Updating an interface implies that all Security Association bound to this interface MUST be updated. In the use case illustrated by figure 3, the End User a single Security Association per interface, so interface and Security Association management have similar granularity. On the other end, for the Security Gateway with a single interface, i.e. (all SG @IP_outer(i), i in{1, 2, 3} are the same), interface and Security Association do not have the same granularity. Note that with a single interface the Security Gateway would be able to use MOBIKE, but not with two interface (i.e. is SG @IP_outer2 and SG @IP_outer3 would be the same).

3.2.2.3. One IKE channel per Interface

A fourth alternative consists renegotiating an complete independent IKEv2 channel and a new Security Association. This is out of the scope of this document. This may result as having a IKEv2 channel per interface. Furthermore, independent IKEv2 channels may not simplify IPsec configuration and may result in multiple Security Associations matching a given Traffic Selector, which may cause trouble at least for outbound traffic. Furthermore, in this case, the End User and the Security Gateway must proceed to an authentication.

3.2.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 3, IKEv2 SHOULD make possible:

- 1. To update the outer IP address of the tunnel with a IP address that differs from those used for the IKEv2 channel. The Update is not a per security Association negotiation but SHOULD replace all Security Association associated to the old IP address. For all these Security Associations, the old IP

address is replaced by the new IP address. This consists in extending MOBIKE UPDATE_SA_ADDRESSES exchange.

3.3. Multihoming

3.3.1. Description

This section considers how a node can take advantage of multiple interfaces with multihoming. In case one of these interface fails, then another interface can be used instead. Moving the traffic from one interface to the other is called mobility. This section deals with multihoming, that is the two peers agree that in case an interface fails, a mobility should be triggered on the agreed interface.

Suppose, as represented in figure 4, EU @IP_outer3 is not reachable anymore. Applications that are multiple interfaces aware, and also bound to the others EU @IP_inner(i) (i in {1, 2}) IP addresses may handle EU @IP_outer3 non reachability. On the other hand non multiple interfaces aware applications (like regular TCP connections) bound to EU @IP_outer3 are stalled and cannot use the other interfaces.

One way to recover the EU @IP_inner3 unreachability is to reconfigure the Security Association and replace EU @IP_outer3 by EU @IP_outer(i) (i in {1, 2}). Figure 5 shows that EU @IP_outer3 is replaced by EU @IP_outer2. EU @IP_outer2 has been provided as an Alternate IP address of EU @IP_outer3. This means that when one or the other peer notice EU @IP_outer3 is down, it can trigger a mobility with the appropriated outer IP address. More specifically, the Security Gateway can overcome the failure of EU @IP_outer3, if it detects the failure before the End User. The End User and the Security Gateway can also agree on an ordered list of Alternate IP addresses.

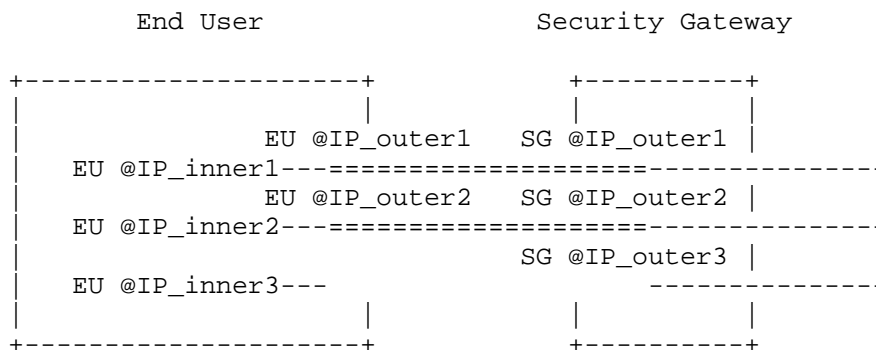


Figure 4: VPN with Mobility/Multihoming between

Multiple Interfaces: EU @IP_outer3 unreachable

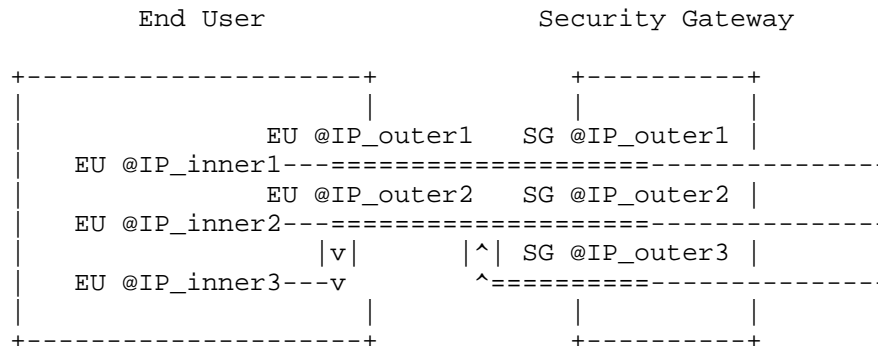


Figure 5: VPN with Mobility/Multihoming between Multiple Interfaces: EU @IP_outer2 replaces EU @IP_outer3

3.3.2. Problem Statement

Currently Multihoming is handled by MOBIKE with the ADDITIONAL_IP*_ADDRESS Notify Payloads. As with mobility, these payloads are only provided for the interface used by the IKEv2 channel. The main reason is that MOBIKE has been designed for a single interface. In our case MOBIKE would only make possible to provide Alternate IP addresses to EU @IP_outer1.

What happens to packets when the Security Gateway performs Multihoming and the End User has not updated its Security Association? Both End User and Security Gateway Security Associations are configured to use the EU @IP_outer3 IP address. When the Security Gateway notices EU @IP_outer3 is not reachable it updates its Security Association, triggers a mobility exchange and may start sending packets to EU @IP_outer2 before the End User has proceeded to the update of its Security Associations. The End User receives this packet and performs a Security Association match. Outer IP addresses will not performed a match, and the match occurs with the Security Policy Index (SPI). The packet is checked against the Security Policy Databases Selectors. These selectors are based on the inner IP addresses and have not been modified. As a result, packets will not be discarded.

3.3.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 3, IKEv2 SHOULD make possible:

- 1. To provide Alternate IP addresses for IP addresses that are different from the one used by the IKEv2 channel. This extends the Multihoming features of MOBIKE to multiple interfaces.
- 2. Reduce the complexity of Multihoming. Although a node **MUST** be able to provide Alternate IP address for a given IP address, it should also be able to provide all its interfaces, and if multihoming is supported on both side, a multihoming rule should be derived by default from this list.

3.4. Adding an Interface

3.4.1. Description

Nodes with multiple interfaces may have some interfaces supporting the VPN whereas other interfaces have not been assigned an IP address. When this interface has been assigned an IP address, the current VPN communication may take advantage of this newly available interface. This section is concerned on how a given communication can take advantage of a newly available interface and set its IPsec settings in an optimal way.

Figure 6 represents the End User with multiple interfaces connected to the Security Gateway. We only represented a single interface for the Security Gateway but more interfaces may be also considered. In figure 7, the Security Gateway has an additional interface that becomes active, it advertises the End User this interface is available. The End User may perform some latency and Round Trip Time measurements and decide to use it. In the figure 7, the End User moves the traffic associated to its interface EU @IP_outer3 to the newly available interface SG @IP_outer2 of the Security Gateway. Moving the traffic is performed through a mobility operation as described in Section 3.2.

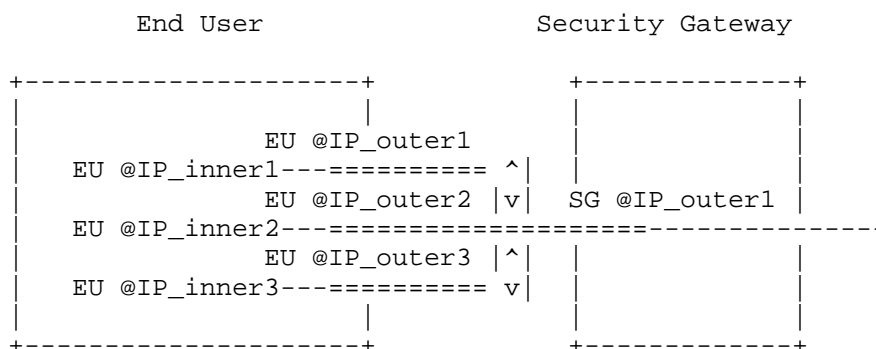


Figure 6: Security Gateway with a single Interface

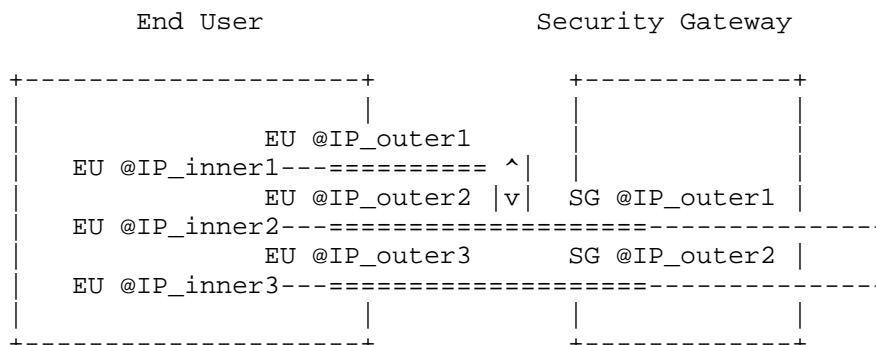


Figure 7: Security Gateway adding an Interface
New Interface used by the EU @IP_outer3

Figure 6 and 7 illustrated the case, where the Security Gateway has an additional active interface. In this case, the Security Gateway let the End User decide which interface they prefer to use. By announcing the newly available interfaces no new Security Associations are created. On the other hand, the End User may also want that any service using the other interfaces can use this newly available interface. This requires to derive the Security Associations associated to the new interface from those associated to the already established interfaces. The Security Associations derived for the newly active interface are not created from scratch with a complete negotiation. This case is illustrated by figure 8 and 9.

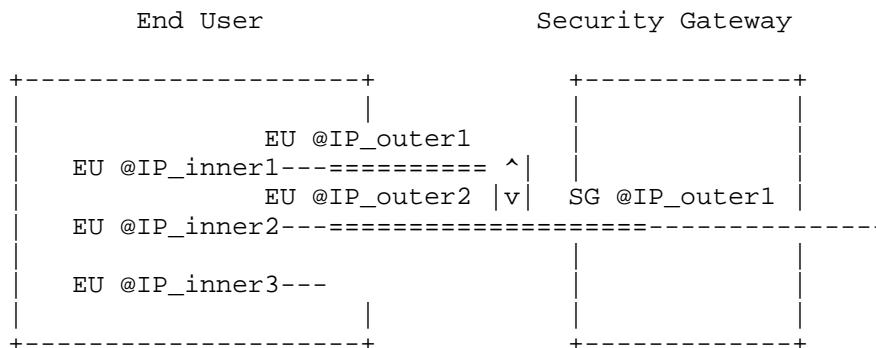


Figure 7: End User with an inactive interface

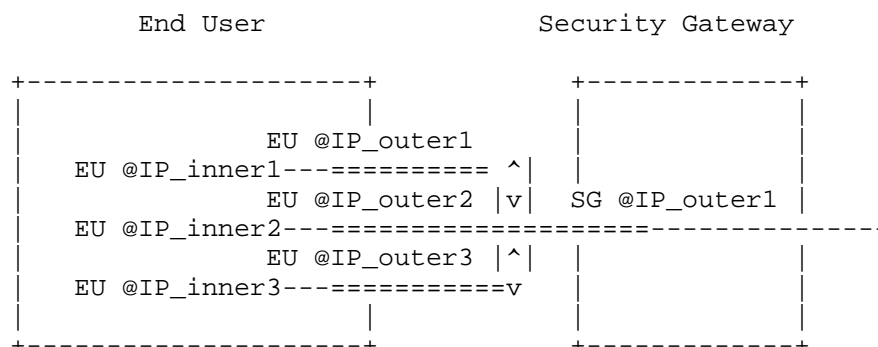


Figure 8: End User with a newly active interface EU @IP_outer3. All traffic associated to EU @IP_outer1 and EU @IP_outer2 is able to use EU @IP_outer3

In our case, the End User already had a specific inner IP address associated to the newly available interface EU @IP_outer3. This makes possible the End User to generate the new IPsec Security Associations and new Security Policies associated to EU @IP_outer3. When the Security Gateway receives the request to add the newly available interface, it may set the newly Security Policies and Security Associations. However, the End User may not have an inner IP address EU @IP_inner3, and may combine the request to the Security Gateway to add the new interface, with a request for a EU @IP_inner3 address. In that case, the Security Gateway first sets the IPsec databases, and the End User sets the IPsec databases when it receives the inner IP address.

When an interface is added, unless otherwise specified, the End User wants that all services, except IKEv2 using the available outer IP addresses (EU @IP_outer1 and @IP_outer2 addresses) may also be configured to use the newly available IP address EU @IP_outer3. By adding an interface the End User is not using a finer granularity than the interface granularity. In other words, it does not want to specify how Security Associations are derived. They should be derived in an automatic way. In return, deriving Security Associations and Security Policies is expect to optimize their creation as opposed to using CREATE_CHILD_SA.

In the example of figure 7 and 8, the End User is likely to create Security Associations derived from those established with the interfaces EU @IP_outer1 and EU @IP_outer2. All services using EU @IP_outer1 or EU @IP_outer2 will be able to use EU @IP_outer3 with the inner IP address EU @IP_inner3.

The idea is to copy the Security Association associated with EU @IP_outer1 replace EU @IP_outer1 by EU @IP_outer3 and EU @IP_inner1 by EU @IP_inner3. SPIs MUST also be changed since there are unique for the Security Association. Then we perform the same with EU @IP_outer2.

Note that it is important to specify an ordered list of EU @IP_outer address from which the new SAs are derived, so to guarantee that these new Security Associations are derived the same way on both peers. Then the new Security Association MUST be created only if there are no already existing matching SPD selectors.

In the most basic case of VPN, we only have one Security Association per interface. All services using EU @IP_inner(i) are tunneled to EU @IP_outer(i) i in{1,2}. Adding EU @IP_outer3 only requires to derive Security Association from one interface EU @IP_outer1 and EU @IP_outer2. Then, the End User needs to specify the inner and outer IP addresses EU @IP_inner3, EU @IP_outer3 and in the specific case represented on figure 7 the outer IP address of the Security Gateway SG @IP_outer3. The resulting exchange may look something like the exchange represented in figure 10. The mandatory parameters are the IP address used for the traffic selectors, and the outer IP address for the Tunnel on the End User. The destination outer IP address of the Tunnel is optional and, if not specified may be the one used by the IKEv2 channel. The list of interfaces from which are derived the Security Associations and the Security Policies may also be optional. A default value for this list may be the ordered list of associated outer IP addresses of the End User. The nonce may be used to create SPIs.

	End User	Security Gateway
request	Add Interface (EU @IP_inner3, ---> EU @IP_outer3, [outer-destination] [interface-list] [nonce])	
normal case		<--- N()
error case		<--- N(error)

Figure 10: Principle of the Adding Interface exchange

3.4.2. Problem Statement

Currently IPsec does not provide any means for a peer to advertise a new interface is available. MOBIKE makes possible to advertise a Alternate IP address is available. However Alternate IP addresses

are only intended to be use in case the Primary Interface is down. In our case, the interface is ready for use. This issue is similar to the one detailed in Section 3.1.2. However, here the announcement corresponds to a dynamic changes, and the list of available IP address does not occurs during the IKE_INIT exchange, but in a regular information exchange.

Currently the only way IKEv2 provides to create new Security Associations is the CREATE_CHILD_SA exchange. Disadvantages of this exchange have been described in Section 3.2.2. The key advantage of adding an interface is to provide an optimized interface management exchange instead of a Security Association management exchange.

3.4.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible:

- 1. Make possible the Responder and Initiator to announce its interfaces outside the IKE_INIT exchange. This requirements is similar to the one of Section 3.1.3
- 1. Make possible the Responder and Initiator to automatically derive Security Associations and Security Policies from the existing interface.

3.5. Deleting an Interface

3.5.1. Description

Nodes with multiple interfaces in dynamic environment may have interfaces that are not reachable anymore. This may trigger mobility or multihoming actions. However, the node may also want to delete the Security Associations bound to this interface either as a Tunnel outer IP address or as a Traffic Selector.

3.5.2. Problem Statement

Currently IKEv2 does not make possible to delete an interface from multiple Security Associations. IKEv2 provides a Delete Payload (Section 3.11 [RFC5996] that deletes one or multiple specific Security Associations, identified by their SPI.

3.5.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 3, IKEv2 SHOULD make possible:

- 1. Delete an interface, that is to say all Security Associations associated to that interface.

4. Use Case 2: MIF applications and IPsec Tunnel mode

This section considers applications that can deal with multiple interfaces. This ability can be done with transport layer protocols like MPTCP or SCTP or with applications using one or multiple UDP / TCP connections over the various interfaces, and that manages how to send the data.

The difference between multiple interfaces aware applications and the VPN use case is that the tunnels are established per services, whereas the VPN tunnel all traffic is tunneled to a unique Security Gateway. This may increase the number of Security Associations between the End User and the Security Gateway. This section details motivation for using the IPsec Tunnel mode with multiple interfaces aware applications and position it to the VPN use case of Section 3.

Applications may use the tunnel mode for end-to-end security and to benefit from the Mobility features provided by the Tunnel mode. More specifically, using the Tunnel mode provides Mobility without breaking the connectivity, if upper layer is not mobility aware.

Other motivations for using the Security Gateway is that the End User chose not to tunnel all its traffic to the Security Gateway, but only the traffic that worth being protected. For example, an End User may chose not to tunnel its "youtube" traffic, as well as some of its "https" traffic (as well as its application layer protected traffic). On the other hand, it may want to tunnel all non-protected "http" (as well as other non protected communications).

If each service proposes different Security Gateways, the use case is very similar to the VPN use case, for each service. The main difference is that Security Association are established with different Traffic Selectors.

If multiple services are using the same Security Gateway, this will result for each interface, in multiple Security Associations established with the same Security Gateway - one per service. This case is very similar to the VPN use case but with multiple Security Associations. If "s" is the number of Services connected on the Security Gateway the number of Security Associations is at least "s" (5services are considered independent). If some applications are using multiple flows, then this number may be even larger. In that case, adding an interface results in at least negotiating "s" new Security Associations. Using the CREATE_CHILD_SA exchange may

require "s" exchanges whereas using the Adding interface exchange requires only one exchange. This use case is represented in figure 11.

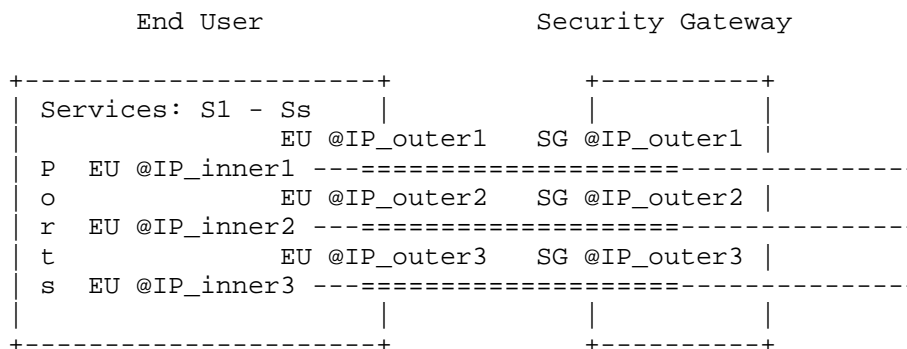


Figure 11: MIF aware applications

Requirements of this use case have already been mentioned in the VPN use case.

5. Use Case 3: MIF aware applications with Transport mode

This Use Case is very similar to the Use Case 2 except that the Transport mode is used instead of the Tunnel mode. The Use Case is illustrated with figure 12.

Unlike in the VPN use case in Section 3 or for multiple interfaces aware applications described in Section 4 using IPsec tunnel mode, the IPsec Transport mode does not involve inner IP addresses.

With Transport mode, we may consider two types of applications. The applications that can handle multiple interfaces. This can be done with transport protocols like MPTCP or SCTP or with a connection manager at the application layer. These applications may have Security Associations on all interfaces. Other Applications with a single using TCP/UDP and without specific connection managers may only deal with a single interface and may only have an Security Association associated to this interface.

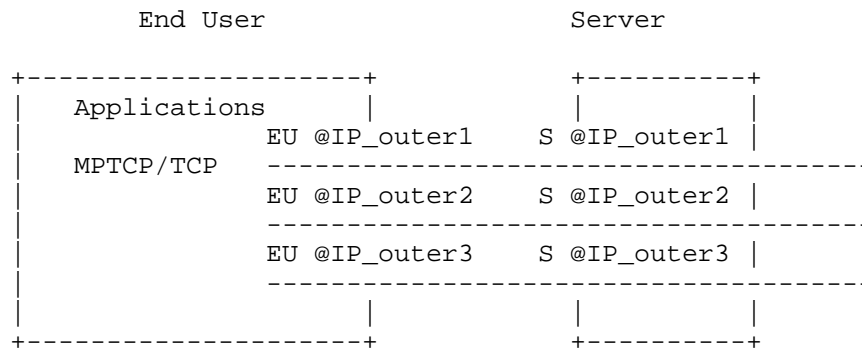


Figure 12: MIF aware applications with the Transport mode

5.1. Initial MIF IPsec Configuration

5.1.1. Description

In Figure 12, the End User initiates an IKEv2 negotiation using EU @IP_outer1 and S @IP_outer1. The Server provides the End User the available interfaces (S @IP_outer1 i in {1, 2, 3}). Then the End User negotiates Security Associations between the EU @IP_outer(i) and S @IP_outer(i) i in {1,2,3} using different Traffic Selectors.

5.1.2. Problem Statement

Currently IKEv2 does not make possible a node to announce its available interfaces.

The Transport mode, does not involve tunnel outer IP addresses. Current Security Association exchange enables Traffic Selectors negotiation. These Traffic Selectors are used both for the Security Policy Index (Traffic Selectors) for outgoing traffic and for the Security Association Index for incoming traffic. Current IKEv2 specification enables to set IPsec as described in figure 11.

5.1.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible

- 1. Make possible the Responder and Initiator to announce its interfaces. This requirement is similar to the requirements for VPNs.

5.2. Mobility

With regular TCP connection a change of the IP address breaks the connection. Applications may use mobility with the Transport mode with transport protocols that handles with multiple interfaces (like MPTCP or SCTP for example), with multiple independent TCP/UDP connections on the different interfaces. The application manages its connections at the application layer.

Mobility with Transport mode MUST be understood as updating an existing Security Association. The purpose of the IPsec Mobility and the Transport mode is to avoid to create a new Security Association when the IP address of an interface is changing. IPsec configures the layer so that the application can securely go on with its communications. TCP connections are restarted, since changing the IP address will most likely break the existing connection. UDP will start sending on the other interface. Mobility is intended to reduce the time IPsec requires to configure its Security Associations.

With the Tunnel mode, IPsec was in charge of securing and transporting IP datagrams. With the Transport mode, IPsec only secures the communication. Transport of the IP datagrams is shared between the application and the transport layer. Application and IPsec layers are independent and have their own way to handle with mobility. Synchronization between these two layers MUST be performed to avoid that the application moves the traffic on an interface whereas IPsec DISCARD this traffic. Although we do not intend to provide a complete list of how to synchronize these two layers, the list below provides some example where these two layers are synchronized:

- 1. For End Users with two interfaces. In that case, the interface the application may use is determined.
- 2. For applications that are configured with two interfaces.
- 3. For applications that we know the interface they will choose. Like those setting priority to interfaces. This could be set by using Multihoming and ordering the Alternate IP addresses.
- 4. If the Mobility exchange is triggered by the new socket, new packet sent. This case reduces the latency over a CREATE_CHILD_SA exchange, but does not anticipate the decision of the application.

5.2.1. Description

The mobility scenario we consider in this section is an application using a single interface EU @IP_outer3 for example. As represented in figure 13, this interface is down. Then the End User get assigned a new IP address EU @IP_outer4 and uses this interface as represented in figure 14. Both End User and Server MUST update Security Policies

and Security Associations that used EU @IP_outer3 and replace the value with EU @IP_outer4. Unlike the Tunnel mode, Traffic Selectors also need to be updated.

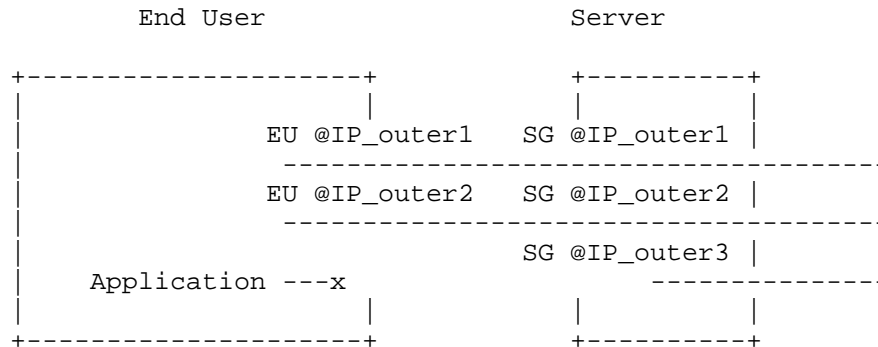


Figure 13: Mobility with Transport mode and Multiple Interfaces: EU @IP_outer3 unreachable.

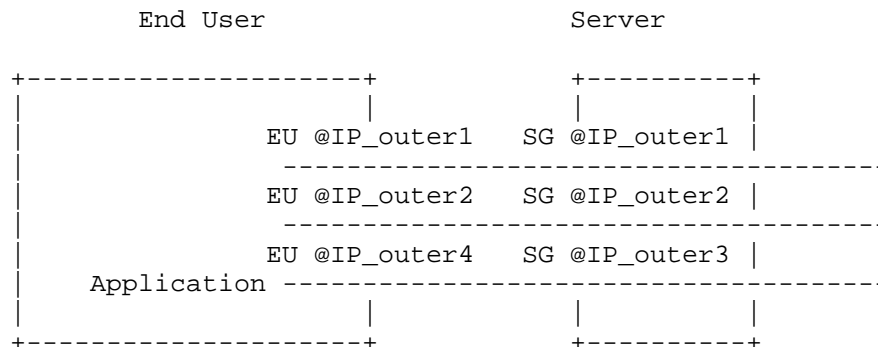


Figure 14: Mobility with Transport mode and Multiple Interfaces: EU @IP_outer4 replaces EU @IP_outer3.

5.2.2. Problem Statement

Currently IKEv2 does not provide extension that perform any mobility operation.

MOBIKE has only been designed for the Tunnel mode.

The CREATE_CHILD_SA suffers for limitations exposed in Section 3.2.2: It is not mandatory in IKEv2 implementation, the exchange requires

much resources as updating the Security associations. Most of the time, it requires an addition Delete exchange and is a per Security Association exchange. However, because no tunnel IP address requires to be negotiated, the CREATE_CHILD_SA can set the Security Associations and Policies as described in figure 14.

5.2.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible

- 1. Extend MOBIKE to the Transport mode
- 2. Extend MOBIKE with Transport mode to multiple interfaces requirements described in Section 3.2.3.

5.3. Multihoming

Multihoming consists in providing Alternate Interfaces in case a running interface is down, so peers are aware of the parameters to update. Multihoming can be seen as pre-configuring an mobility operation.

5.3.1. Description

With Multihoming, when the End User sets its IPsec configuration as illustrated in figure 12, the End User also specifies for each interface the corresponding Alternate IP address. Although this can be done on a per interface value, we suggest that when multiple interfaces are provided, Alternate IP addresses can be derived automatically and assigned to each interface without being explicitly mentioned. Suppose that in the case of figure 13, for example EU @IP_outer2 is provisioned as the Alternate IP address of EU @IP_outer3.

When EU @IP_outer3 is down, then the End User or the Server triggers a mobility exchange as described in section Section 5.2.1.

5.3.2. Problem Statement

Currently IKEv2 does not make possible to provision Alternate IP addresses for the Transport mode. MOBIKE has only been designed for the Tunnel mode, then as mentioned in Section 3.3.2, MOBIKE only assigns the Alternate IP address for the IP address used by the IKEv2 channel. This is because MOBIKE has been designed for a single interface.

Note that with the Transport mode, the Alternate Address is provided to the outer IP address that is also used as a Traffic Selector, whereas in the Tunnel mode, the Alternate IP address is provided for

the tunnel outer IP address.

Note also that the IKEv2 channel is a special case where Alternate Address is associated to the Transport mode. In fact the IKEv2 channel uses Transport mode, not the Tunnel mode.

5.3.3. Requirements

In order to make the End User set its IPsec configuration as represented in figure 1, IKEv2 SHOULD make possible to:

- 1. Extend MOBIKE Multihoming to the Transport mode
- 2. Extend MOBIKE with Transport mode to multiple interfaces requirements described in Section 3.3.3. Alternate IP address should be assigned to any interface and can be automatically be derived. Alternate IP address concerns Traffic Selectors and Security Association Indexes.

5.4. Adding an Interface

Adding an interface works exactly as described in Section 3.4. The only difference is that when an interface is added with the Transport mode, Traffic Selectors will automatically be associated to this newly added interface, which was not necessarily the case with the Tunnel mode.

5.5. Delete an Interface

Similarly to the addition of a new interface, Deleting an interface works exactly as described in Section 3.5. The only difference is that with the Transport mode, Security Associations and Security Policies to delete are these where the specified interface appears as a Traffic Selector rather than as an outer tunnel IP address.

6. Security Considerations

The whole document sets MIF requirements for a security protocol.

7. IANA Considerations

There is no IANA consideration here.

8. Acknowledgment

We would like to thank Daniel Palomares, Pierrick Seite, Brian Carpenter, Hui Deng, Jong-Hyouk Lee, Juan Carlos Zuniga and

Konstantinos Pentikousis for their useful comments.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

9.2. Informational References

- [Cisco] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010-2015", February 2011.
- [I-D.arora-ipsecme-ikev2-alt-tunnel-addresses] Arora, J. and P. Kumar, "Alternate Tunnel Addresses for IKEv2", draft-arora-ipsecme-ikev2-alt-tunnel-addresses-00 (work in progress), April 2010.
- [RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.
- [Raiciu] Arora, C., Barre, S., Plunkte, C., Greenhalgh, A., Wischik, D., and M. Handley, "Improving datacenter performance and robustness with multipath TCP", SIGCOMM 2011 Toronto, Canada, August 2011.

Authors' Addresses

Daniel Migault
Francetelecom - Orange
38 rue du General Leclerc
92794 Issy-les-Moulineaux Cedex 9
France

Phone: +33 1 45 29 60 52
Email: mglt.ietf@gmail.com

Carl Williams
MCSR Labs
Philadelphia, PA 19103
USA

Phone: 650-279-5903
Email: carlw@mcsr-labs.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2013

X. Zhang
Juniper Networks
T. Tsou
Futurewei Technologies
W. Liu
Huawei Technologies
October 22, 2012

Multiple Path IP Security
draft-zhang-ipsecme-multi-path-ipsec-02

Abstract

This document presents one approach to enhance data protection when transmitting IPsec datagrams across the insecure networks. The method affords the stronger protection to the traffic by splitting it among a set of sub-tunnels. All the Security Associations (SAs) are set up independently for all sub-tunnels. Both the sending and receiving entity combine all the sub-tunnels to one clustered tunnel. As different sub-tunnel uses different crypto key materials and processing parameters, it may achieve the stronger protection of the traffic across the insecure networks. In addition, it could possibly bring more benefits in terms of the network control.

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 April 25, 2013.

Copyright Notice

Copyright (c) 2012 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
2. Terminology	3
3. Multiple Path IPsec	3
3.1. The SA setup	4
3.2. The outbound packet processing	4
3.3. The inbound packet processing	5
3.4. The SA expiration	5
3.5. Multiple paths	5
3.6. Interoperability	5
3.7. Reorder packets	6
4. The benefit for SA cluster	6
5. Acknowledgements	6
6. Security Considerations	6
7. IANA Considerations	6
8. References	7
8.1. Normative References	7
8.2. Informative References	7
Authors' Addresses	7

1. Introduction

IPsec protocols suite specifies the base architecture for IPsec-compliant systems. It describes how to provide a set of security services for traffic at the IP layer, in both the IPv4 and IPv6 environments. It defines security association (SA) as the fundamental concept to IPsec, which defines a simplex "connection" that affords security services to the traffic carried by it. Security services are afforded to an SA by the use of AH [RFC4302], or ESP [RFC4303], but not both. If both AH and ESP protection are applied to a traffic stream, then two SAs must be created and coordinated to effect protection through iterated application of the security protocols.

Since one SA is used to carry uni-cast traffic, a pair of SAs must be established in point-to-point communication. The two SAs create one uni-cast IPsec tunnel between two security gateways. In order to differentiate different SAs, the Security Parameters Index (SPI), one 32-bit value, is used by a receiver to identify the SA to which an incoming packet should be bound. The SPI assignment is done at the creator of the SA, or usually the receiving side. At the sending side, additional destination IP address information can be used to resolve the SPI conflict. In this way, the sending side can select the correct SA under which IP packet will be processed. In this document, the new method also makes use of multiple SPIs. Nevertheless, it enhances the security service in different way from SA.

2. Terminology

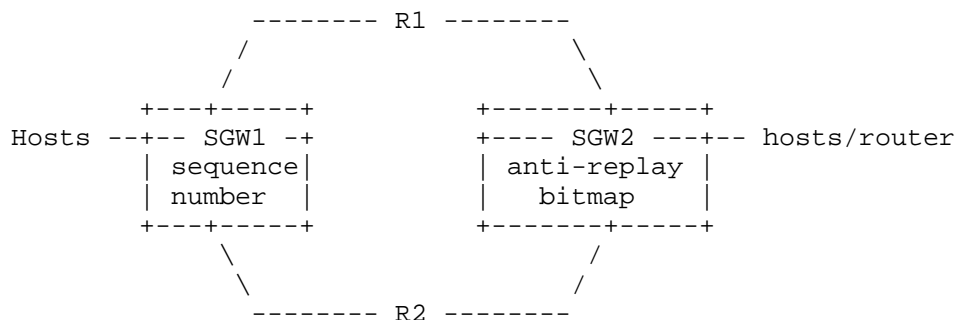
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Multiple Path IPsec

Data confidentiality is the protection of transmitted data from passive attacks, such as eavesdropping. In current IPsec implementation, all the IP datagrams transmitted inside one IPsec tunnel are afforded protection by one SA. In order to enhance the confidential security service, we use a set of SAs to protect the traffic. We propose to set up multiple tunnels between two entities and then cluster them together to form one clustered tunnel. One IP packet is still protected by one single SA. The sending entity just splits the traffic among all these SAs. The receiving entity must multiplex the traffic from the different IPsec tunnels. All these

tunnels clustered together are termed "sub-tunnels". The SAs for these sub-tunnels are termed "sub-SA". The IP traffic, which should be protected inside one clustered tunnel, is split among all the sub-tunnels. The term "security association cluster", or "SA cluster", is used to describe the combination of SAs through which the traffic must be processed to satisfy a security policy.

As multiple sub-tunnels are set up for the same flow of traffic between two secure entities, the physical paths may be different. The processing order of these clustered SAs is only local matter as all these SAs are not nested SAs.



3.1. The SA setup

The SA cluster setup consists of multiple sub-SA setups. All these sub-tunnels are set up independently. After setup, the sub-tunnel can be added to the cluster one by one. But it is the local matter as how to add the sub-SAs into the SA cluster. All the collaborative sub-tunnels have different SPI values. There is no limitation on how many sub-tunnels can be used for one clustered tunnel. Both the sending entity and receiving entity agree on SA cluster which will be used before any IPsec traffic goes through any of these sub-tunnels. After the traffic flows inside clustered tunnel, new SA can still be able to set up and join the SA cluster.

Even though all the sub-tunnels are independent, they share only one sequence number source. The IPsec packet carried inside the clustered tunnel has unique sequence number.

3.2. The outbound packet processing

The sending entity splits or alternates the IPsec traffic through different sub-tunnels. When the SA cluster is selected for the traffic processing based on security policy configuration, one sub-SA is chosen for outbound IPsec processing only for that packet. It is the local implementation that determines which SA should be applied

to the specific IP packet. Except that the sequence number is shared among all sub-SAs, all the other processing procedures are not altered. A local implementation at sending entity can choose any method to obtain the sequence number for this packet, which is independent of sub-SA.

3.3. The inbound packet processing

The selection of sub-SA is the same as the selection of single SA, which is based on SPI and IP address information. Except that the sequence number processing is a bit different, all other aspects are not changed.

With the use of multiple sub-tunnels, by its nature, it could cause out-of-order delivery of IPsec packets for the secure communication channel between two entities. As the remedy, the sequence number in IPsec header can be used if the receiving entity needs to maintain the sending order.

If anti-replay is enabled, all these sub-tunnels will use one shared anti-replay bitmap at the receiving entity. The anti-replay check is done against the SA cluster instead of sub-SA. But it does not change how anti-replay is done.

3.4. The SA expiration

If sub-SA is negotiated through IKE negotiation, it may have its own soft and hard lifetime. But there is no lifetime for SA cluster. There is no change as to maintenance of each sub-SA.

If one sub-SA becomes invalid, it could not be used for further packet processing. If SA cluster does not hold any valid sub-SA, it becomes invalid too.

3.5. Multiple paths

All these sub-tunnels are set up independently. The traffic through the different sub-tunnels can go the same route. It can also go the different routes based on the routing policy. The path selection algorithm is out the scope of this document.

3.6. Interoperability

In case that SA cluster contains only one sub-SA, it must not have any interoperability issue with the current IPsec implementation if the current one does not support SA cluster.

3.7. Reorder packets

The solution of multipath introduces the issue of the possibility of out of order delivery. Actually, this is the only solution which causes the disorder problem. Even with single SA, it can also bring in the out of order problem. Technically, the reorder process can be done at aggregate node or end host, based on the topology of network, just like TCP reorder or IP reassembly [RFC5236][Zhang02]. The reorder algorithm is out the scope of this document.

4. The benefit for SA cluster

The method enhances the security service by spreading the traffic onto multiple paths. For example, it makes it harder for the attacker to intercept all the packets if different routes are used. Even with the same route used, it is harder for the attacker to know which set of SAs are clustered SA, thus harder to decrypt the intercepted packets. With multiple paths selected, it provides high reliability especially in case of link failure. It also provides the option for optimized performance and optimal network control, which is not covered in this document.

5. Acknowledgements

Wait for comments.

6. Security Considerations

This document intends to enhance the security service which IPsec provides. SA cluster provides the option to perform the different cryptographic transformation on the different packet. In addition, it also provides the option to transmit the packets along the different paths.

7. IANA Considerations

None.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

8.2. Informative References

- [Piratla06] N. M. Piratla, et al., "Reordering of Packets due to Multipath Forwarding - An Analysis", 2006.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.
- [RFC5236] Jayasumana, A., Piratla, N., Banka, T., Bare, A., and R. Whitner, "Improved Packet Reordering Metrics", RFC 5236, June 2008.
- [Zhang02] M. Zhang, et al., "Improving TCP's Performance under Reordering with DSACK", 2002.

Authors' Addresses

Xiangyang Zhang
Juniper Networks
1194 N. Mathilda Ave
Sunnyvale, CA 94089
USA

Email: vzhang2008@yahoo.com

Tina Tsou
Futurewei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1 408 330 4424
Email: tina.tsou.zouting@huawei.com

Will Liu
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: liushucheng@huawei.com

