

Guidelines for Implementation and
Priorities in Testing
IKEv2

Technical Document

Version 1.1.0

IPv6 Promotion Council
Certification WG
IPsec SWG

*All Rights Reserved. Copyright (C) 2008-2010
NTT Network Service Systems Laboratories*



Modification Record

Version 1.1.0 May 20, 2010

- RFC4868 and RFC5114 was added to “Related standards”.
- The condition of pseudo-random function of IKE SA changed.
BASIC : PRF_HMAC_SHA1
ADVANCED : PRF_AES128_XCBC, PRF_HMAC_SHA2_256
- The condition of Integrity Algorithm of IKE SA changed.
BASIC : AUTH_HMAC_SHA1_96
ADVANCED : AUTH_AES_XCBC_96, AUTH_HMAC_SHA2_256_128
Not Supported : AUTH_HMAC_MD5_96
- The condition of Diffie-Hellman Group of IKE SA changed.
BASIC : Group 2
ADVANCED : Group14, Group24
- The condition of Integrity Algorithm of CHILD SA changed.
BASIC : AUTH_HMAC_SHA1_96
ADVANCED : AUTH_AES_XCBC_96, AUTH_HMAC_SHA2_256_128
Not Supported : AUTH_HMAC_MD5_96
- RFC4306 page 51 line 2825 : test requirement changed to ADVANCED because PRF_HMAC_SHA2_256 is ADVANCED algorithm.
- RFC4306 page 51 line 2839 : test requirement changed to ADVANCED because AUTH_HMAC_SHA2_256_128 is ADVANCED algorithm.
- RFC4306 page 51 line 2852 : test requirement changed to ADVANCED because Group 24 is ADVANCED group.

Version 1.0.2 Mar 25, 2010

- The condition of ID types Receiving and Sending changed.
BASIC : IPV6_ADDR
ADVANCED : FQDN , RFC822_ADDR ,
Not Supported : IPV4_ADDR, KEY_ID
- The condition of encryption algorithms of IKE SA changed.
BASIC : ENCR_3DES
ADVANCED : ENCR_AES_CBC
- RFC4306 page 4 line 189 and 199 : test number changed because EN.I.1.3.2.1 and SGW.I.1.3.2.1 were removed.
- RFC4306 page 11 line 577, 581 and 590 :
target and test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 12 line 638 : target and test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 13 line 720 : test number changed because EN.I.1.3.2.1 and SGW.I.1.3.2.1 were removed.
- RFC4306 page 15 line 830 :
target and test number changed because EN.I.1.1.3.4, EN.R.1.1.3.3, SGW.I.1.1.3.3 and SGW.I.1.1.3.4 were removed.
- RFC4306 page 15 line 834 and 836 : test number changed because EN.I.1.1.3.5 and SGW.I.1.1.3.5 were removed.
- RFC4306 page 17 line 906 : test number changed because EN.I.1.1.3.7 and SGW.I.1.1.3.7 were removed.
- RFC4306 page 18 line 968, 969 and 987 : test number changed because EN.I.1.3.1.1 and SGW.I.1.1.3.1 were removed.
- RFC4306 page 19 line 1035 test requirement changed to Not Support because of untestable.
- RFC4306 page 19 line 1046 target and test number changed because EN.R.1.1.5.1 and SGW.R.1.1.5.1 were removed.
- RFC4306 page 20 line 1070 target and test number changed because EN.R.1.1.5.1 and SGW.R.1.1.5.1 were removed.
- RFC4306 page 21 line 1164 : test requirement changed to ADVANCED because DH#14 is ADVANCED group.
- RFC4306 page 28 line 1520, 1522, 1525, 1542, 1544 and 1584 :
test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 29 line 1586 : test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 36 line 1986 :
test number changed because EN.I.1.1.6.8, EN.I.1.1.8.1, EN.I.1.1.8.2, EN.I.1.3.4.1, EN.R.1.1.8.1, EN.R.1.1.8.2, EN.R.1.1.8.3, SGW.I.1.1.6.8, SGW.I.1.1.8.1, SGW.I.1.1.8.2, SGW.I.1.3.4.1, SGW.R.1.1.8.1, SGW.R.1.1.8.2 and SGW.R.1.1.8.3 were removed.
- RFC4306 page 36 line 1988, 2002 and 2005 : test requirement changed to Not Support because of untestable.
- RFC4306 page 41 line 2263 and 2277 : test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 42 line 2302, 2320, 2324, 2334, 2342, 2347 and 2361 :
test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 43 line 2376, 2379, 2393 and 2400 :
test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 44 line 2414, 2420, 2430, 2440 and 2459 :
test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 45 line 2492, 2508 and 2511 : test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 45 line 2508 and 2511 : test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 46 line 2560 : test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 48 line 2638 and 2669 : test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 50 line 2754 : test number changed because EN.I.1.3.3.1 and SGW.I.1.3.3.1 were removed.

- RFC4306 page 51 line 2811 :
test number changed because EN.I.1.6.1.1, EN.R.1.6.1.1, SGW.I.1.6.1.1 and SGW.R.1.6.1.1 were removed.
- RFC4306 page 55 line 3049 : test requirement changed to ADVANCED because DH#14 is ADVANCED group.
- RFC4306 page 56 line 3114 : test requirement changed to ADVANCED because DH#14 is ADVANCED group.
- RFC4306 page 57 line 3179 : test number changed because this test is not support expect ID_IPV6_ADDR, FQDN and RFC822_ADDR.
- RFC4306 page 57 line 3183 : test requirement changed to ADVANCED because only with RSA-DSS auth.
- RFC4306 page 58 line 3198 : test requirement changed to ADVANCED because only with RSA-DSS auth.
- RFC4306 page 58 line 3204, 3212, 3217 and 3222 :
test number changed because this test is not support expect ID_IPV6_ADDR, FQDN and RFC822_ADDR,.
- RFC4306 page 58 line 3233, 3235 and 3236 :
test requirement changed Not Support because ID_FQDN and RFC822_ADDR are available only with RSA-DSS auth.
- RFC4306 page 66 line 3678 and 3690 : test requirement change to Not Support because of untestable.
- RFC4306 page 67 line 3705 and 3722 : test requirement change to Not Support because of untestable.
- RFC4306 page 67 line 3733 :
target and test number changed because EN.I.1.1.6.8 and SGW.1.1.6.8 were removed and EN.R.1.1.6.9 and SGW.R.1.1.6.9 were added..
- RFC4306 page 67 line 3737 : test requirement changed ADVANCED to because DH#14 is ADVANCED group.
- RFC4306 page 68 line 3793 : test requirement change to Not Support because of untestable.
- RFC4306 page 69 line 3817 : test requirement change to Not Support because of untestable.
- RFC4306 page 71 line 3926, 3930, 3933 :
target and test number changed because EN.R.1.1.5.1 and SGW.I.R.1.5.1 were removed.
- RFC4306 page 72 line 4011 : test number changed because EN.I.1.1.3.7, SGW.I.1.1.3.7 and SGW.I.1.1.3.8 were removed.
- RFC4306 page 72 line 4025 : test number changed because EN.I.1.1.3.7, SGW.I.1.1.3.7 and SGW.I.1.1.3.8 were removed.
- RFC4306 page 73 line 4038 : test number changed because EN.I.1.1.3.7, SGW.I.1.1.3.7 and SGW.I.1.1.3.8 were removed.
- RFC4306 page 73 line 4042 : test number changed because EN.I.1.1.3.7, SGW.I.1.1.3.7 and SGW.I.1.1.3.8 were removed.
- RFC4306 page 73 line 4045 : test number changed because EN.I.1.1.3.7, SGW.I.1.1.3.7 and SGW.I.1.1.3.8 were removed.
- RFC4306 page 77 line 4276, 4280, 4284 and 4292 :
test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 78 line 4318, 4337, 4345, 4348, 4357 and 4360 :
test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 79 line 4374 and 4381 : test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 86 line 4794 : test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4306 page 86 line 4797 : target and test number changed because EN.I.1.3.1.1 and SGW.I.1.3.1.1 were removed.
- RFC4718 page 4 line 218 : test requirement changed to Not Support because of untestable.
- RFC4718 page 4 line 240 : test requirement changed to ADVANCED because DH#14 is ADVANCED group.
- RFC4718 page 5 line 254 : test number changed because EN.R.1.1.5.1 and SGW.R.1.1.5.1 were removed.
- RFC4718 page 7 line 375 : target and test number changed because EN.R.1.1.5.3 and SGW.R.1.1.5.3 were removed.
- RFC4718 page 26 line 1417 : test requirement changed to ADVANCE because PRF_AES128_XCBC is ADVANCED algorithm.
- RFC4718 page 28 line 1548, 1580 and 1592 : test requirement changed to Not Support because of untestable.
- RFC4718 page 32 line 1766, 1776, 1783 and 1804 : test requirement changed to Not Support because of untestable.
- RFC4718 page 33 line 1811 and 1818 : test requirement changed to Not Support because of untestable.
- RFC4718 page 34 line 1859, 1865 and 1873 : test requirement changed to Not Support because of untestable.
- RFC4718 page 35 line 1913, 1920, 1935, 1946 and 1956 : test requirement changed to Not Support because of untestable.
- RFC4718 page 36 line 1970, 1977, 1980, 1985, 1991, 1994, 1997, 2000 and 2003 :
test requirement changed to Not Support because of untestable.
- RFC4718 page 37 line 2034 : test requirement changed to Not Support because of untestable.
- RFC4718 page 48 line 2646 and 2650 : test number changed because EN.R.1.1.8.3 and SGW.R.1.1.8.3 were removed.
- RFC4718 page 48 line 2661 : test requirement changed to Not Support because of untestable.
- RFC4718 page 56 line 3110 and 3114 :
target and test number changed because EN.R.1.3.1.1 and SGW.R.1.3.1.1 were removed.

Version 1.0.1

June 5, 2009

- The condition of ID types Receiving and Sending changed.
BASIC : IPV6_ADDR
Not Supported : IPV4_ADDR, FQDN , RFC822_ADDR , KEY_ID
- Function of "restarting the entire IKE_SA" is Not Supported.

Version 1.0.0

December 11, 2008

- Initial release.

Table of Contents

| | |
|---|----|
| 1. Overview..... | 5 |
| 2. Scope of the IKEv2 Guidelines for Implementation | 6 |
| 2.1. Reference Network Architecture | 6 |
| 2.2. Related standards | 6 |
| 3. Classification of IKEv2 functions | 8 |
| 3.1. Viewpoints of the classification | 8 |
| 3.2. The method to classify the IKEv2 functions..... | 10 |
| 4. IKEv2 Sequences and Payloads | 19 |
| 4.1. IKEv2 BASIC sequences and payloads..... | 19 |
| 4.1.1. Initial exchange | 20 |
| 4.1.2. Rekey..... | 24 |
| 4.2. IKEv2 ADVANCED sequences and payloads | 28 |
| 4.2.1. Mutual authentication using public signature..... | 28 |
| 5. Priorities of IKEv2 function for testing..... | 32 |

1. Overview

This document gives guidelines for implementing the functions specified in IKEv2 prescribed in IETF to ensure interoperability.

The IKEv2 Test Profile consists of two volumes. One is “Guidelines for Implementation and Priorities in Testing”, which means this document, and the other is “Test Specifications”.

The contents of this document include following items.

- Guidelines for the implementation of the nodes supporting IKEv2
- Specifications of the IKEv2 sequences and payload type in each message between the nodes supporting IKEv2 (i.e. SGW and End-Node)
- Priorities for the testing of each node function according to the function’s importance to interoperability.

This document is in complete accord with the IETF RFC specifications for IKEv2 but includes some extra information for clarification and thus more strongly ensures interoperability.

Term Description

-End-Node

IPv6 host which can terminate IKEv2 protocol.

End-Node is denoted by “EN” in this document.

-Security Gateway

IPv6 node including a router or a firewall that intermediate system which support IKEv2 protocols.

Security Gateway is denoted by “SGW” in this document.

2. Scope of the IKEv2 Guidelines for Implementation

2.1. Reference Network Architecture

Figure 2-1 shows the network architecture covered by IKEv2 Guidelines for Implementation.

- I/F1 is an interface that showed the protocol confirmation between EN and EN.
- I/F2 is an interface that showed the protocol confirmation between EN and SGW.
- I/F3 is an interface that showed the protocol confirmation between SGW and SGW.

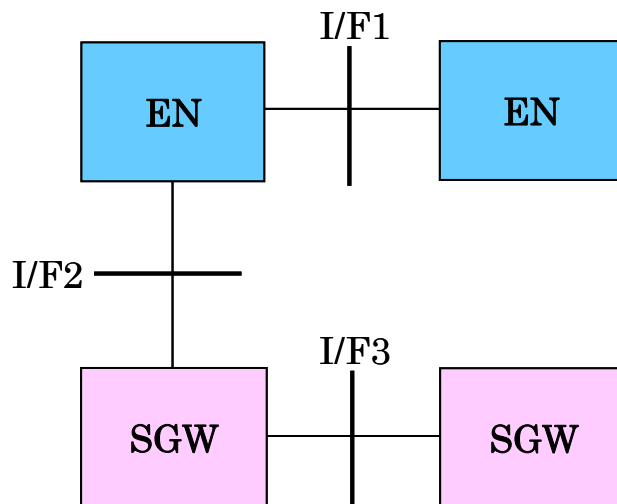


Figure 2-1 Reference Network Architecture

This document only covers IKEv2 specifications. Testing of generic IPv6 functions is beyond the scope of this test; however, some of the generic IPv6 functions are necessary to IKEv2 functions and are thus supported in this test.

2.2. Related standards

This document covers the functions specified in the following IETF RFCs.

- (1) RFC4301 “Security Architecture for the Internet Protocol”
(<http://www.ietf.org/rfc/rfc4301.txt>)
- (2) RFC4302 “IP Authentication Header”
(<http://www.ietf.org/rfc/rfc4302.txt>)
- (3) RFC4303 “IP Encapsulating Security Payload (ESP)”
(<http://www.ietf.org/rfc/rfc4303.txt>)
- (4) RFC4305 “Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)”

- <http://www.ietf.org/rfc/rfc4305.txt>
- (5) RFC4306 “Internet Key Exchange (IKEv2) Protocol”
<http://www.ietf.org/rfc/rfc4306.txt>
 - (6) RFC4307 “Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)”
<http://www.ietf.org/rfc/rfc4307.txt>
 - (7) RFC2404 “The Use of HMAC-SHA-1-96 within ESP and AH”
<http://www.ietf.org/rfc/rfc2404.txt>
 - (8) RFC2410 “The NULL Encryption Algorithm and Its Use With IPsec”
<http://www.ietf.org/rfc/rfc2410.txt>
 - (9) RFC2451 “The ESP CBC-Mode Cipher Algorithms”
<http://www.ietf.org/rfc/rfc2451.txt>
 - (10) RFC3526 “More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)”
<http://www.ietf.org/rfc/rfc3526.txt>
 - (11) RFC3566 “The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec”
<http://www.ietf.org/rfc/rfc3566.txt>
 - (12) RFC3602 “The AES-CBC Cipher Algorithm and Its Use with IPsec”
<http://www.ietf.org/rfc/rfc3602.txt>
 - (13) RFC3686 “Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)”
<http://www.ietf.org/rfc/rfc3686.txt>
 - (14) RFC4434 “The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)”
<http://www.ietf.org/rfc/rfc4301.txt>
 - (15) RFC4718 “IKEv2 Clarifications and Implementation Guidelines”
<http://www.ietf.org/rfc/rfc4718.txt>
 - (16) RFC4868 “Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec”
<http://www.ietf.org/rfc/rfc4868.txt>
 - (17) RFC5114 “Additional Diffie-Hellman Groups for Use with IETF Standards”
<http://www.ietf.org/rfc/rfc5114.txt>

3. Classification of IKEv2 functions

This section describes ways to classify the IKEv2 functions needed for interoperability and provided as test functions in the IKEv2 Conformance Test.

3.1. Viewpoints of the classification

The classification of IKEv2 functions is from the following viewpoints.

(A) IETF specification

(B) Test Requirements

(A) IETF specification

IETF specification refers to the classification of each of the IKEv2 functions from the viewpoint of importance for implementation as indicated by usage of the keywords below in the RFCs.

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” are defined in RFC 2119.

The keyword “recommend” is used in RFC4718 because the category of this document is “Informational”.

(B) Test Requirements

Test Requirement is the classification from the viewpoint of the importance for testing and interoperability. Test Requirement has three classification Rank; that is, “BASIC”, “ADVANCED”, “Not support”.

Testing of the functions classified as “BASIC” are included in the minimum test package, for the testing functions which are essential to conformance and interoperability.

Testing of the functions classified as “ADVANCED” are optional; this depends on the application to be used. The testing of “ADVANCED” (Optional Test) items is selectively incorporated in the test package according to the functions to be supported by the EN / SGW.

Testing of the functions classified as “Not supported” are functions that need not to support.

Table 3-1 shows the definition of Test Requirements

Table 3-1 Definitions of Test Requirements

| | Definitions of Test Requirement |
|--|--|
| <p>BASIC (Required Test)</p> | <p>These functions are essential to conformance and interoperability and should basically be implemented.</p> <p>Testing of the functions classified as “BASIC” is included in the minimum test package, for the testing of functions that are essential to conformance and interoperability.</p> |
| <p>ADVANCED (Optional Test)</p> | <p>Implementation of these functions is optional.</p> <p>Testing of the functions classified as “ADVANCED” may not be needed; this depends on the application to be used.</p> <p>The testing of “ADVANCED” (Optional Test) items is selectively incorporated in the test package according to the functions to be supported by the EN / SGW.</p> |
| <p>Not support</p> | <p>Testing of the functions classified as “Not supported” are functions that need not to support.</p> |

3.2. The method to classify the IKEv2 functions


Table 3-2 shows the relationships between IETF Specification and Test Requirement in this document.


IKEv2 functions with descriptions “MUST”, “SHOULD”, “MUST NOT” and “SHOULD NOT” in the IETF RFC are basically classified as BASIC, however some of these functions are described as ADVANCED or Not support, if necessary.

In the same way, IKEv2 functions with descriptions “MAY” and “No description” are basically classified as Not support, however some of these functions are described as BASIC or ADVANCED, if necessary.

Table 3-2 Relationship of classifications between IETF Specification and Test Requirement

| (A) IETF | (B) Test Requirement |
|----------------------|-----------------------------|
| MUST MUST NOT | BASIC (Required Test) |
| SHOULD SHOULD NOT | ADVANCED (Optional Test) |
| MAY | Not Supported |
| No descriptions | Not Supported |

 supported

 not supported

As reference, the classification of functions as BASIC, ADVANCED and Not Supported is described for each node about a typical IKEv2 function at Table 3-3 to Table 3-4. The classification of Notify Status Types is described at Table 3-5.

Table 3-3 IKEv2 functions and its classifications for EN

| Function | | EN | | |
|-------------------|------------------------------|---|--|------------------------|
| | | BASIC | ADVANCED | Not Supported |
| Initial Exchanges | Initiator or Responder | Initiator, Responder | - | - |
| | Sending proposal | simple transform in single proposal(patternA) | complex transform in single proposal(patternB) multiple proposals(patternC) | - |
| | Receiving proposal | simple transform in single proposal(patternA) complex transform in single proposal(patternB) multiple proposals(patternC) | - | - |
| | Retransmission | Supported | - | - |
| | ID Type receiving | IPV6_ADDR | FQDN, RFC822_ADDR | IPV4_ADDR, KEY_ID, |
| | ID Type sending | IPV6_ADDR | FQDN, RFC822_ADDR | IPV4_ADDR, KEY_ID, |
| | Auth method | Pre-shared Key | RSA Digital Signature | DSS Digital Signature |
| | Certificate Encoding | - | X.509 Certificate - Signature | - |
| | Traffic Selector Type | TS_IPV6_ADDR_RANG E | - | TS_IPV4_ADDR_RANG E |
| | Configuration Type | - | CFG_REQUEST, CFG_REPLY | CFG_SET, CFG_ACK |
| | Configuration Attribute Type | - | INTERNAL_IP6_ADDR ESS | - |
| | EAP Authentication | - | - | Not support |
| | NAT Traversal | - | - | Not support |
| | Cookies | - | sending, receiving | - |
| Vendor ID | - | - | Not support | |

| Function | | EN | | |
|----------|--------------------------------|--------------------------------|--|-------------------|
| | | BASIC | ADVANCED | Not Supported |
| IKE_SA | Transform Type(IKE) | ENCR, PRF, INTEG, D-H | - | - |
| | Encryption Algorithm (ENCR) | ENCR_3DES | ENCR_AES_CBC | ENCR_DES, |
| | Pseudo-random Function(PRF) | PRF_HMAC_SHA1 | PRF_AES128_XCBC, PRF_HMAC_SHA2_256 | PRF_HMAC_MD5, |
| | Integrity Algorithm (INTEG) | AUTH_HMAC_SHA1_96 | AUTH_AES_XCBC_96, AUTH_HMAC_SHA2_25 6_128 | AUTH_HMAC_MD5_96, |
| | Diffie-Hellman Group (D-H) | Group2 (1024 MODP) | Group14 (2048 MODP) Group24 (2048-bit MODP Group with 256-bit Prime Order Subgroup) | - |
| CHILD_SA | IPsec mode | Transport | Tunnel | - |
| | Security Protocol | ESP | - | AH |
| | Transform Type(ESP) | ENCR, INTEG, ESN | - | - |
| | Encryption Algorithm (ENCR) | ENCR_3DES | ENCR_NULL, ENCR_AES_CBC, ENCR_AES_CTR | ENCR_DES, |
| | Integrity Algorithm (INTEG) | AUTH_HMAC_SHA1_96 | AUTH_AES_XCBC_96, NONE, AUTH_HMAC_SHA2_25 6_128 | AUTH_HMAC_MD5_96, |
| | Extended Sequence Numbers(ESN) | No Extended Sequence Numbers | Extended Sequence Number | - |
| | IPcomp | - | - | Not support |

| Function | | EN | | |
|--------------------------|------------------------------|---|--|------------------------------|
| | | BASIC | ADVANCED | Not Supported |
| CREATE_CHILD_SA Exchange | Initiator or Responder | Initiator, Responder | - | - |
| | Sending proposal | simple transform in single proposal(patternA) | complex transform in single proposal(patternB) multiple proposals(patternC) | - |
| | Receiving proposal | simple transform in single proposal(patternA) complex transform in single proposal(patternB) multiple proposals(patternC) | - | - |
| | Retransmission | Support | - | - |
| | Rekeying | IKE_SA rekeying, CHILD_SA rekeying | - | restarting the entire IKE_SA |
| | additional CHILD_SA | - | Support | - |
| | perfect forward secrecy(PFS) | - | Support | - |
| | Traffic Selector Type | IPV6_ADDR_RANGE | - | IPV4_ADDR_RANGE |
| | Vendor ID | - | - | Not support |
| INFORMATIONAL Exchange | Initiator or Responder | Initiator, Responder | - | - |
| | Retransmission | Support | - | - |
| | Liveness Check | Sending, Responding | - | - |
| | Delete SA | IKE_SA delete, CHILD_SA delete | - | - |
| | Multiple SPIs deletion | receiving | - | sending |
| | Request peer's version | - | - | requesting, answering |
| | Vendor ID | - | - | Not support |

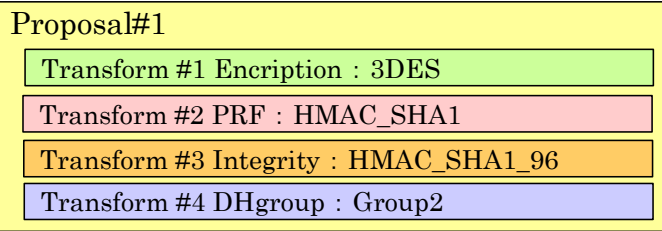
Table 3-4 IKEv2 functions and its classifications for SGW

| Function | | SGW | | |
|-------------------|------------------------------|---|--|------------------------|
| | | BASIC | ADVANCED | Not Supported |
| Initial Exchanges | Initiator or Responder | Initiator, Responder | - | - |
| | Sending proposal | simple transform in single proposal(patternA) | complex transform in single proposal(patternB) multiple proposals(patternC) | - |
| | Receiving proposal | simple transform in single proposal(patternA) complex transform in single proposal(patternB) multiple proposals(patternC) | - | - |
| | Retransmission | Supported | - | - |
| | ID Type receiving | IPV6_ADDR | FQDN, RFC822_ADDR | IPV4_ADDR, KEY_ID, |
| | ID Type sending | IPV6_ADDR | FQDN, RFC822_ADDR, | IPV4_ADDR, KEY_ID, |
| | Auth method | Pre-shared Key | RSA Digital Signature | DSS Digital Signature |
| | Certificate Encoding | - | X.509 Certificate - Signature | - |
| | Traffic Selector Type | TS_IPV6_ADDR_RANG E | - | TS_IPV4_ADDR_RANG E |
| | Configuration Type | - | CFG_REQUEST, CFG_REPLY | CFG_SET, CFG_ACK |
| | Configuration Attribute Type | - | INTERNAL_IP6_ADDR ESS | - |
| | EAP Authentication | - | - | Not support |
| | NAT Traversal | - | - | Not support |
| | Cookies | - | sending, receiving | - |
| Vendor ID | - | - | Not support | |

| Function | | SGW | | |
|----------|--------------------------------|--------------------------------|--|-------------------|
| | | BASIC | ADVANCED | Not Supported |
| IKE_SA | Transform Type(IKE) | ENCR, PRF, INTEG, D-H | - | - |
| | Encryption Algorithm (ENCR) | ENCR_3DES | ENCR_AES_CBC | ENCR_DES, |
| | Pseudo-random Function(PRF) | PRF_HMAC_SHA1 | PRF_AES128_XCBC, PRF_HMAC_SHA2_256 | PRF_HMAC_MD5, |
| | Integrity Algorithm (INTEG) | AUTH_HMAC_SHA1_96 | AUTH_AES_XCBC_96, AUTH_HMAC_SHA2_25 6_128 | AUTH_HMAC_MD5_96, |
| | Diffie-Hellman Group (D-H) | Group2 (1024 MODP) | Group14 (2048 MODP) Group24 (2048-bit MODP Group with 256-bit Prime Order Subgroup) | - |
| CHILD_SA | IPsec mode | Tunnel | - | - |
| | Security Protocol | ESP | - | AH |
| | Transform Type(ESP) | ENCR, INTEG, ESN | - | - |
| | Encryption Algorithm (ENCR) | ENCR_3DES | ENCR_NULL, ENCR_AES_CBC, ENCR_AES_CTR | ENCR_DES, |
| | Integrity Algorithm (INTEG) | AUTH_HMAC_SHA1_96 | AUTH_AES_XCBC_96, NONE AUTH_HMAC_SHA2_25 6_128 | AUTH_HMAC_MD5_96, |
| | Extended Sequence Numbers(ESN) | No Extended Sequence Numbers | Extended Sequence Number | - |
| | IPcomp | - | - | Not support |

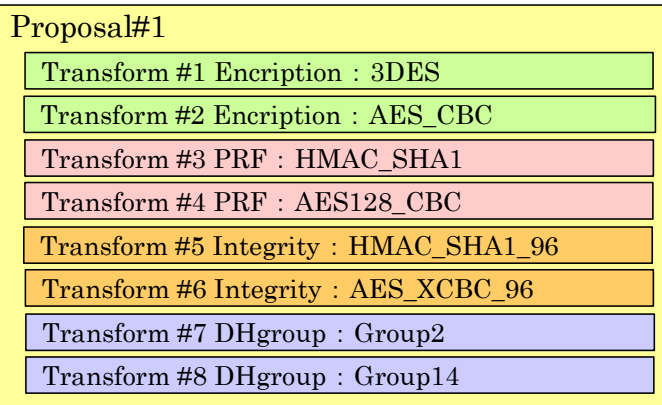
| Function | | SGW | | |
|--------------------------|------------------------------|---|--|------------------------------|
| | | BASIC | ADVANCED | Not Supported |
| CREATE_CHILD_SA Exchange | Initiator or Responder | Initiator, Responder | - | - |
| | Sending proposal | simple transform in single proposal(patternA) | complex transform in single proposal(patternB) multiple proposals(patternC) | - |
| | Receiving proposal | simple transform in single proposal(patternA) complex transform in single proposal(patternB) multiple proposals(patternC) | - | - |
| | Retransmission | Support | - | - |
| | Rekeying | IKE_SA rekeying, CHILD_SA rekeying | - | restarting the entire IKE_SA |
| | additional CHILD_SA | - | Support | - |
| | perfect forward secrecy(PFS) | - | Support | - |
| | Traffic Selector Type | IPV6_ADDR_RANGE | - | IPV4_ADDR_RANGE |
| | Vendor ID | - | - | Not support |
| INFORMATIONAL Exchange | Initiator or Responder | Initiator, Responder | - | - |
| | Retransmission | Support | - | - |
| | Liveness Check | Sending, Responding | - | - |
| | Delete SA | IKE_SA delete, CHILD_SA delete | - | - |
| | Multiple SPIs deletion | receiving | - | sending |
| | Request peer's version | - | - | requesting, answering |
| | Vendor ID | - | - | Not support |

SA Payload



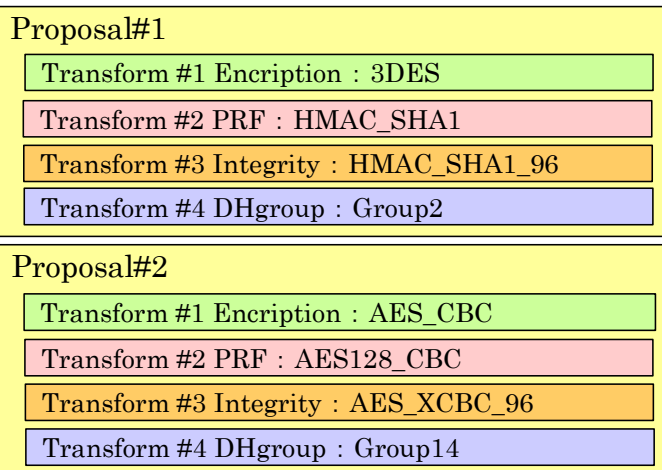
patternA: simple transform in single proposal

SA Payload



patternB: complex transform in single proposal

SA Payload



patternC: multiple proposals

Table 3-5 IKEv2 Notify message for Status types and its classifications

| Status Types | Classification | recital |
|-------------------------------|----------------|---|
| INITIAL_CONTACT | ADVANCED | sending and receiving |
| SET_WINDOW_SIZE | Not support | |
| ADDITIONAL_TS_POSSIBLE | Not support | |
| IPCOMP_SUPPORTED | Not support | |
| NAT_DETECTION_SOURCE_IP | Not support | IPv6 network |
| NAT_DETECTION_DESTINATION_IP | Not support | IPv6 network |
| COOKIE | ADVANCED | |
| USE_TRANSPORT_MODE(only EN) | BASIC | use only End-Node |
| HTTP_CERT_LOOKUP_SUPPORTED | Not support | |
| REKEY_SA | BASIC | Rekey function BASIC |
| ESP_TFC_PADDING_NOT_SUPPORTED | BASIC | TFC Padding function Not support in IPsec Guidelines(IPsec v2) |
| NON_FIRST_FRAGMENTS_ALSO | Not support | |

4. IKEv2 Sequences and Payloads

This section describes the IKEv2 sequences and payloads used in the IKEv2 Guidelines for Implementation. Sequences of test packet are sent to the target and expects to receive corresponding acknowledgement packets from the target. Details of the test sequences and payloads utilized in each test are given in the Test Specification documents. A gray color payload means the encrypted in the figure of payload and a double allow means the IPsec communication in the figure of sequence.

4.1. IKEv2 BASIC sequences and payloads

This section consist of two items, initial exchange and rekey.

The Initial exchange sequences and payloads are shown from Figure 4-1 to Figure 4-8. The rekey exchange sequences and payloads are shown from Figure 4-9 to Figure 4-20.

4.1.1. Initial exchange

4.1.1.1. EN to EN

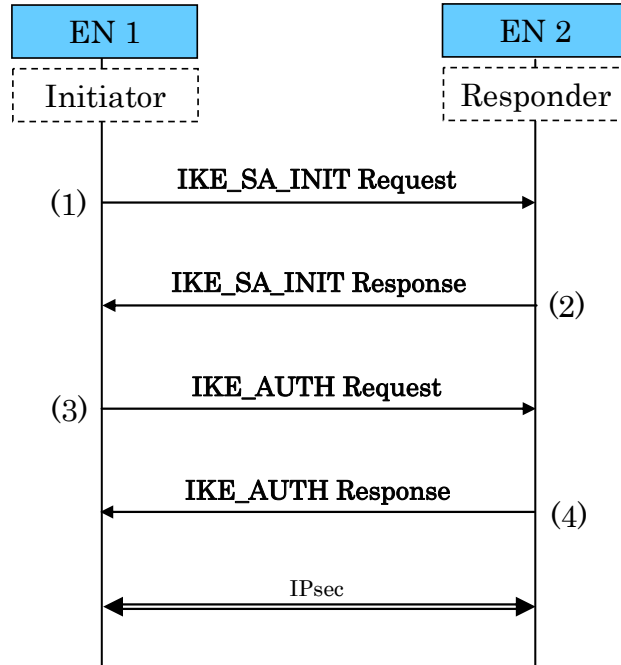
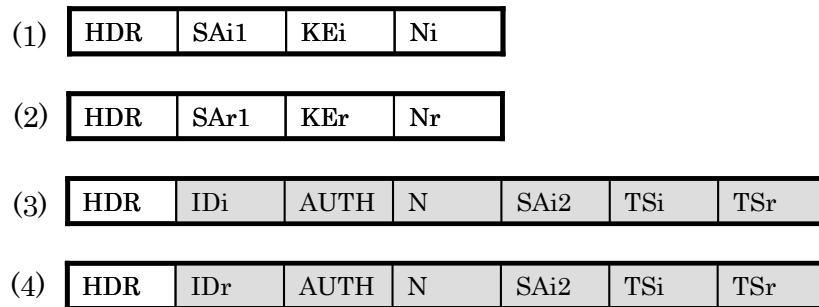


Figure 4-1 EN to EN



N : USE_TRANSPORT_MODE

Figure 4-2 EN to EN Payloads

4.1.1.2. EN to SGW

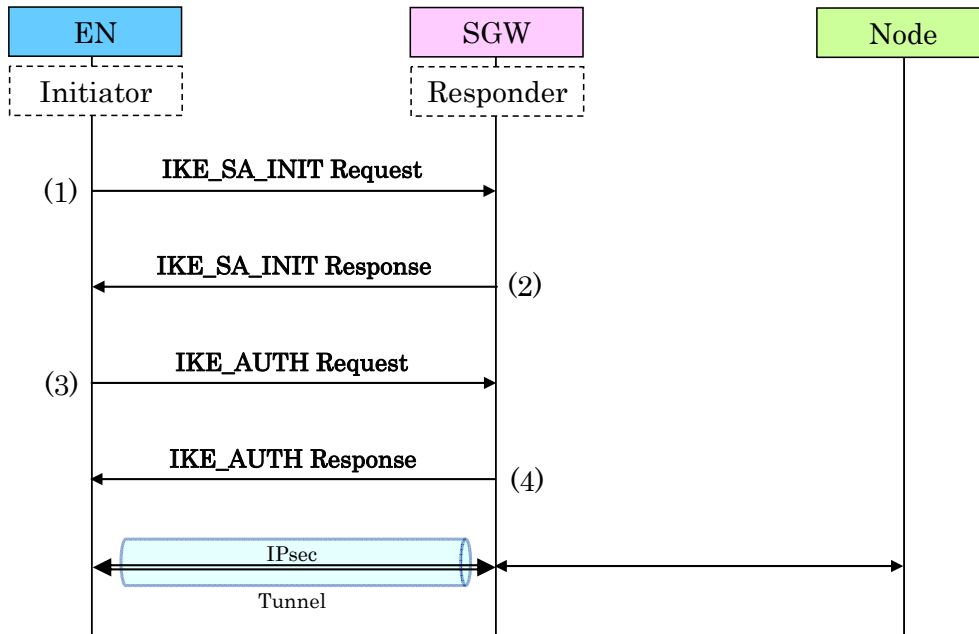


Figure 4-3 EN to SGW

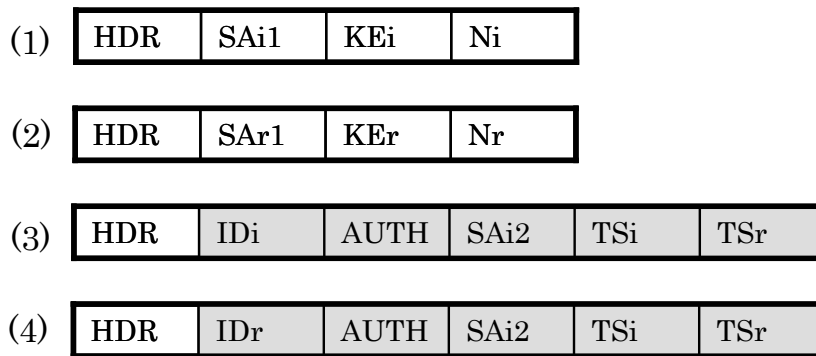


Figure 4-4 EN to SGW Payloads

4.1.1.3. SGW to EN

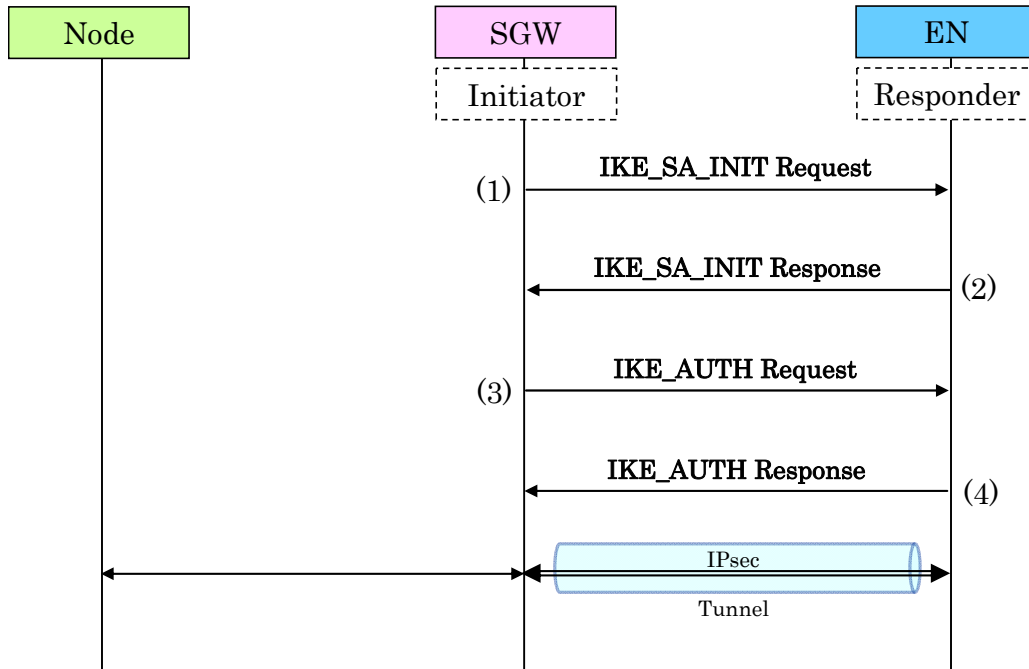


Figure 4-5 SGW to EN

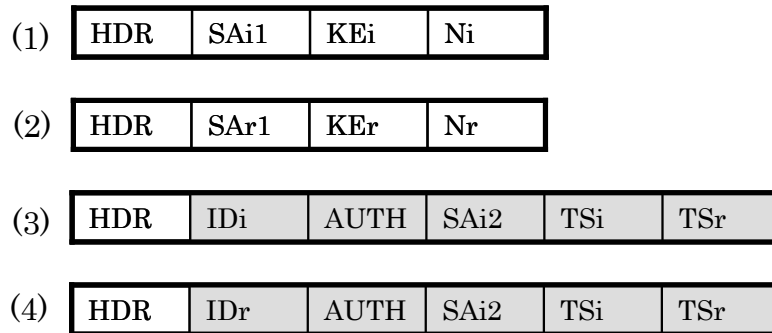


Figure 4-6 SGW to EN Payloads

4.1.1.4. SGW to SGW

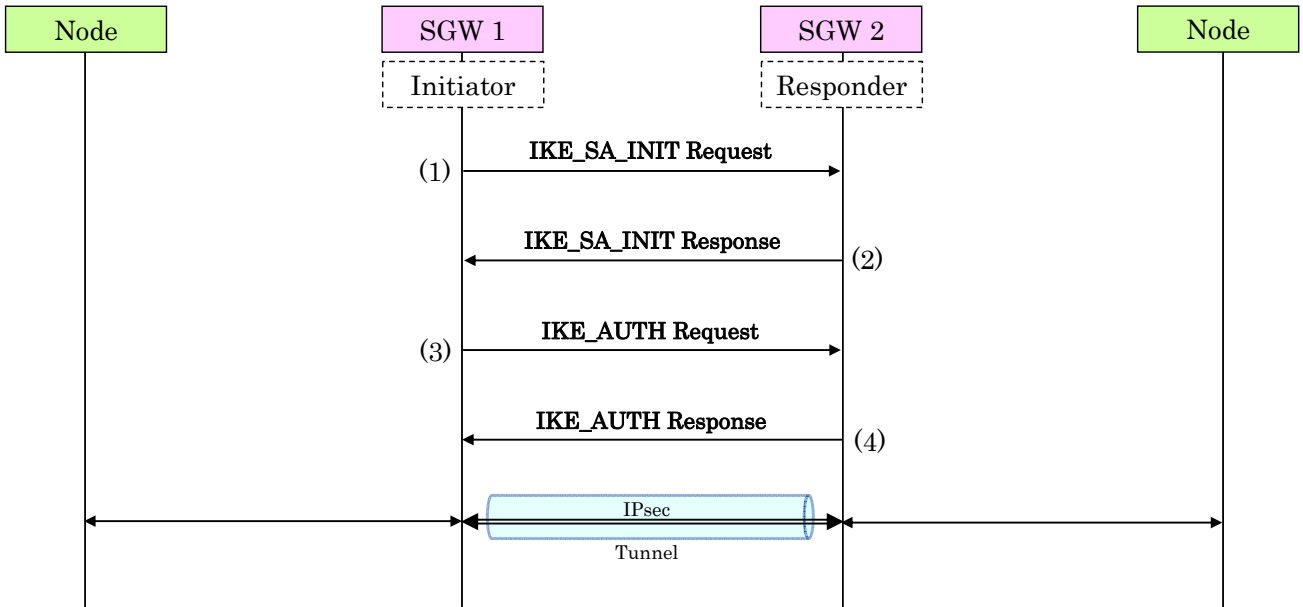


Figure 4-7 SGW to SGW

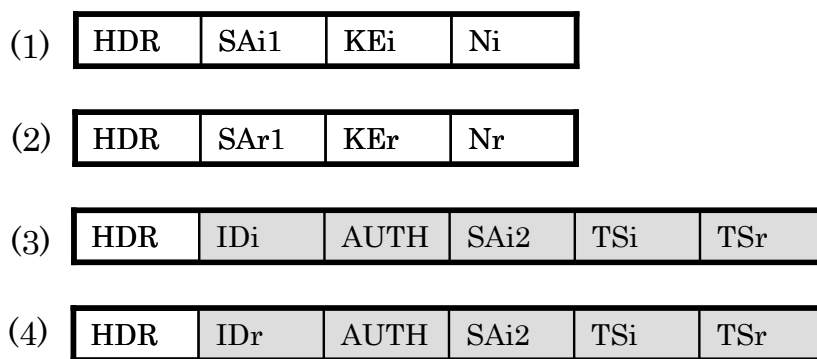


Figure 4-8 SGW to SGW Payloads

4.1.2. Rekey

4.1.2.1. Rekey by EN to EN

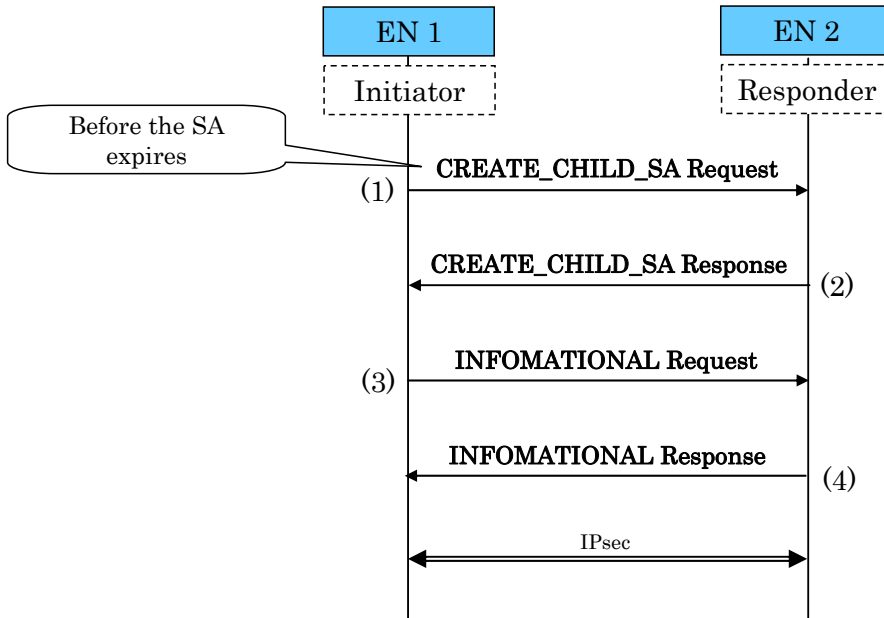


Figure 4-9 Rekey by EN to EN

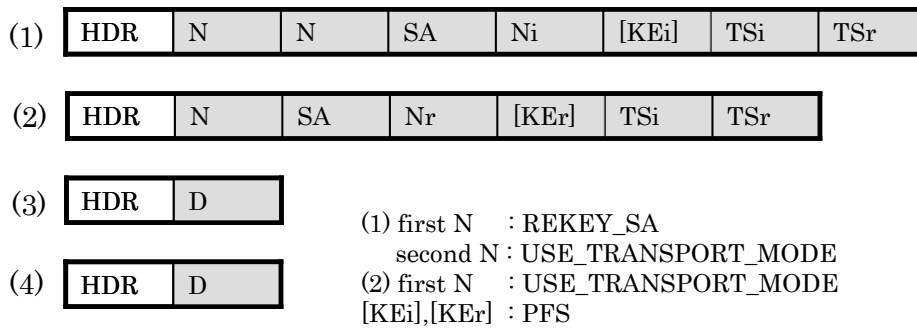


Figure 4-10 CHILD_SA Rekey Payloads by EN to EN

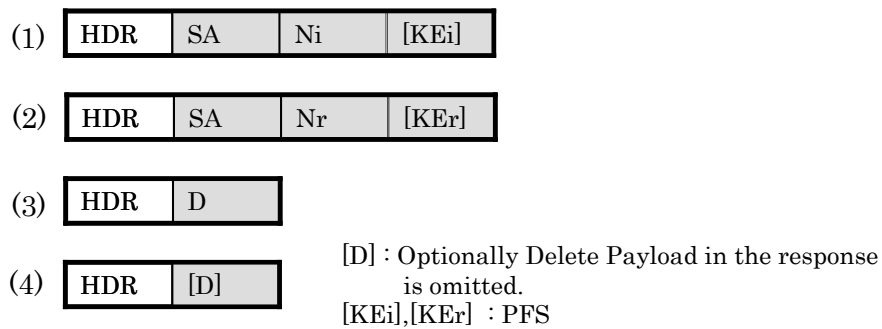


Figure 4-11 IKE_SA Rekey Payloads by EN to EN

4.1.2.2. Rekey by EN to SGW

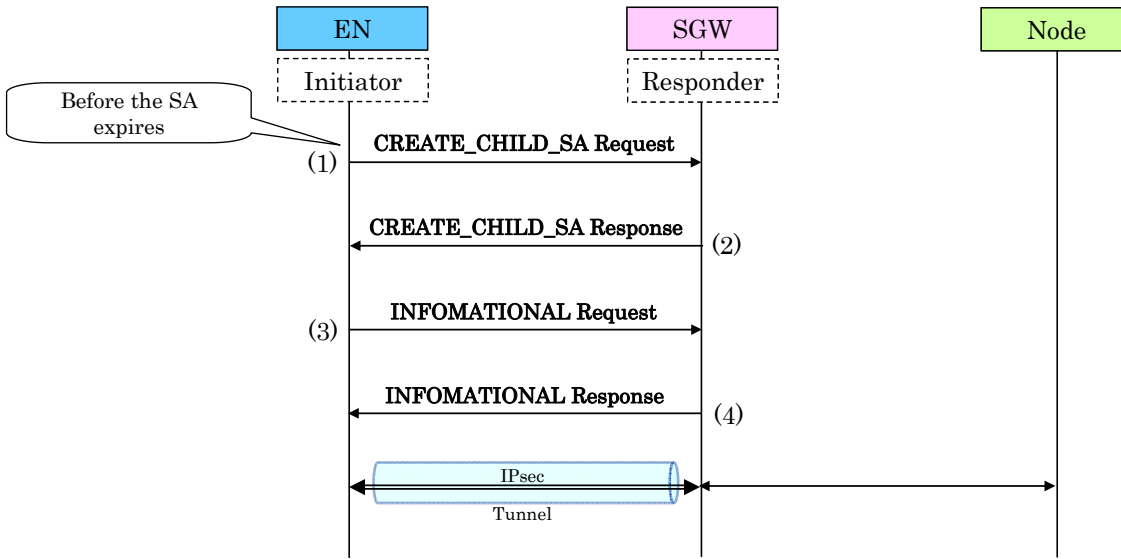


Figure 4-12 Rekey by EN to SGW

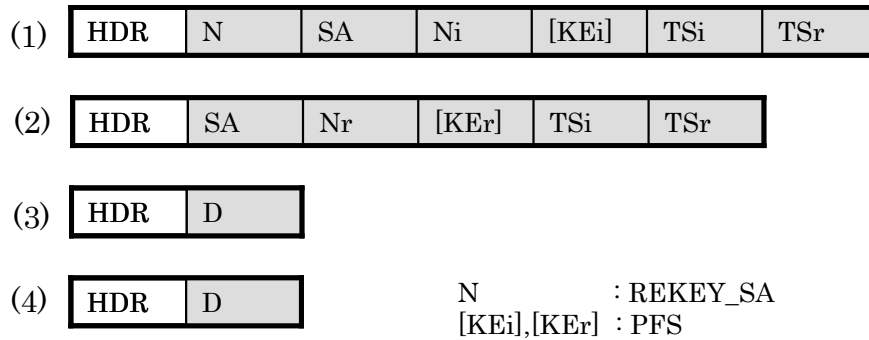


Figure 4-13 CHILD_SA Rekey Payloads by EN to SGW

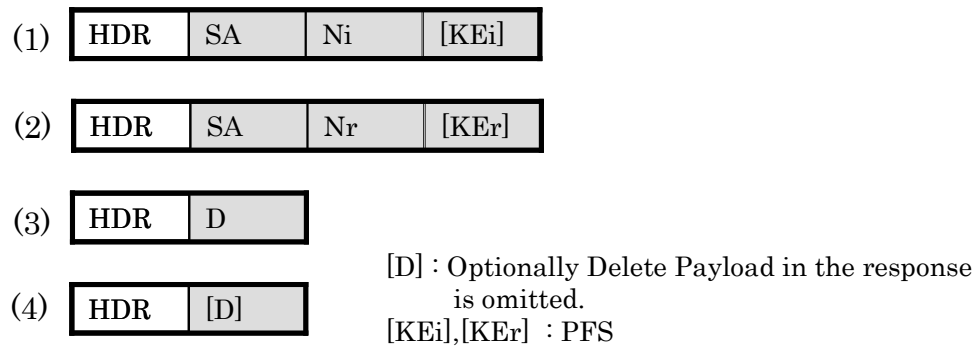


Figure 4-14 IKE_SA Rekey Payloads by EN to SGW

4.1.2.3. Rekey by SGW to EN

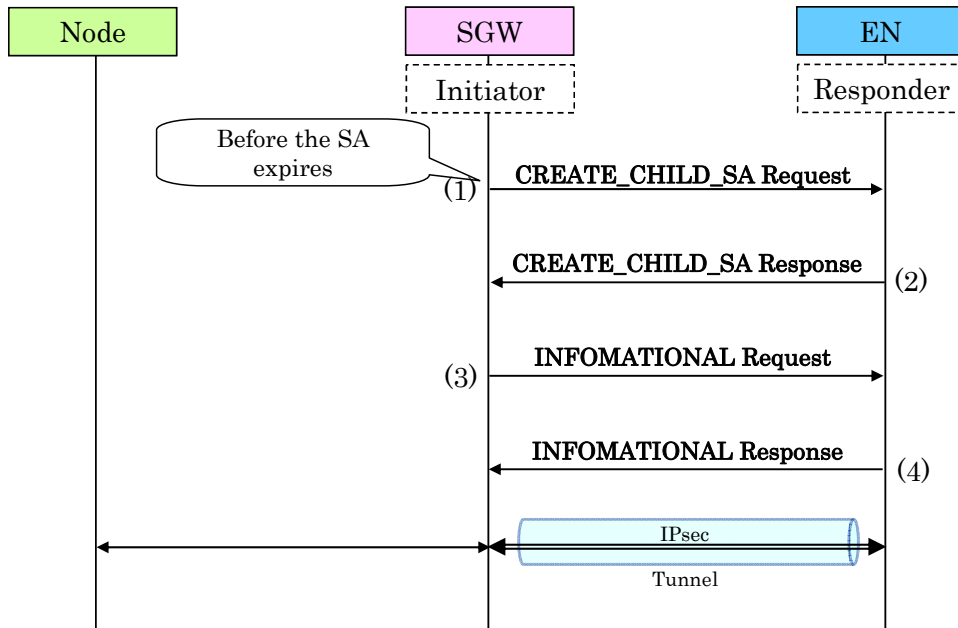


Figure 4-15 Rekey by SGW to EN

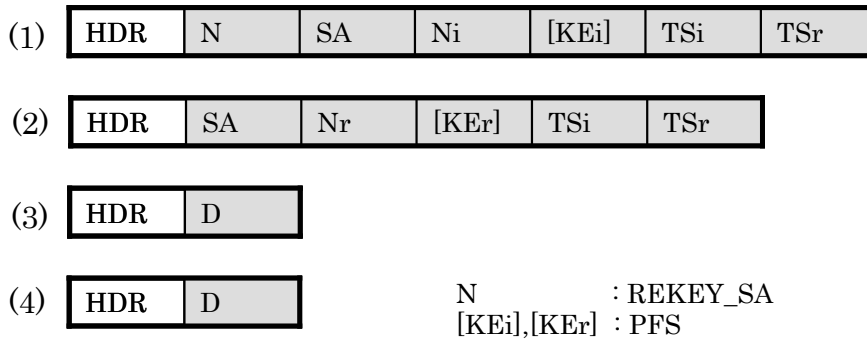


Figure 4-16 CHILD_SA Rekey Payloads by SGW to EN

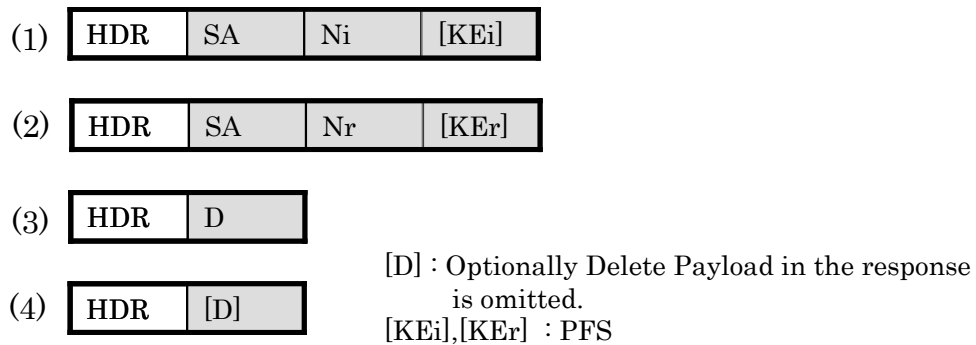


Figure 4-17 IKE_SA Rekey Payloads by SGW to EN

4.1.2.4. Rekey by SGW to SGW

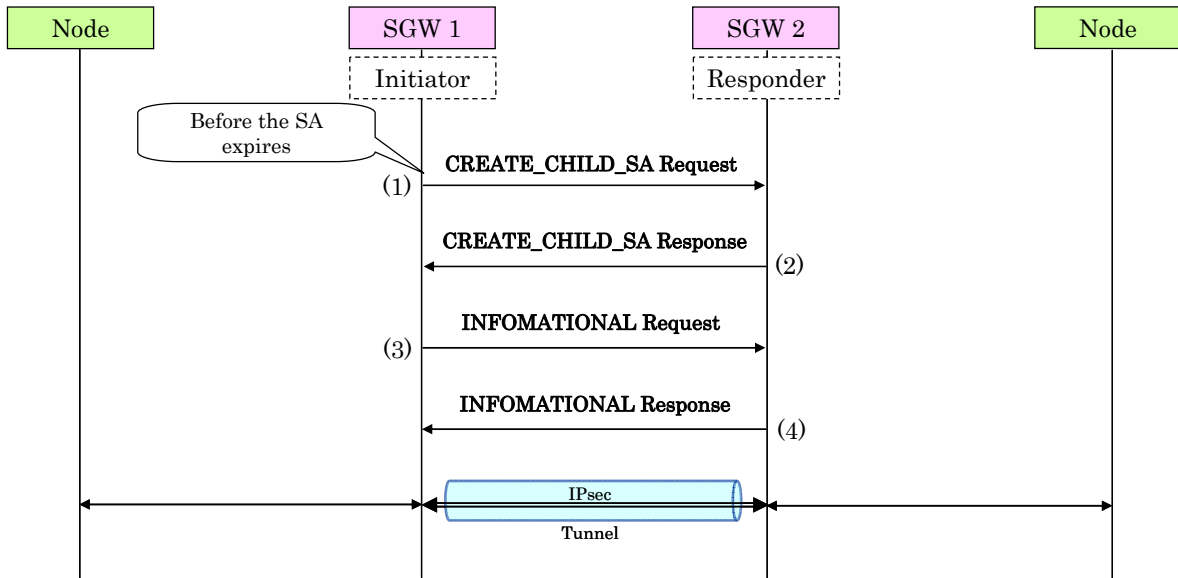


Figure 4-18 Rekey by SGW to SGW

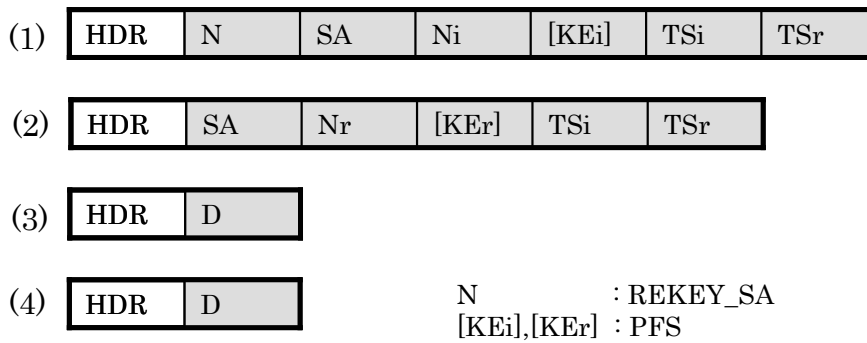


Figure 4-19 CHILD_SA Rekey Payloads by SGW to SGW

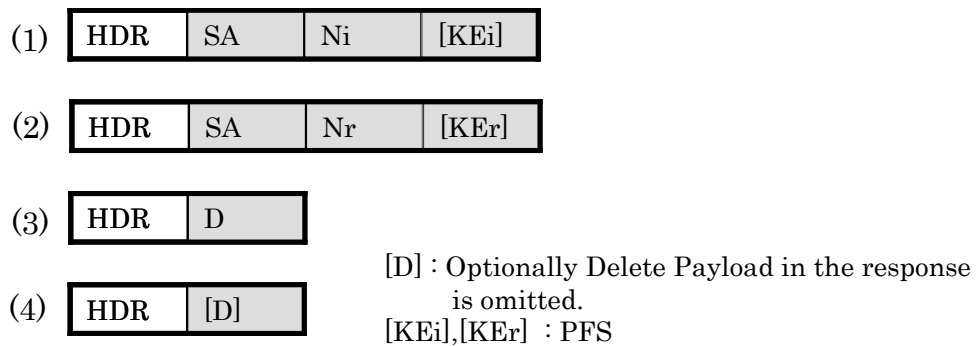


Figure 4-20 IKE_SA Rekey Payloads by SGW to SGW

4.2. IKEv2 ADVANCED sequences and payloads

This section consist of two cases, mutual authentication using public key signature and Extensible Authentication Protocol (EAP) method. EAP method utilize in this document is EAP with MD5 (EAP-MD5).

The authentication using public key signature sequences and payloads are shown from Figure 4-21 to Figure 4-28.

EAP-MD5 sequences and payloads are shown in Appendix_A.

EAP-TLS sequences and payloads are shown in Appendix_B.

4.2.1. Mutual authentication using public signature

4.2.1.1. Mutual authentication using public key signature in the case of EN-EN

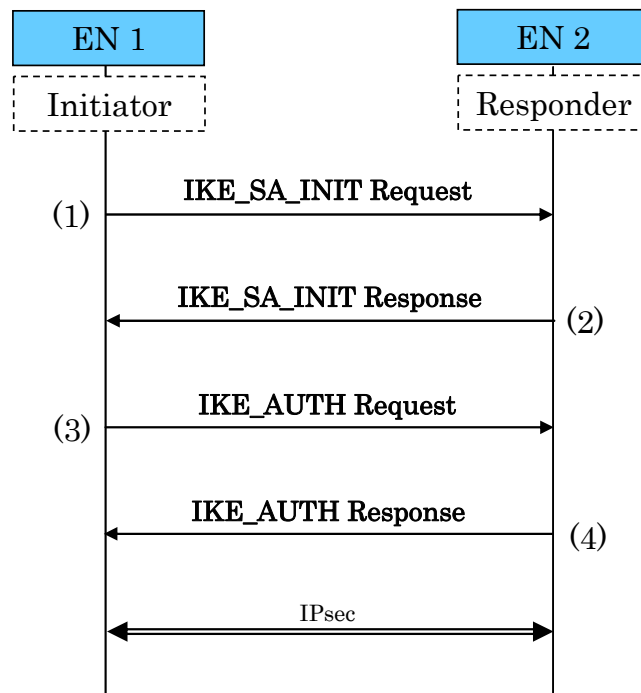
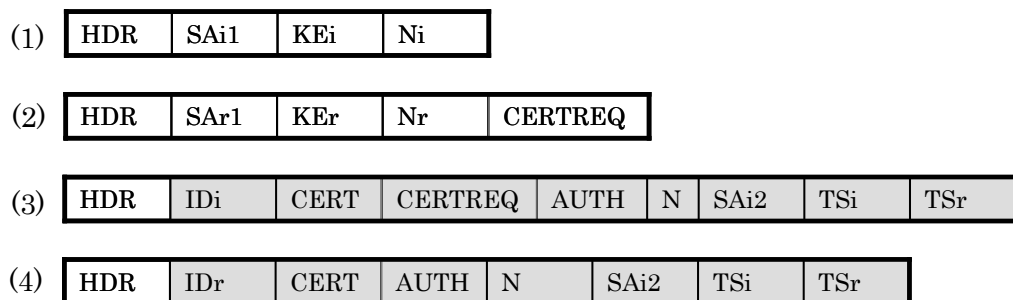


Figure 4-21 Mutual authentication using public key signature by EN to EN



N : USE_TRANSPORT_MODE

Figure 4-22 Payloads mutual authentication using public key signature by EN to EN

4.2.1.2. Mutual authentication using public key signature in the case of EN-SGW

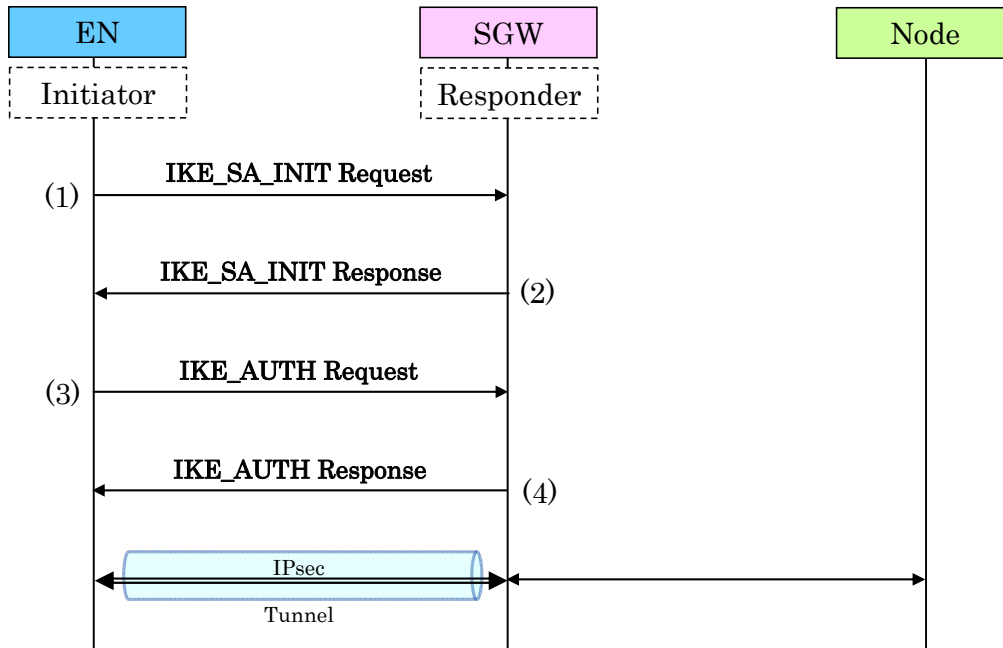


Figure 4-23 Mutual authentication using public key signature by EN to SGW

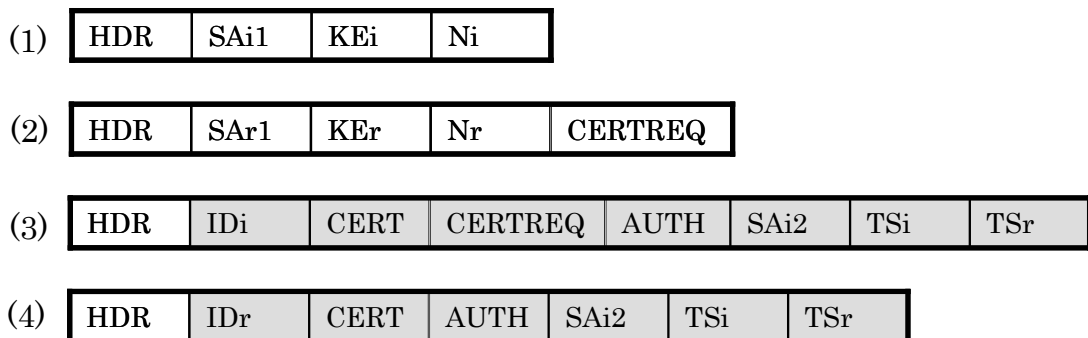


Figure 4-24 Payloads mutual authentication using public key signature by EN to SGW

4.2.1.3. Mutual authentication using public key signature in the case of SGW-EN

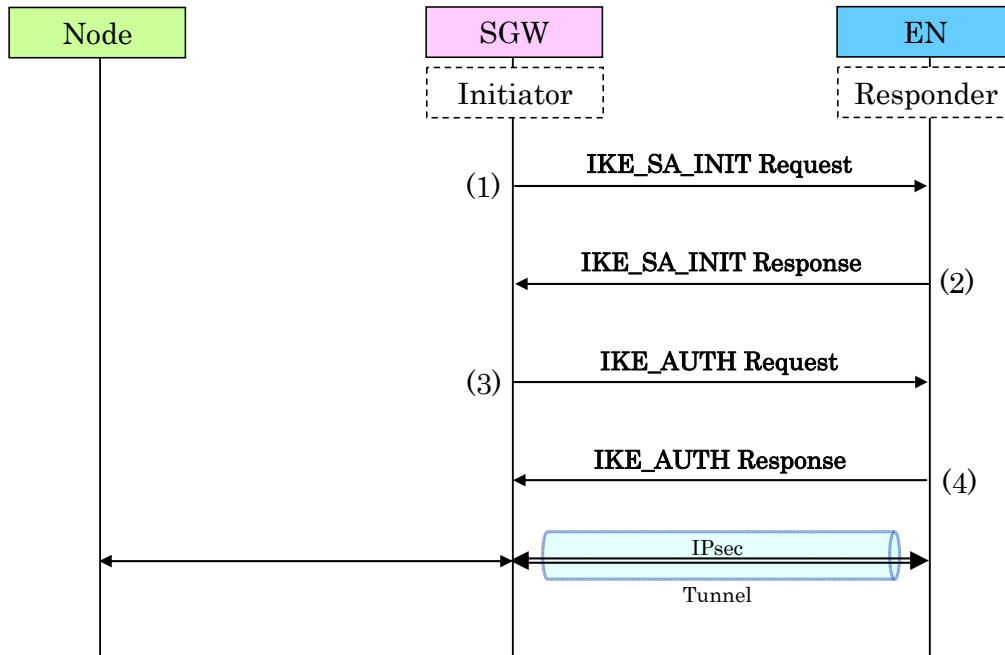


Figure 4-25 Mutual authentication using public key signature by SGW to EN

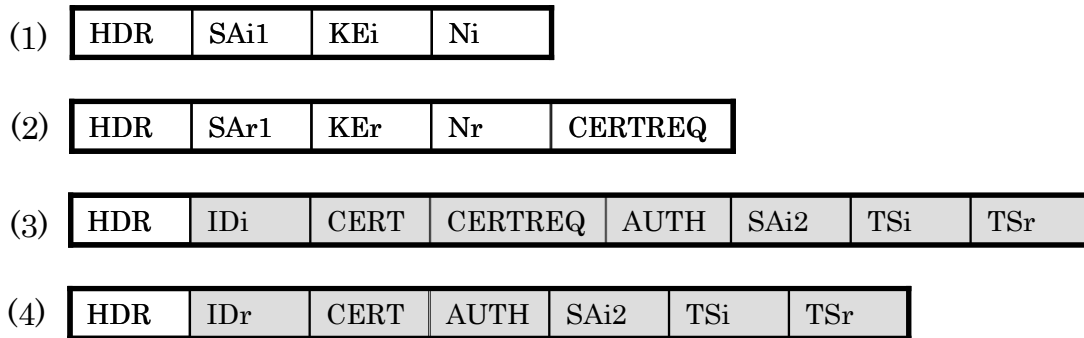


Figure 4-26 Payloads mutual authentication using public key signature by SGW to EN

4.2.1.4. Mutual authentication using public key signature in the case of SGW-SGW

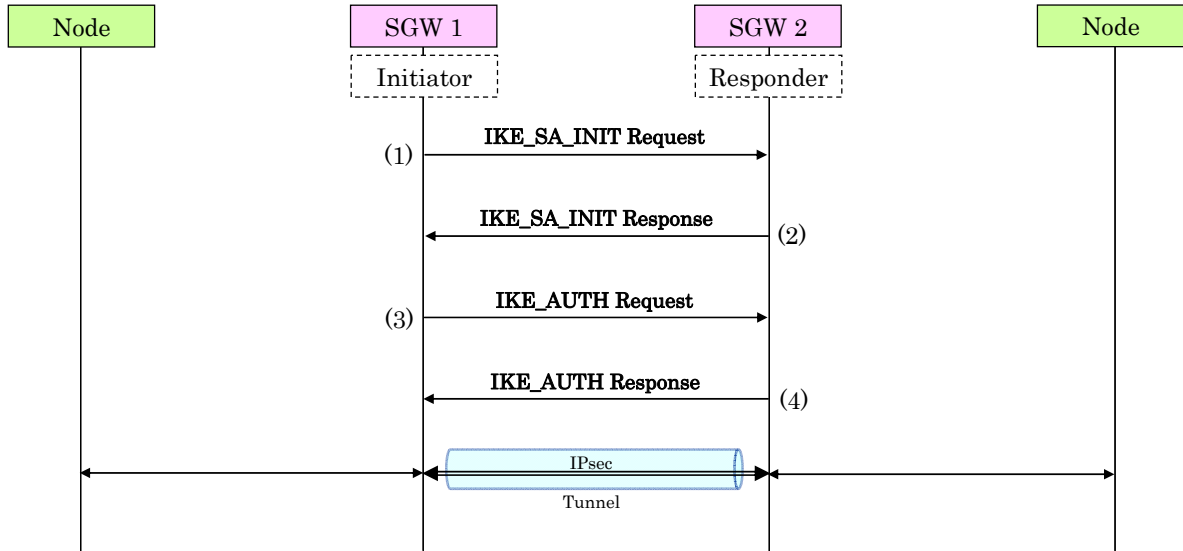


Figure 4-27 Mutual authentication using public key signature by SGW to SGW

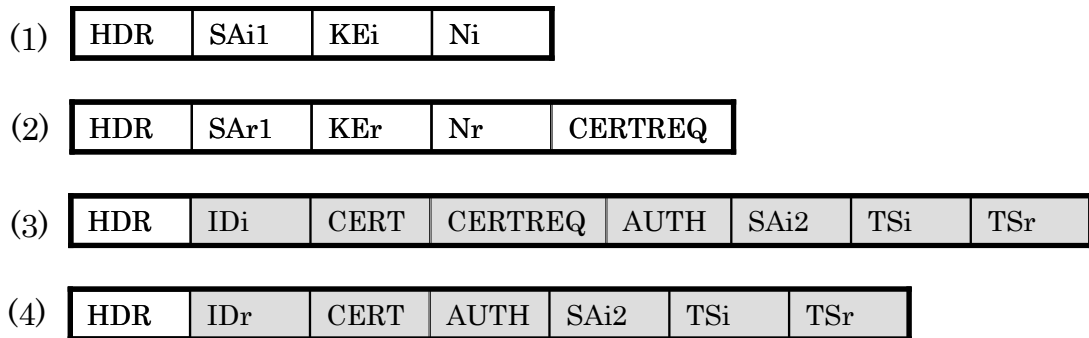


Figure 4-28 Payloads mutual authentication using public key signature by SGW to SGW

5. Priorities of IKEv2 function for testing

This chapter describes the detail IKEv2 functional classifications on the basis of the classifications given in chapter 3.

Priorities of IKEv2 function for testing in RFC4306 and in RFC4718 are shown at Table 5-2 and Table 5-3 respectively.

Notes

- “**page**” gives the corresponding page number in RFC4306 and RFC4718.
- “**line**” gives the corresponding line number in RFC4306 and RFC4718.
- “**sentence**” gives the statement in RFC4306 and RFC4718.
- “**RFC requirement**” gives the corresponding requirement like “MUST” etc. in RFC4306 and RFC4718.
- “**test requirement**” gives the corresponding requirement of the conformance test in RFC4306 and RFC4718.
- “**target**” gives the corresponding target of the test as follows.
 - If the test requirement is “BASIC” or “ADVANCED”, the value of this column indicates one or more supporting functions including EN(initiator), EN(responder) , SGW(initiator) and SGW(responder).
 - If the test requirement is “Not support”,the value of this column is blank.
- “**test number or reason**” gives the corresponding test numbers or reasons as follows.
 - If the test requirement is “BASIC” or “ADVANCED”, the value of this column indicates the test number of the conformance test specification.
 - If the test requirement is “Not support”, the value of this column indicates the reason why the part of this description is “Not support”.

Table 5-2 IKEv2 functions and its classifications for RFC4306

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--------|-----------------------|
| 1 | 14 | Status of This Memo | | | | |
| 1 | 16 | This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited. | | Not support | | Explanation |
| 1 | 26 | Abstract | | | | |
| 1 | 28 | This document describes version 2 of the Internet Key Exchange (IKE) protocol. IKE is a component of IPsec used for performing mutual authentication and establishing and maintaining security associations(SAs). | | Not support | | Explanation |
| 1 | 33 | This version of the IKE specification combines the contents of what were previously separate documents, including Internet Security Association and Key Management Protocol (ISAKMP, RFC 2408), IKE (RFC2409), the Internet Domain of Interpretation (DOI, RFC 2407), Network Address Translation (NAT) Traversal, Legacy authentication, and remote address acquisition. | | Not support | | Explanation |
| 1 | 40 | Version 2 of IKE does not interoperate with version 1, but it has enough of the header format in common that both versions can unambiguously run over the same UDP port. | | Not support | | Explanation |
| 3 | 132 | 1. Introduction | | | | |
| 3 | 134 | IP Security (IPsec) provides confidentiality, data integrity, access control, and data source authentication to IP datagrams. These services are provided by maintaining shared state between the source and the sink of an IP datagram. This state defines, among other things, the specific services provided to the datagram, which cryptographic algorithms will be used to provide the services, and the keys used as input to the cryptographic algorithms. | | Not support | | Explanation |
| 3 | 142 | Establishing this shared state in a manual fashion does not scale well. Therefore, a protocol to establish this state dynamically is needed. This memo describes such a protocol -- the Internet Key Exchange (IKE). This is version 2 of IKE. Version 1 of IKE was defined in RFCs 2407, 2408, and 2409 [Pip98, MSST98, HC98]. This single document is intended to replace all three of those RFCs. | | Not support | | Explanation |
| 3 | 149 | Definitions of the primitive terms in this document (such as Security Association or SA) can be found in [RFC4301]. | | Not support | | Explanation |
| 3 | 152 | Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in [Bra97]. | | Not support | | Explanation |
| 3 | 156 | The term "Expert Review" is to be interpreted as defined in [RFC2434]. | | Not support | | Explanation |
| 3 | 159 | IKE performs mutual authentication between two parties and establishes an IKE security association (SA) that includes shared secret information that can be used to efficiently establish SAs for Encapsulating Security Payload (ESP) [RFC4303] and/or Authentication Header (AH) [RFC4302] and a set of cryptographic algorithms to be used by the SAs to protect the traffic that they carry. In this document, the term "suite" or "cryptographic suite" refers to a complete set of algorithms used to protect an SA. | | Not support | | Explanation |
| 4 | 174 | An initiator proposes one or more suites by listing supported algorithms that can be combined into suites in a mix-and-match fashion. IKE can also negotiate use of IP Compression (IPComp) [IPCOMP] in connection with an ESP and/or AH SA. We call the IKE SA an "IKE_SA". The SAs for ESP and/or AH that get set up through that IKE_SA we call "CHILD_SAs". | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 4 | 182 | All IKE communications consist of pairs of messages: a request and a response. The pair is called an "exchange". We call the first messages establishing an IKE_SA IKE_SA_INIT and IKE_AUTH exchanges and subsequent IKE exchanges CREATE_CHILD_SA or INFORMATIONAL exchanges. In the common case, there is a single IKE_SA_INIT exchange and a single IKE_AUTH exchange (a total of four messages) to establish the IKE_SA and the first CHILD_SA. In exceptional cases, there may be more than one of each of these exchanges. | | Not support | | Explanation |
| 4 | 189 | In all cases, all IKE_SA_INIT exchanges MUST complete before any other exchange type, then all IKE_AUTH exchanges MUST complete, and following that any number of CREATE_CHILD_SA and INFORMATIONAL exchanges may occur in any order. | MUST MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.2.1 EN.I.1.1.2.2 EN.I.1.1.2.3 EN.I.1.1.2.4 EN.I.1.2.2.1 EN.I.1.2.2.2 EN.I.1.3.2.2 EN.R.1.1.2.1 EN.R.1.1.2.2 EN.R.1.2.2.1 EN.R.1.3.2.1 SGW.I.1.1.2.1 SGW.I.1.1.2.2 SGW.I.1.1.2.3 SGW.I.1.1.2.4 SGW.I.1.2.2.1 SGW.I.1.2.2.2 SGW.I.1.3.2.2 SGW.R.1.1.2.1 SGW.R.1.1.2.2 SGW.R.1.2.2.1 SGW.R.1.3.2.1 |
| 4 | 193 | In some scenarios, only a single CHILD_SA is needed between the IPsec endpoints, and therefore there would be no additional exchanges. Subsequent exchanges MAY be used to establish additional CHILD_SAs between the same authenticated pair of endpoints and to perform housekeeping functions. | MAY | Not support | | Not need to test |
| 4 | 199 | IKE message flow always consists of a request followed by a response. It is the responsibility of the requester to ensure reliability. If the response is not received within a timeout interval, the requester needs to retransmit the request (or abandon the connection). | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.2.1 EN.I.1.1.2.2 EN.I.1.1.2.3 EN.I.1.1.2.4 EN.I.1.2.2.1 EN.I.1.2.2.2 EN.I.1.3.2.2 EN.R.1.1.2.1 EN.R.1.1.2.2 EN.R.1.2.2.1 EN.R.1.3.2.1 SGW.I.1.1.2.1 SGW.I.1.1.2.2 SGW.I.1.1.2.3 SGW.I.1.1.2.4 SGW.I.1.2.2.1 SGW.I.1.2.2.2 SGW.I.1.3.2.2 SGW.R.1.1.2.1 SGW.R.1.1.2.2 SGW.R.1.2.2.1 SGW.R.1.3.2.1 |
| 4 | 204 | The first request/response of an IKE session (IKE_SA_INIT) negotiates security parameters for the IKE_SA, sends nonces, and sends Diffie-Hellman values. | | Not support | | Explanation |
| 4 | 208 | The second request/response (IKE_AUTH) transmits identities, proves knowledge of the secrets corresponding to the two identities, and sets up an SA for the first (and often only) AH and/or ESP CHILD_SA. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|----------------------------------|--|
| 4 | 212 | The types of subsequent exchanges are CREATE_CHILD_SA (which creates a CHILD_SA) and INFORMATIONAL (which deletes an SA, reports error conditions, or does other housekeeping). Every request requires a response. An INFORMATIONAL request with no payloads (other than the empty Encrypted payload required by the syntax) is commonly used as a check for liveness. These subsequent exchanges cannot be used until the initial exchanges have completed. | | Not support | | Explanation |
| 5 | 230 | In the description that follows, we assume that no errors occur. Modifications to the flow should errors occur are described in section 2.21. | | Not support | | Explanation |
| 5 | 234 | 1.1. Usage Scenarios | | | | |
| 5 | 236 | IKE is expected to be used to negotiate ESP and/or AH SAs in a number of different scenarios, each with its own special requirements. | | Not support | | Explanation |
| 5 | 239 | 1.1.1. Security Gateway to Security Gateway Tunnel | | | | |
| 5 | 241 | <pre> +++++++ ++++++ ! !IPsec ! ! Protected !Tunnel !tunnel !Tunnel ! Protected Subnet <->!Endpoint !<----->!Endpoint !<-> Subnet ! ! ! ! +++++++ ++++++ </pre> <p>Figure 1: Security Gateway to Security Gateway Tunnel</p> | | BASIC | SGW(initiator) SGW(responder) | SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 |
| | | | | Not support | EN(initiator) EN(responder) | EN is out of scope here. |
| 5 | 250 | In this scenario, neither endpoint of the IP connection implements IPsec, but network nodes between them protect traffic for part of the way. Protection is transparent to the endpoints, and depends on ordinary routing to send packets through the tunnel endpoints for processing. Each endpoint would announce the set of addresses "behind" it, and packets would be sent in tunnel mode where the inner IP header would contain the IP addresses of the actual endpoints. | | BASIC | SGW(initiator) SGW(responder) | SGW.I.1.1.1.3 SGW.R.1.1.1.3 |
| | | | | Not support | EN(initiator) EN(responder) | EN is out of scope here. |
| 5 | 258 | 1.1.2. Endpoint-to-Endpoint Transport | | | | |
| 5 | 260 | <pre> +++++++ ++++++ ! ! IPsec transport ! ! !Protected! or tunnel mode SA !Protected! !Endpoint !<----->!Endpoint ! ! ! ! ! +++++++ ++++++ </pre> <p>Figure 2: Endpoint to Endpoint</p> | | Not support | SGW(initiator) SGW(responder) | SGW is out of scope here. |
| | | | | BASIC | EN(initiator) EN(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 |
| 5 | 269 | In this scenario, both endpoints of the IP connection implement IPsec, as required of hosts in [RFC4301]. Transport mode will commonly be used with no inner IP header. If there is an inner IP header, the inner addresses will be the same as the outer addresses. A single pair of addresses will be negotiated for packets to be protected by this SA. These endpoints MAY implement application layer access controls based on the IPsec authenticated identities of the participants. This scenario enables the end-to-end security that has been a guiding principle for the Internet since [RFC1958],[RFC2775], and a method of limiting the inherent problems with complexity in networks noted by [RFC3439]. Although this scenario may not be fully applicable to the IPv4 Internet, it has been deployed successfully in specific scenarios within intranets using IKEv1. It should be more broadly enabled during the transition to IPv6 and with the adoption of IKEv2. | MAY | Not support | SGW(initiator) SGW(responder) | SGW is out of scope here. |
| | | | | BASIC | EN(initiator) EN(responder) | EN.I.1.1.1.3 EN.R.1.1.1.3 |
| 6 | 293 | It is possible in this scenario that one or both of the protected endpoints will be behind a network address translation (NAT) node, in which case the tunneled packets will have to be UDP encapsulated so that port numbers in the UDP headers can be used to identify individual endpoints "behind" the NAT (see section 2.23). | | Not support | | Explanation |
| 6 | 299 | 1.1.3. Endpoint to Security Gateway Tunnel | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 6 | 301 | <pre> +++++++ ! ! IPsec ! ! Protected !Protected! tunnel !Tunnel ! Subnet !Endpoint !<----->!Endpoint !<--- and/or ! ! ! ! ! Internet +++++++ +++++++ </pre> <p>Figure 3: Endpoint to Security Gateway Tunnel</p> | | BASIC | SGW(initiator) SGW(responder) | SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| | | | | ADVANCED | EN(initiator) EN(responder) | EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.2.1.1.1 EN.R.2.1.1.2 |
| 6 | 310 | In this scenario, a protected endpoint (typically a portable roaming computer) connects back to its corporate network through an IPsec-protected tunnel. It might use this tunnel only to access information on the corporate network, or it might tunnel all of its traffic back through the corporate network in order to take advantage of protection provided by a corporate firewall against Internet-based attacks. In either case, the protected endpoint will want an IP address associated with the security gateway so that packets returned to it will go to the security gateway and be tunneled back. This IP address may be static or may be dynamically allocated by the security gateway. In support of the latter case, IKEv2 includes a mechanism for the initiator to request an IP address owned by the security gateway for use for the duration of its SA. | | BASIC | SGW(initiator) SGW(responder) | SGW.I.2.1.1.2 SGW.R.2.1.1.2 |
| | | | | ADVANCED | EN(initiator) EN(responder) | EN.I.2.1.1.2 EN.R.2.1.1.2 |
| 6 | 324 | In this scenario, packets will use tunnel mode. On each packet from the protected endpoint, the outer IP header will contain the source IP address associated with its current location (i.e., the address that will get traffic routed to the endpoint directly), while the inner IP header will contain the source IP address assigned by the security gateway (i.e., the address that will get traffic routed to the security gateway for forwarding to the endpoint). The outer destination address will always be that of the security gateway, while the inner destination address will be the ultimate destination for the packet. | | BASIC | SGW(initiator) SGW(responder) | SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.2.1.1.1 SGW.R.2.1.1.2 SGW.R.2.1.2.1 SGW.R.2.1.2.2 SGW.R.2.1.2.3 |
| | | | | ADVANCED | EN(initiator) EN(responder) | EN.I.2.1.1.2 EN.I.2.1.2.1 EN.I.2.1.2.2 EN.R.2.1.1.1 EN.R.2.1.1.2 |
| 7 | 342 | In this scenario, it is possible that the protected endpoint will be behind a NAT. In that case, the IP address as seen by the security gateway will not be the same as the IP address sent by the protected endpoint, and packets will have to be UDP encapsulated in order to be routed properly. | | Not support | | Explanation |
| 7 | 348 | 1.1.4. Other Scenarios | | | | |
| 7 | 350 | Other scenarios are possible, as are nested combinations of the above. One notable example combines aspects of 1.1.1 and 1.1.3. A subnet may make all external accesses through a remote security gateway using an IPsec tunnel, where the addresses on the subnet are routed to the security gateway by the rest of the Internet. An example would be someone's home network being virtually on the Internet with static IP addresses even though connectivity is provided by an ISP that assigns a single dynamically assigned IP address to the user's security gateway (where the static IP addresses and an IPsec relay are provided by a third party located elsewhere). | | Not support | | Explanation |
| 7 | 361 | 1.2. The Initial Exchanges | | | | |
| 7 | 363 | Communication using IKE always begins with IKE_SA_INIT and IKE_AUTH exchanges (known in IKEv1 as Phase 1). These initial exchanges normally consist of four messages, though in some scenarios that number can grow. | | Not support | | Explanation |
| 7 | 366 | All communications using IKE consist of request/response pairs. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 |
| 7 | 367 | We'll describe the base exchange first, followed by variations. | | Not support | | Explanation |
| 7 | 368 | The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms, exchange nonces, and do a Diffie-Hellman exchange [DH]. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 10 | 536 | If the SA offers include different Diffie-Hellman groups, KEi MUST be an element of the group the initiator expects the responder to accept. | MUST | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.2.3.7 SGW.I.1.2.3.7 |
| 10 | 538 | If it guesses wrong, the CREATE_CHILD_SA exchange will fail, and it will have to retry with a different KEi. | | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.2.3.9 SGW.I.1.2.3.9 |
| 10 | 542 | The message following the header is encrypted and the message including the header is integrity protected using the cryptographic algorithms negotiated for the IKE_SA. | | Not support | | Explanation |
| 10 | 546 | The CREATE_CHILD_SA response contains: | | Not support | | Explanation |
| 10 | 548 | <- HDR, SK {SA, Nr, [KEr], [TSi, TSr]} | | BASIC | EN(responder) SGW(responder) | EN.R.1.2.1.1 EN.R.1.2.7.1 SGW.R.1.2.1.1 SGW.R.1.2.7.1 |
| 10 | 551 | The responder replies (using the same Message ID to respond) with the accepted offer in an SA payload, and a Diffie-Hellman value in the KEr payload if KEi was included in the request and the selected cryptographic suite includes that group. | | Not support | | Explanation |
| 10 | 554 | If the responder chooses a cryptographic suite with a different group, it MUST reject the request. | MUST | ADVANCED | EN(responder) SGW(responder) | EN.R.1.2.5.7 SGW.R.1.2.5.7 |
| 10 | 556 | The initiator SHOULD repeat the request, but now with a KEi payload from the group the responder selected. | SHOULD | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.3.9 SGW.I.1.2.3.9 |
| 11 | 566 | The traffic selectors for traffic to be sent on that SA are specified in the TS payloads, which may be a subset of what the initiator of the CHILD_SA proposed. | | Not support | | Explanation |
| 11 | 568 | Traffic selectors are omitted if this CREATE_CHILD_SA request is being used to change the key of the IKE_SA. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.4.2 EN.R.1.2.6.3 SGW.I.1.2.4.2 SGW.R.1.2.6.3 |
| 11 | 572 | 1.4. The INFORMATIONAL Exchange | | | | |
| 11 | 574 | At various points during the operation of an IKE_SA, peers may desire to convey control messages to each other regarding errors or notifications of certain events. To accomplish this, IKE defines an INFORMATIONAL exchange. | | Not support | | Explanation |
| 11 | 577 | INFORMATIONAL exchanges MUST ONLY occur after the initial exchanges and are cryptographically protected with the negotiated keys. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.3.1.1 SGW.R.1.3.1.1 |
| 11 | 581 | Control messages that pertain to an IKE_SA MUST be sent under that IKE_SA. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.3.1.1 SGW.R.1.3.1.1 |
| 11 | 582 | Control messages that pertain to CHILD_SAs MUST be sent under the protection of the IKE_SA which generated them (or its successor if the IKE_SA was replaced for the purpose of rekeying). | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.9 EN.I.1.1.3.10 EN.R.1.1.3.8 EN.R.1.1.3.9 SGW.I.1.1.3.9 SGW.I.1.1.3.10 SGW.R.1.1.3.8 SGW.R.1.1.3.9 |
| 11 | 586 | Messages in an INFORMATIONAL exchange contain zero or more Notification, Delete, and Configuration payloads. | | Not support | | Explanation |
| 11 | 587 | The Recipient of an INFORMATIONAL exchange request MUST send some response (else the Sender will assume the message was lost in the network and will retransmit it). | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.10 EN.R.1.1.3.9 SGW.I.1.1.3.10 SGW.R.1.1.3.9 |
| 11 | 590 | That response MAY be a message with no payloads. The request message in an INFORMATIONAL exchange MAY also contain no payloads. This is the expected way an endpoint can ask the other endpoint to verify that it is alive. | MAY MAY | BASIC | EN(responder) SGW(responder) | EN.R.1.3.1.1 SGW.R.1.3.1.1 |
| 11 | 595 | ESP and AH SAs always exist in pairs, with one SA in each direction. When an SA is closed, both members of the pair MUST be closed. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.9 EN.I.1.1.3.10 EN.R.1.1.3.8 EN.R.1.1.3.9 SGW.I.1.1.3.9 SGW.I.1.1.3.10 SGW.R.1.1.3.8 SGW.R.1.1.3.9 |
| 11 | 596 | When SAs are nested, as when data (and IP headers if in tunnel mode) are encapsulated first with IPComp, then with ESP, and finally with AH between the same pair of endpoints, all of the SAs MUST be deleted together. | MUST | Not support | | IPComp is "Not support." |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|---------------------------------|--|
| 12 | 663 | Since UDP is a datagram (unreliable) protocol, IKE includes in its definition recovery from transmission errors, including packet loss, packet replay, and packet forgery. IKE is designed to function so long as (1) at least one of a series of retransmitted packets reaches its destination before timing out; and (2) the channel is not so full of forged and replayed packets so as to exhaust the network or CPU capacities of either endpoint. Even in the absence of those minimum performance requirements, IKE is designed to fail cleanly (as though the network were broken). | | Not support | | Explanation |
| 13 | 682 | Although IKEv2 messages are intended to be short, they contain structures with no hard upper bound on size (in particular, X.509 certificates), and IKEv2 itself does not have a mechanism for fragmenting large messages. IP defines a mechanism for fragmentation of oversize UDP messages, but implementations vary in the maximum message size supported. Furthermore, use of IP fragmentation opens an implementation to denial of service attacks [KPS03]. Finally, some NAT and/or firewall implementations may block IP fragments. | | Not support | | Explanation |
| 13 | 691 | All IKEv2 implementations MUST be able to send, receive, and process IKE messages that are up to 1280 bytes long, and they SHOULD be able to send, receive, and process messages that are up to 3000 bytes long. | MUST SHOULD | Not support | | Difficult to test |
| 13 | 694 | IKEv2 implementations SHOULD be aware of the maximum UDP message size supported and MAY shorten messages by leaving out some certificates or cryptographic suite proposals if that will keep messages below the maximum. | SHOULD MAY | Not support | | Not need to test |
| 13 | 697 | Use of the "Hash and URL" formats rather than including certificates in exchanges where possible can avoid most problems. Implementations and configuration should keep in mind, however, that if the URL lookups are possible only after the IPsec SA is established, recursion issues could prevent this technique from working. | | Not support | | Explanation |
| 13 | 704 | 2.1. Use of Retransmission Timers | | | | |
| 13 | 706 | All messages in IKE exist in pairs: a request and a response. The setup of an IKE_SA normally consists of two request/response pairs. Once the IKE_SA is set up, either end of the security association may initiate requests at any time, and there can be many requests and responses "in flight" at any given moment. But each message is labeled as either a request or a response, and for each request/response pair one end of the security association is the initiator and the other is the responder. | | Not support | | Explanation |
| 13 | 715 | For every pair of IKE messages, the initiator is responsible for retransmission in the event of a timeout. | | Not support | | Explanation |
| 13 | 716 | The responder MUST never retransmit a response unless it receives a retransmission of the request. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.2.1 EN.R.1.1.2.2 EN.R.1.2.2.1 EN.R.1.3.2.1 SGW.R.1.1.2.1 SGW.R.1.1.2.2 SGW.R.1.2.2.1 SGW.R.1.3.2.1 |
| 13 | 718 | In that event, the responder MUST ignore the retransmitted request except insofar as it triggers a retransmission of the response. | MUST | Not support | | Difficult to test |
| 13 | 720 | The initiator MUST remember each request until it receives the corresponding response. | MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.2.1 EN.I.1.1.2.3 EN.I.1.2.2.1 SGW.I.1.1.2.1 SGW.I.1.1.2.3 SGW.I.1.2.2.1 |
| 13 | 721 | The responder MUST remember each response until it receives a request whose sequence number is larger than the sequence number in the response plus its window size (see section 2.3). | MUST | Not support | | Window size is "Not support" |
| 14 | 734 | IKE is a reliable protocol, in the sense that the initiator MUST retransmit a request until either it receives a corresponding reply OR it deems the IKE security association to have failed and it discards all state associated with the IKE_SA and any CHILD_SAs negotiated using that IKE_SA. | MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.2.2 EN.I.1.1.2.4 EN.I.1.2.2.2 EN.I.1.3.2.2 SGW.I.1.1.2.2 SGW.I.1.1.2.4 SGW.I.1.2.2.2 SGW.I.1.3.2.2 |
| 14 | 740 | 2.2. Use of Sequence Numbers for Message ID | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 14 | 742 | Every IKE 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. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.2.2 EN.I.1.1.2.4 EN.I.1.2.2.2 EN.I.1.3.2.2 EN.R.1.1.2.1 EN.R.1.1.2.2 EN.R.1.2.2.1 EN.R.1.3.2.1 SGW.I.1.1.2.2 SGW.I.1.1.2.4 SGW.I.1.2.2.2 SGW.I.1.3.2.2 SGW.R.1.1.2.1 SGW.R.1.1.2.2 SGW.R.1.2.2.1 SGW.R.1.3.2.1 |
| 14 | 746 | The Message ID is a 32-bit quantity, which is zero for the first IKE request in each direction. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 SGW.I.1.1.1.1 SGW.R.1.1.1.1 |
| 14 | 747 | The IKE_SA initial setup messages will always be numbered 0 and 1. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.R.1.1.1.2 SGW.I.1.1.1.2 SGW.R.1.1.1.2 |
| 14 | 748 | Each endpoint in the IKE Security Association maintains two "current" Message IDs: the next one to be used for a request it initiates and the next one it expects to see in a request from the other end. | | Not support | | Explanation |
| 14 | 751 | These counters increment as requests are generated and received. Responses always contain the same message ID as the corresponding request. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.R.1.1.1.2 SGW.I.1.1.1.2 SGW.R.1.1.1.2 |
| 14 | 753 | That means that after the initial exchange, each integer n may appear as the message ID in four distinct messages: the nth request from the original IKE initiator, the corresponding response, the nth request from the original IKE responder, and the corresponding response. If the two ends make very different numbers of requests, the Message IDs in the two directions can be very different. There is no ambiguity in the messages, however, because the (I)nitiator and (R)esponse bits in the message header specify which of the four messages a particular one is. | | Not support | | Explanation |
| 14 | 763 | Note that Message IDs are cryptographically protected and provide protection against message replays. | | Not support | | Explanation |
| 14 | 764 | In the unlikely event that Message IDs grow too large to fit in 32 bits, the IKE_SA MUST be closed. Rekeying an IKE_SA resets the sequence numbers. | MUST | Not support | | Difficult to test |
| 14 | 768 | 2.3. Window Size for Overlapping Requests | | | | |
| 14 | 770 | In order to maximize IKE throughput, an IKE endpoint MAY issue multiple requests before getting a response to any of them if the other endpoint has indicated its ability to handle such requests. | MAY | Not support | | Not need to test |
| 14 | 773 | For simplicity, an IKE implementation MAY choose to process requests strictly in order and/or wait for a response to one request before issuing another. Certain rules must be followed to ensure interoperability between implementations using different strategies. | MAY | Not support | | Not need to test |
| 14 | 778 | After an IKE_SA is set up, either end can initiate one or more requests. These requests may pass one another over the network. | | Not support | | Explanation |
| 14 | 779 | An IKE endpoint MUST be prepared to accept and process a request while it has a request outstanding in order to avoid a deadlock in this situation. | MUST | Not support | | Difficult to test |
| 15 | 791 | An IKE endpoint SHOULD be prepared to accept and process multiple requests while it has a request outstanding. | SHOULD | Not support | | Difficult to test |
| 15 | 794 | An IKE endpoint MUST wait for a response to each of its messages before sending a subsequent message unless it has received a SET_WINDOW_SIZE Notify message from its peer informing it that the peer is prepared to maintain state for multiple outstanding messages in order to allow greater throughput. | MUST | Not support | | Difficult to test |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|------------------|------------------|--|--|
| 15 | 800 | An IKE endpoint MUST NOT exceed the peer's stated window size for transmitted IKE requests. In other words, if the responder stated its window size is N, then when the initiator needs to make a request X, it MUST wait until it has received responses to all requests up through request X-N. | MUST NOT MUST | Not support | | window size is "Not support" |
| 15 | 804 | An IKE endpoint MUST keep a copy of (or be able to regenerate exactly) each request it has sent until it receives the corresponding response. | MUST | Not support | | Internal process |
| 15 | 806 | An IKE endpoint MUST keep a copy of (or be able to regenerate exactly) the number of previous responses equal to its declared window size in case its response was lost and the initiator requests its retransmission by retransmitting the request. | MUST | Not support | | Internal process |
| 15 | 811 | An IKE endpoint supporting a window size greater than one SHOULD be capable of processing incoming requests out of order to maximize performance in the event of network failures or packet reordering. | SHOULD | Not support | | window size is "Not support" |
| 15 | 815 | 2.4. State Synchronization and Connection Timeouts | | | | |
| 15 | 817 | An IKE endpoint is allowed to forget all of its state associated with an IKE_SA and the collection of corresponding CHILD_SAs at any time. This is the anticipated behavior in the event of an endpoint crash and restart. It is important when an endpoint either fails or reinitializes its state that the other endpoint detect those conditions and not continue to waste network bandwidth by sending packets over discarded SAs and having them fall into a black hole. | | Not support | | Explanation |
| 15 | 825 | Since IKE is designed to operate in spite of Denial of Service (DoS) attacks from the network, an endpoint MUST NOT conclude that the other endpoint has failed based on any routing information (e.g., ICMP messages) or IKE messages that arrive without cryptographic protection (e.g., Notify messages complaining about unknown SPIs). | MUST NOT | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.1 EN.I.1.1.3.2 EN.R.1.1.3.1 EN.R.1.1.3.2 SGW.I.1.1.3.1 SGW.I.1.1.3.2 SGW.R.1.1.3.1 SGW.R.1.1.3.2 |
| 15 | 830 | An endpoint MUST conclude that the other endpoint has failed only when repeated attempts to contact it have gone unanswered for a timeout period or when a cryptographically protected INITIAL_CONTACT notification is received on a different IKE_SA to the same authenticated identity. | MUST | BASIC | SGW(responder) | SGW.R.1.1.3.3 |
| 15 | 834 | An endpoint SHOULD suspect that the other endpoint has failed based on routing information and initiate a request to see whether the other endpoint is alive. | SHOULD | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.6 EN.R.1.1.3.4 EN.R.1.1.3.5 SGW.I.1.1.3.6 SGW.R.1.1.3.4 SGW.R.1.1.3.5 |
| 15 | 836 | To check whether the other side is alive, IKE specifies an empty INFORMATIONAL message that (like all IKE requests) requires an acknowledgement (note that within the context of an IKE_SA, an "empty" message consists of an IKE header followed by an Encrypted payload that contains no payloads). | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.6 EN.R.1.1.3.4 EN.R.1.1.3.5 SGW.I.1.1.3.6 SGW.R.1.1.3.4 SGW.R.1.1.3.5 |
| 16 | 849 | If a cryptographically protected message has been received from the other side recently, unprotected notifications MAY be ignored. Implementations MUST limit the rate at which they take actions based on unprotected messages. | MAY MUST | Not support | | Not need to test |
| 16 | 854 | Numbers of retries and lengths of timeouts are not covered in this specification because they do not affect interoperability. It is suggested that messages be retransmitted at least a dozen times over a period of at least several minutes before giving up on an SA, but different environments may require different rules. | | Not support | | Explanation |
| 16 | 858 | To be a good network citizen, retransmission times MUST increase exponentially to avoid flooding the network and making an existing congestion situation worse. | MUST | Not support | | Difficult to test |
| 16 | 861 | If there has only been outgoing traffic on all of the SAs associated with an IKE_SA, it is essential to confirm liveness of the other endpoint to avoid black holes. | | Not support | | Explanation |
| 16 | 863 | If no cryptographically protected messages have been received on an IKE_SA or any of its CHILD_SAs recently, the system needs to perform a liveness check in order to prevent sending messages to a dead peer. | | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.3.11 SGW.I.1.1.3.11 |
| 16 | 867 | Receipt of a fresh cryptographically protected message on an IKE_SA or any of its CHILD_SAs ensures liveness of the IKE_SA and all of its CHILD_SAs. | | Not support | | Explanation |
| 16 | 869 | Note that this places requirements on the failure modes of an IKE endpoint. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 16 | 870 | An implementation MUST NOT continue sending on any SA if some failure prevents it from receiving on all of the associated SAs. | MUST NOT | Not support | | Difficult to test |
| 16 | 872 | If CHILD_SAs can fail independently from one another without the associated IKE_SA being able to send a delete message, then they MUST be negotiated by separate IKE_SAs. | MUST | Not support | | Difficult to test |
| 16 | 876 | There is a Denial of Service attack on the initiator of an IKE_SA that can be avoided if the initiator takes the proper care. Since the first two messages of an SA setup are not cryptographically protected, an attacker could respond to the initiator's message before the genuine responder and poison the connection setup attempt. To prevent this, the initiator MAY be willing to accept multiple responses to its first message, treat each as potentially legitimate, respond to it, and then discard all the invalid half-open connections when it receives a valid cryptographically protected response to any one of its requests. Once a cryptographically valid response is received, all subsequent responses should be ignored whether or not they are cryptographically valid. | MAY | Not support | | Not need to test |
| 16 | 889 | Note that with these rules, there is no reason to negotiate and agree upon an SA lifetime. If IKE presumes the partner is dead, based on repeated lack of acknowledgement to an IKE message, then the IKE SA and all CHILD_SAs set up through that IKE_SA are deleted. | | Not support | | Explanation |
| 17 | 902 | An IKE endpoint may at any time delete inactive CHILD_SAs to recover resources used to hold their state. | | Not support | | Explanation |
| 17 | 903 | If an IKE endpoint chooses to delete CHILD_SAs, it MUST send Delete payloads to the other end notifying it of the deletion. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.9 EN.I.1.1.3.10 EN.R.1.1.3.8 EN.R.1.1.3.9 EN.R.1.2.3.1 EN.R.1.2.3.2 SGW.I.1.1.3.9 SGW.I.1.1.3.10 SGW.R.1.1.3.8 SGW.R.1.1.3.9 SGW.R.1.2.3.1 SGW.R.1.2.3.2 |
| 17 | 905 | It MAY similarly time out the IKE_SA. | MAY | Not support | | Not need to test |
| 17 | 906 | Closing the IKE_SA implicitly closes all associated CHILD_SAs. In this case, an IKE endpoint SHOULD send a Delete payload indicating that it has closed the IKE_SA. | SHOULD | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.8 EN.R.1.1.3.6 EN.R.1.1.3.7 SGW.I.1.1.3.8 SGW.R.1.1.3.6 SGW.R.1.1.3.7 |
| 17 | 910 | 2.5. Version Numbers and Forward Compatibility | | | | |
| 17 | 912 | This document describes version 2.0 of IKE, meaning the major version number is 2 and the minor version number is zero. It is likely that some implementations will want to support both version 1.0 and version 2.0, and in the future, other versions. | | Not support | | Explanation |
| 17 | 917 | The major version number should be incremented only if the packet formats or required actions have changed so dramatically that an older version node would not be able to interoperate with a newer version node if it simply ignored the fields it did not understand and took the actions specified in the older specification. | | Not support | | Explanation |
| 17 | 921 | The minor version number indicates new capabilities, and MUST be ignored by a node with a smaller minor version number, but used for informational purposes by the node with the larger minor version number. For example, it might indicate the ability to process a newly defined notification message. The node with the larger minor version number would simply note that its correspondent would not be able to understand that message and therefore would not send it. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.4.1 SGW.R.1.1.4.1 |
| 17 | 930 | If an endpoint receives a message with a higher major version number, it MUST drop the message and SHOULD send an unauthenticated notification message containing the highest version number it supports. | MUST SHOULD | BASIC | EN(responder) SGW(responder) | EN.R.1.1.4.2 SGW.R.1.1.4.2 |
| 17 | 933 | If an endpoint supports major version n, and major version m, it MUST support all versions between n and m. | MUST | Not support | | Current versions are only 1 and 2 |
| 17 | 934 | If it receives a message with a major version that it supports, it MUST respond with that version number. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.1.1 SGW.R.1.1.1.1 |
| 17 | 936 | In order to prevent two nodes from being tricked into corresponding with a lower major version number than the maximum that they both support, IKE has a flag that indicates that the node is capable of speaking a higher major version number. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 17 | 941 | Thus, the major version number in the IKE header indicates the version number of the message, not the highest version number that the transmitter supports. If the initiator is capable of speaking versions n, n+1, and n+2, and the responder is capable of speaking versions n and n+1, then they will negotiate speaking n+1, where the initiator will set the flag indicating its ability to speak a higher version. | | Not support | | Explanation |
| 17 | 947 | If they mistakenly (perhaps through an active attacker sending error messages) negotiate to version n, then both will notice that the other side can support a higher version number, and they MUST break the connection and reconnect using version n+1. | MUST | Not support | | V-bit in IKE header is always "0" at IKEv2 tests. |
| 18 | 962 | Note that IKEv1 does not follow these rules, because there is no way in v1 of noting that you are capable of speaking a higher version number. So an active attacker can trick two v2-capable nodes into speaking v1. When a v2-capable node negotiates down to v1, it SHOULD note that fact in its logs. | SHOULD | Not support | | Internal process |
| 18 | 968 | Also for forward compatibility, all fields marked RESERVED MUST be set to zero by a version 2.0 implementation | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 18 | 969 | and their content MUST be ignored by a version 2.0 implementation ("Be conservative in what you send and liberal in what you receive"). In this way, future versions of the protocol can use those fields in a way that is guaranteed to be ignored by implementations that do not understand them. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.1 EN.I.1.1.11.2 EN.I.1.2.7.1 EN.R.1.1.11.1 EN.R.1.1.11.2 EN.R.1.2.9.1 EN.R.1.3.3.1 SGW.I.1.1.11.1 SGW.I.1.1.11.2 SGW.I.1.2.7.1 SGW.R.1.1.11.1 SGW.R.1.1.11.2 SGW.R.1.2.9.1 SGW.R.1.3.3.1 |
| 18 | 974 | Similarly, payload types that are not defined are reserved for future use; implementations of version 2.0 MUST skip over those payloads and ignore their contents. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.4.1 EN.I.1.1.4.2 EN.R.1.1.4.3 EN.R.1.1.4.4 SGW.I.1.1.4.1 SGW.I.1.1.4.2 SGW.R.1.1.4.3 SGW.R.1.1.4.4 |
| 18 | 978 | IKEv2 adds a "critical" flag to each payload header for further flexibility for forward compatibility. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.4.1 EN.I.1.1.4.2 EN.R.1.1.4.3 EN.R.1.1.4.4 SGW.I.1.1.4.1 SGW.I.1.1.4.2 SGW.R.1.1.4.3 SGW.R.1.1.4.4 |
| 18 | 978 | If the critical flag is set and the payload type is unrecognized, the message MUST be rejected and the response to the IKE request containing that payload MUST include a Notify payload UNSUPPORTED_CRITICAL_PAYLOAD, indicating an unsupported critical payload was included. | MUST MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.4.2 EN.R.1.1.4.4 SGW.I.1.1.4.4 SGW.R.1.1.4.4 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|------------------|------------------|--|---|
| 18 | 979 | If the critical flag is not set and the payload type is unsupported, that payload MUST be ignored. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.4.1 EN.R.1.1.4.3 SGW.I.1.1.4.1 SGW.R.1.1.4.3 |
| 18 | 987 | Although new payload types may be added in the future and may appear interleaved with the fields defined in this specification, implementations MUST send the payloads defined in this specification in the order shown in the figures in section 2 | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 18 | 990 | and implementations SHOULD reject as invalid a message with those payloads in any other order. | SHOULD | BASIC | EN(responder) SGW(responder) | EN.R.1.1.4.5 SGW.R.1.1.4.5 |
| 18 | 994 | 2.6. Cookies | | | | |
| 18 | 996 | The term "cookies" originates with Karn and Simpson [RFC2522] in Photuris, an early proposal for key management with IPsec, and it has persisted. The Internet Security Association and Key Management Protocol (ISAKMP) [MSST98] fixed message header includes two eight-octet fields titled "cookies", and that syntax is used by both IKEv1 and IKEv2 though in IKEv2 they are referred to as the IKE SPI and there is a new separate field in a Notify payload holding the cookie. The initial two eight-octet fields in the header are used as a connection identifier at the beginning of IKE packets. | | Not support | | Explanation |
| 19 | 1014 | Each endpoint chooses one of the two SPIs and SHOULD choose them so as to be unique identifiers of an IKE_SA. An SPI value of zero is special and indicates that the remote SPI value is not yet known by the sender. | SHOULD | Not support | | Internal process |
| 19 | 1018 | Unlike ESP and AH where only the recipient's SPI appears in the header of a message, in IKE the sender's SPI is also sent in every message. Since the SPI chosen by the original initiator of the IKE_SA is always sent first, an endpoint with multiple IKE_SAs open that wants to find the appropriate IKE_SA using the SPI it assigned must look at the I(nitiator) Flag bit in the header to determine whether it assigned the first or the second eight octets. | | Not support | | Explanation |
| 19 | 1026 | In the first message of an initial IKE exchange, the initiator will not know the responder's SPI value and will therefore set that field to zero. | | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.1.1 SGW.I.1.1.1.1 |
| 19 | 1030 | An expected attack against IKE is state and CPU exhaustion, where the target is flooded with session initiation requests from forged IP addresses. This attack can be made less effective if an implementation of a responder uses minimal CPU and commits no state to an SA until it knows the initiator can receive packets at the address from which it claims to be sending them. | | Not support | | Explanation |
| 19 | 1035 | To accomplish this, a responder SHOULD -- when it detects a large number of half-open IKE_SAs -- reject initial IKE messages unless they contain a Notify payload of type COOKIE. It SHOULD instead send an unprotected IKE message as a response and include COOKIE Notify payload with the cookie data to be returned. | SHOULD SHOULD | Not support | | untestable |
| 19 | 1040 | Initiators who receive such responses MUST retry the IKE_SA_INIT with a Notify payload of type COOKIE containing the responder supplied cookie data as the first payload and all other payloads unchanged. The initial exchange will then be as follows: | MUST | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.1.5.1 SGW.I.1.1.5.1 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|---------------------------------------|------------------|---------------------------------|-------------------------------|
| 19 | 1046 | <pre> Initiator Responder ----- - HDR(A,0), SAi1, KEi, Ni --> <-- HDR(A,0), N(COOKIE) HDR(A,0), N(COOKIE), SAi1, KEi, Ni --> <-- HDR(A,B), SAR1, KEr, Nr, [CERTREQ] HDR(A,B), SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr} --> <-- HDR(A,B), SK {IDr, [CERT,] AUTH, SAR2, TSi, TSr} </pre> | | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.1.5.1 SGW.I.1.1.5.1 |
| 20 | 1070 | The first two messages do not affect any initiator or responder state except for communicating the cookie. | | Not support | | Explanation |
| 20 | 1070 | In particular, the message sequence numbers in the first four messages will all be zero | | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.1.5.1 SGW.I.1.1.5.1 |
| 20 | 1070 | and the message sequence numbers in the last two messages will be one. | | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.1.5.1 SGW.I.1.1.5.1 |
| 20 | 1070 | A' is the SPI assigned by the initiator, while 'B' is the SPI assigned by the responder. | | Not support | | Explanation |
| 20 | 1077 | An IKE implementation SHOULD implement its responder cookie generation in such a way as to not require any saved state to recognize its valid cookie when the second IKE_SA_INIT message arrives. The exact algorithms and syntax they use to generate cookies do not affect interoperability and hence are not specified here. The following is an example of how an endpoint could use cookies to implement limited DOS protection. | SHOULD | Not support | | Internal process |
| 20 | 1085 | A good way to do this is to set the responder cookie to be: | | Not support | | Explanation |
| 20 | 1087 | Cookie = <VersionIDofSecret> Hash(Ni IPi SPIi <secret>) | | Not support | | Explanation |
| 20 | 1089 | where <secret> is a randomly generated secret known only to the responder and periodically changed and indicates concatenation. <VersionIDofSecret> should be changed whenever <secret> is regenerated. | | Not support | | Explanation |
| 20 | 1092 | The cookie can be recomputed when the IKE_SA_INIT arrives the second time and compared to the cookie in the received message. If it matches, the responder knows that the cookie was generated since the last change to <secret> and that IPi must be the same as the source address it saw the first time. Incorporating SPIi into the calculation ensures that if multiple IKE_SAs are being set up in parallel they will all get different cookies (assuming the initiator chooses unique SPIi's). Incorporating Ni into the hash ensures that an attacker who sees only message 2 can't successfully forge a message 3. | | Not support | | Explanation |
| 20 | 1103 | If a new value for <secret> is chosen while there are connections in the process of being initialized, an IKE_SA_INIT might be returned with other than the current <VersionIDofSecret>. The responder in that case MAY reject the message by sending another response with a new cookie or it MAY keep the old value of <secret> around for a short time and accept cookies computed from either one. The responder SHOULD NOT accept cookies indefinitely after <secret> is changed, since that would defeat part of the denial of service protection. The responder SHOULD change the value of <secret> frequently, especially if under attack. | MAY MAY SHOULD NOT SHOULD | Not support | | Internal process |
| 21 | 1126 | 2.7. Cryptographic Algorithm Negotiation | | | | |
| 21 | 1128 | The payload type known as "SA" indicates a proposal for a set of choices of IPsec protocols (IKE, ESP, and/or AH) for the SA as well as cryptographic algorithms associated with each protocol. | | Not support | | Explanation |
| 21 | 1132 | An SA payload consists of one or more proposals. Each proposal includes one or more protocols (usually one). 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 identifier does not completely specify the cryptographic algorithm). | | Not support | | Explanation |
| 21 | 1139 | This hierarchical structure was designed to efficiently encode proposals for cryptographic suites when the number of supported suites is large because multiple values are acceptable for multiple transforms. The responder MUST choose a single suite, which MAY be any subset of the SA proposal following the rules below: | MUST MAY | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|---|--|--|
| 21 | 1145 | Each proposal contains one or more protocols. If a proposal is accepted, the SA response MUST contain the same protocols in the same order as the proposal. The responder MUST accept a single proposal or reject them all and return an error. (Example: if a single proposal contains ESP and AH and that proposal is accepted, both ESP and AH MUST be accepted. If ESP and AH are included in separate proposals, the responder MUST accept only one of them). | MUST MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.I.1.1.6.4 EN.I.1.1.6.6 EN.R.1.1.6.1 EN.R.1.1.6.2 EN.R.1.1.6.4 EN.R.1.1.6.6 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.I.1.1.6.4 SGW.I.1.1.6.6 SGW.R.1.1.6.1 SGW.R.1.1.6.2 SGW.R.1.1.6.4 SGW.R.1.1.6.6 |
| 21 | 1153 | 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. | MUST MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.I.1.1.6.3 EN.I.1.1.6.5 EN.R.1.1.6.1 EN.R.1.1.6.2 EN.R.1.1.6.3 EN.R.1.1.6.5 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.I.1.1.6.3 SGW.I.1.1.6.5 SGW.R.1.1.6.1 SGW.R.1.1.6.2 SGW.R.1.1.6.3 SGW.R.1.1.6.5 |
| 21 | 1162 | Since the initiator sends its Diffie-Hellman value in the IKE_SA_INIT, it must guess the Diffie-Hellman group that the responder will select from its list of supported groups. | | Not support | | Explanation |
| 21 | 1164 | If the initiator guesses wrong, the responder will respond with a Notify payload of type INVALID_KEY_PAYLOAD indicating the selected group. | | ADVANCED *Because DH#14 is ADVANCED group. | EN(responder) SGW(responder) | EN.R.1.1.6.7 SGW.R.1.1.6.7 |
| 21 | 1164 | In this case, the initiator MUST retry the IKE_SA_INIT with the corrected Diffie-Hellman group. | MUST | ADVANCED *Because DH#14 is ADVANCED group. | EN(initiator) SGW(initiator) | EN.I.1.1.6.7 SGW.I.1.1.6.7 |
| 21 | 1168 | The initiator MUST again propose its full set of acceptable cryptographic suites because the rejection message was unauthenticated and otherwise an active attacker could trick the endpoints into negotiating a weaker suite than a stronger one that they both prefer. | MUST | Not support | | Difficult to test |
| 22 | 1182 | 2.8. Rekeying | | | | |
| 22 | 1184 | IKE, ESP, and AH security associations use secret keys that SHOULD be used only for a limited amount of time and to protect a limited amount of data. This limits the lifetime of the entire security association | SHOULD | Not support | | Internal process |
| 22 | 1187 | When the lifetime of a security association expires, the security association MUST NOT be used. | MUST NOT | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.3.3 EN.I.1.2.4.3 SGW.I.1.2.3.3 SGW.I.1.2.4.3 |
| 22 | 1189 | If there is demand, new security associations MAY be established. Reestablishment of security associations to take the place of ones that expire is referred to as "rekeying". | MAY | Not support | | Not need to test |
| 22 | 1193 | To allow for minimal IPsec implementations, the ability to rekey SAs without restarting the entire IKE_SA is optional. An implementation MAY refuse all CREATE_CHILD_SA requests within an IKE_SA. | MAY | Not support | | Not need to test |
| 22 | 1194 | If an SA has expired or is about to expire and rekeying attempts using the mechanisms described here fail, an implementation MUST close the IKE_SA and any associated CHILD_SAs and then MAY start new ones. | MUST MAY | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.3.6 SGW.I.1.2.3.6 |
| 22 | 1199 | Implementations SHOULD support in-place rekeying of SAs, since doing so offers better performance and is likely to reduce the number of packets lost during the transition. | SHOULD | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 22 | 1203 | To rekey a CHILD_SA within an existing IKE_SA, create a new, equivalent SA (see section 2.17 below), and when the new one is established, delete the old one. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.3.1 EN.I.1.2.4.1 EN.R.1.2.5.1 EN.R.1.2.6.4 SGW.I.1.2.3.1 SGW.I.1.2.4.1 SGW.R.1.2.5.1 SGW.R.1.2.6.4 |
| 22 | 1203 | To rekey an IKE_SA, establish a new equivalent IKE_SA (see section 2.18 below) with the peer to whom the old IKE_SA is shared using a CREATE_CHILD_SA within the existing IKE_SA. An IKE_SA so created inherits all of the original IKE_SA's CHILD_SAs. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.4.1 EN.R.1.2.6.4 SGW.I.1.2.4.1 SGW.R.1.2.6.4 |
| 22 | 1209 | Use the new IKE_SA for all control messages needed to maintain the CHILD_SAs created by the old IKE_SA, and delete the old IKE_SA. The Delete payload to delete itself MUST be the last request sent over an IKE_SA. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.4.1 EN.R.1.2.6.4 SGW.I.1.2.4.1 SGW.R.1.2.6.4 |
| 22 | 1214 | SAs SHOULD be rekeyed proactively, i.e., the new SA should be established before the old one expires and becomes unusable. Enough time should elapse between the time the new SA is established and the old one becomes unusable so that traffic can be switched over to the new SA. | SHOULD | Not support | | Difficult to test |
| 22 | 1220 | A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. | | Not support | | Explanation |
| 22 | 1223 | If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. | | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.1.1 SGW.I.1.2.1.1 |
| 22 | 1225 | If an SA bundle has been inactive for a long time and if an endpoint would not initiate the SA in the absence of traffic, the endpoint MAY choose to close the SA instead of rekeying it when its lifetime expires. It SHOULD do so if there has been no traffic since the last time the SA was rekeyed. | MAY SHOULD | Not support | | Not need to test |
| 23 | 1238 | If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered (delayed by a random amount of time after the need for rekeying is noticed). | SHOULD | Not support | | Internal process |
| 23 | 1244 | This form of rekeying may temporarily result in multiple similar SAs between the same pairs of nodes. | | Not support | | Explanation |
| 23 | 1245 | When there are two SAs eligible to receive packets, a node MUST accept incoming packets through either SA. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.3.2 EN.I.1.2.3.8 EN.I.1.2.4.2 EN.I.1.2.4.6 EN.R.1.2.5.2 EN.R.1.2.5.6 EN.R.1.2.6.2 EN.R.1.2.6.3 SGW.I.1.2.3.2 SGW.I.1.2.3.8 SGW.I.1.2.4.2 SGW.I.1.2.4.6 SGW.R.1.2.5.2 SGW.R.1.2.5.6 SGW.R.1.2.6.2 SGW.R.1.2.6.3 |
| 23 | 1247 | If redundant SAs are created though such a collision, the SA created with the lowest of the four nonces used in the two exchanges SHOULD be closed by the endpoint that created it. | SHOULD | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.6.3 EN.I.1.2.6.5 SGW.I.1.2.6.3 SGW.I.1.2.6.5 |
| 23 | 1251 | Note that IKEv2 deliberately allows parallel SAs with the same traffic selectors between common endpoints. One of the purposes of this is to support traffic quality of service (QoS) differences among the SAs (see [RFC2474], [RFC2475], and section 4.1 of [RFC2983]). Hence unlike IKEv1, the combination of the endpoints and the traffic selectors may not uniquely identify an SA between those endpoints, so the IKEv1 rekeying heuristic of deleting SAs on the basis of duplicate traffic selectors SHOULD NOT be used. | SHOULD NOT | Not support | | Internal process |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 23 | 1260 | The node that initiated the surviving rekeyed SA SHOULD delete the replaced SA after the new one is established. | SHOULD | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.3.1 EN.I.1.2.4.1 EN.R.1.2.5.1 EN.R.1.2.6.4 SGW.I.1.2.3.1 SGW.I.1.2.4.1 SGW.R.1.2.5.1 SGW.R.1.2.6.4 |
| 23 | 1263 | There are timing windows -- particularly in the presence of lost packets -- where endpoints may not agree on the state of an SA. The responder to a CREATE_CHILD_SA MUST be prepared to accept messages on an SA before sending its response to the creation request, so there is no ambiguity for the initiator. The initiator MAY begin sending on an SA as soon as it processes the response. The initiator, however, cannot receive on a newly created SA until it receives and processes the response to its CREATE_CHILD_SA request. How, then, is the responder to know when it is OK to send on the newly created SA? | MUST MAY | Not support | | Difficult to test |
| 23 | 1273 | From a technical correctness and interoperability perspective, the responder MAY begin sending on an SA as soon as it sends its response to the CREATE_CHILD_SA request. In some situations, however, this could result in packets unnecessarily being dropped, so an implementation MAY want to defer such sending. | MAY MAY | Not support | | Not need to test |
| 23 | 1279 | The responder can be assured that the initiator is prepared to receive messages on an SA if either (1) it has received a cryptographically valid message on the new SA, or (2) the new SA rekeys an existing SA and it receives an IKE request to close the replaced SA. When rekeying an SA, the responder SHOULD continue to send messages on the old SA until one of those events occurs. When establishing a new SA, the responder MAY defer sending messages on a new SA until either it receives one or a timeout has occurred. | SHOULD MAY | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.3.8 EN.I.1.2.4.6 EN.R.1.2.5.6 EN.R.1.2.6.3 SGW.I.1.2.3.8 SGW.I.1.2.4.6 SGW.R.1.2.5.6 SGW.R.1.2.6.3 |
| 24 | 1294 | If an initiator receives a message on an SA for which it has not received a response to its CREATE_CHILD_SA request, it SHOULD interpret that as a likely packet loss and retransmit the CREATE_CHILD_SA request. An initiator MAY send a dummy message on a newly created SA if it has no messages queued in order to assure the responder that the initiator is ready to receive messages. | SHOULD MAY | Not support | | Difficult to test |
| 24 | 1302 | 2.9. Traffic Selector Negotiation | | | | |
| 24 | 1304 | When an IP packet is received by an RFC4301-compliant IPsec subsystem and matches a "protect" selector in its Security Policy Database (SPD), the subsystem MUST protect that packet with IPsec. When no SA exists yet, it is the task of IKE to create it. Maintenance of a system's SPD is outside the scope of IKE (see [PFKEY] for an example protocol), though some implementations might update their SPD in connection with the running of IKE (for an example scenario, see section 1.1.3). | MUST | Not support | | Internal process |
| 24 | 1313 | 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. This can serve as a consistency check in some scenarios to assure that the SPDs are consistent. In others, it guides the dynamic update of the SPD. | | Not support | | Explanation |
| 24 | 1320 | 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. In support of the scenario described in section 1.1.3, an initiator may request that the responder assign an IP address and tell the initiator what it is. | | Not support | | Explanation |
| 24 | 1328 | IKEv2 allows the responder to choose a subset of the traffic proposed by the initiator. This could happen when the configurations of the two endpoints are being updated but only one end has received the new information. Since the two endpoints may be configured by different people, the incompatibility may persist for an extended period even in the absence of errors. It also allows for intentionally different configurations, as when one end is configured to tunnel all addresses and depends on the other end to have the up-to-date list. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 24 | 1337 | The first of the two TS payloads is known as TS _i (Traffic Selector-initiator). The second is known as TS _r (Traffic Selector-responder). TS _i specifies the source address of traffic forwarded from (or the destination address of traffic forwarded to) the initiator of the CHILD_SA pair. TS _r 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. | | Not support | | Explanation |
| 24 | 1337 | For example, if the original initiator request the creation of a CHILD_SA pair, and wishes to tunnel all traffic from subnet 192.0.1.* on the initiator's side to subnet 192.0.2.* on the responder's side, the initiator would include a single traffic selector in each TS payload. TS _i would specify the address range (192.0.1.0 - 192.0.1.255) and TS _r would specify the address range (192.0.2.0 - 192.0.2.255). Assuming that proposal was acceptable to the responder, it would send identical TS payloads back. (Note: The IP address range 192.0.2.* has been reserved for use in examples in RFCs and similar documents. This document needed two such ranges, and so also used 192.0.1.*. This should not be confused with any actual address.) | | Not support | | Explanation |
| 25 | 1364 | The responder is allowed to narrow the choices by selecting a subset of the traffic, for instance by eliminating or narrowing the range of one or more members of the set of traffic selectors, provided the set does not become the NULL set. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.7.1 EN.R.1.1.7.1 SGW.I.1.1.7.1 SGW.R.1.1.7.1 |
| 25 | 1369 | It is possible for the responder's policy to contain multiple smaller ranges, all encompassed by the initiator's traffic selector, and with the responder's policy being that each of those ranges should be sent over a different SA. Continuing the example above, the responder might have a policy of being willing to tunnel those addresses to and from the initiator, but might require that each address pair be on a separately negotiated CHILD_SA. If the initiator generated its request in response to an incoming packet from 192.0.1.43 to 192.0.2.123, there would be no way for the responder to determine which pair of addresses should be included in this tunnel, and it would have to make a guess or reject the request with a status of SINGLE_PAIR_REQUIRED. | | Not support | | Explanation |
| 25 | 1382 | To enable the responder to choose the appropriate range in this case, if the initiator has requested the SA due to a data packet, the initiator SHOULD include as the first traffic selector in each of TS _i and TS _r a very specific traffic selector including the addresses in the packet triggering the request. | SHOULD | Not support | | Internal process |
| 25 | 1386 | In the example, the initiator would include in TS _i two traffic selectors: the first containing the address range (192.0.1.43 - 192.0.1.43) and the source port and IP protocol from the packet and the second containing (192.0.1.0 - 192.0.1.255) with all ports and IP protocols. The initiator would similarly include two traffic selectors in TS _r . | | Not support | | Explanation |
| 25 | 1393 | If the responder's policy does not allow it to accept the entire set of traffic selectors in the initiator's request, but does allow him to accept the first selector of TS _i and TS _r , then the responder MUST narrow the traffic selectors to a subset that includes the initiator's first choices. | MUST | Not support | | Explanation |
| 26 | 1406 | In this example, the responder might respond with TS _i being (192.0.1.43 - 192.0.1.43) with all ports and IP protocols. | | Not support | | Explanation |
| 26 | 1410 | 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 TS _i and TS _r MAY be ranges rather than specific values, and the responder chooses a subset of the initiator's TS _i and TS _r that are acceptable. | MAY | Not support | | Not need to test |
| 26 | 1415 | If more than one subset is acceptable but their union is not, the responder MUST accept some subset and MAY include a Notify payload of type ADDITIONAL_TS_POSSIBLE to indicate that the initiator might want to try again. This case will occur only when the initiator and responder are configured differently from one another. | MUST MAY | Not support | | Internal process |
| 26 | 1420 | If the initiator and responder agree on the granularity of tunnels, the initiator will never request a tunnel wider than the responder will accept. Such misconfigurations SHOULD be recorded in error logs. | SHOULD | Not support | | Internal process |
| 26 | 1425 | 2.10. Nonces | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|----------------------|------------------|--|--|
| 26 | 1427 | The IKE_SA_INIT messages each contain a nonce. These nonces are used as inputs to cryptographic functions. The CREATE_CHILD_SA request and the CREATE_CHILD_SA response also contain nonces. These nonces are used to add freshness to the key derivation technique used to obtain keys for CHILD_SA, and to ensure creation of strong pseudo-random bits from the Diffie-Hellman key. | | Not support | | explanation |
| 26 | 1432 | Nonces used in IKEv2 MUST be randomly chosen, MUST be at least 128 bits in size, and MUST be at least half the key size of the negotiated prf. ("prf" refers to "pseudo-random function", one of the cryptographic algorithms negotiated in the IKE exchange.) | MUST MUST MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 SGW.I.1.1.1.1 SGW.R.1.1.1.1 |
| 26 | 1436 | If the same random number source is used for both keys and nonces, care must be taken to ensure that the latter use does not compromise the former. | | Not support | | Explanation |
| 26 | 1440 | 2.11. Address and Port Agility | | | | |
| 26 | 1442 | IKE runs over UDP ports 500 and 4500, and implicitly sets up ESP and AH associations for the same IP addresses it runs over. The IP addresses and ports in the outer header are, however, not themselves cryptographically protected, and IKE is designed to work even through Network Address Translation (NAT) boxes. | | Not support | | Explanation |
| 26 | 1446 | An implementation MUST accept incoming requests even if the source port is not 500 or 4500, and MUST respond to the address and port from which the request was received. It MUST specify the address and port at which the request was received as the source address and port in the response. | MUST MUST MUST | Not support | | NAT traversal is "Not support" |
| 26 | 1450 | IKE functions identically over IPv4 or IPv6. | | Not support | | Explanation |
| 27 | 1462 | 2.12. Reuse of Diffie-Hellman Exponentials | | | | |
| 27 | 1464 | IKE generates keying material using an ephemeral Diffie-Hellman exchange in order to gain the property of "perfect forward secrecy". This means that once a connection is closed and its corresponding keys are forgotten, even someone who has recorded all of the data from the connection and gets access to all of the long-term keys of the two endpoints cannot reconstruct the keys used to protect the conversation without doing a brute force search of the session key space. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.3.7 EN.R.1.2.5.5 SGW.I.1.2.3.7 SGW.R.1.2.5.5 |
| 27 | 1473 | Achieving perfect forward secrecy requires that when a connection is closed, each endpoint MUST forget not only the keys used by the connection but also any information that could be used to recompute those keys. In particular, it MUST forget the secrets used in the Diffie-Hellman calculation and any state that may persist in the state of a pseudo-random number generator that could be used to recompute the Diffie-Hellman secrets. | MUST MUST | Not support | | Internal process |
| 27 | 1481 | Since the computing of Diffie-Hellman exponentials is computationally expensive, an endpoint may find it advantageous to reuse those exponentials for multiple connection setups. There are several reasonable strategies for doing this. An endpoint could choose a new exponential only periodically though this could result in less-than-perfect forward secrecy if some connection lasts for less than the lifetime of the exponential. Or it could keep track of which exponential was used for each connection and delete the information associated with the exponential only when some corresponding connection was closed. This would allow the exponential to be reused without losing perfect forward secrecy at the cost of maintaining more state. | | Not support | | Explanation |
| 27 | 1494 | Decisions as to whether and when to reuse Diffie-Hellman exponentials is a private decision in the sense that it will not affect interoperability. An implementation that reuses exponentials MAY choose to remember the exponential used by the other endpoint on past exchanges and if one is reused to avoid the second half of the calculation. | MAY | Not support | | Not need to test |
| 27 | 1501 | 2.13. Generating Keying Material | | | | |
| 27 | 1503 | In the context of the IKE_SA, four cryptographic algorithms are negotiated: an encryption algorithm, an integrity protection algorithm, a Diffie-Hellman group, and a pseudo-random function (prf). The pseudo-random function is used for the construction of keying material for all of the cryptographic algorithms used in both the IKE_SA and the CHILD_SAs. | | Not support | | Explanation |
| 28 | 1518 | We assume that each encryption algorithm and integrity protection algorithm uses a fixed-size key and that any randomly chosen value of that fixed size can serve as an appropriate key. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 28 | 1520 | For algorithms that accept a variable length key, a fixed key size MUST be specified as part of the cryptographic transform negotiated. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 28 | 1522 | For algorithms for which not all values are valid keys (such as DES or 3DES with key parity), the algorithm by which keys are derived from arbitrary values MUST be specified by the cryptographic transform. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 28 | 1525 | For integrity protection functions based on Hashed Message Authentication Code (HMAC), the fixed key size is the size of the output of the underlying hash function. When the prf function takes a variable length key, variable length data, and produces a fixed-length output (e.g., when using HMAC), the formulas in this document apply. When the key for the prf function has fixed length, the data provided as a key is truncated or padded with zeros as necessary unless exceptional processing is explained following the formula. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.1 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 28 | 1535 | Keying material will always be derived as the output of the negotiated prf algorithm. Since the amount of keying material needed may be greater than the size of the output of the prf algorithm, we will use the prf iteratively. We will use the terminology prf+ to describe the function that outputs a pseudo-random stream based on the inputs to a prf as follows: (where indicates concatenation) | | Not support | | Explanation |
| 28 | 1542 | prf+ (K,S) = T1 T2 T3 T4 ... | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.1 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|---|
| 28 | 1544 | where: T1 = prf (K, S 0x01) T2 = prf (K, T1 S 0x02) T3 = prf (K, T2 S 0x03) T4 = prf (K, T3 S 0x04) | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 28 | 1550 | continuing as needed to compute all required keys. The keys are taken from the output string without regard to boundaries (e.g., if the required keys are a 256-bit Advanced Encryption Standard (AES) key and a 160-bit HMAC key, and the prf function generates 160 bits, the AES key will come from T1 and the beginning of T2, while the HMAC key will come from the rest of T2 and the beginning of T3). | | Not support | | Explanation |
| 28 | 1557 | The constant concatenated to the end of each string feeding the prf is a single octet. prf+ in this document is not defined beyond 255 times the size of the prf output. | | Not support | | Explanation |
| 28 | 1561 | 2.14. Generating Keying Material for the IKE_SA | | | | |
| 28 | 1563 | The shared keys are computed as follows. A quantity called SKEYSEED is calculated from the nonces exchanged during the IKE_SA_INIT exchange and the Diffie-Hellman shared secret established during that exchange. | | Not support | | Explanation |
| 29 | 1574 | SKEYSEED is used to calculate seven other secrets: SK_d used for deriving new keys for the CHILD_SAs established with this IKE_SA; SK_ai and SK_ar used as a key to the integrity protection algorithm for authenticating the component messages of subsequent exchanges; SK_ei and SK_er used for encrypting (and of course decrypting) all subsequent exchanges; and SK_pi and SK_pr, which are used when generating an AUTH payload. | | Not support | | Explanation |
| 29 | 1582 | SKEYSEED and its derivatives are computed as follows: | | Not support | | Explanation |
| 29 | 1584 | SKEYSEED = prf(Ni Nr, g ^{ir}) | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|---------------------------------|---|
| 29 | 1586 | {SK_d SK_ai SK_ar SK_ei SK_er SK_pi SK_pr} = prf+(SKEYSEED, Ni Nr SPI SPIr) | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 29 | 1589 | (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. If the negotiated prf takes a fixed-length key and the lengths of Ni and Nr do not add up to that length, half the bits must come from Ni and half from Nr, taking the first bits of each. | | Not support | | Explanation |
| 29 | 1599 | The two directions of traffic flow use different keys. 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. Each algorithm takes a fixed number of bits of keying material, which is specified as part of the algorithm. For integrity algorithms based on a keyed hash, the key size is always equal to the length of the output of the underlying hash function. | | Not support | | Explanation |
| 29 | 1607 | 2.15. Authentication of the IKE_SA | | | | |
| 29 | 1609 | When not using extensible authentication (see section 2.16), the peers are authenticated by having each sign (or MAC using a shared secret as the key) a block of data. | | Not support | | Explanation |
| 29 | 1611 | For the responder, the octets to be signed start with the first octet of the first SPI in the header of the second message and end with the last octet of the last payload in the second message. Appended to this (for purposes of computing the signature) are the initiator's nonce Ni (just the value, not the payload containing it), and the value prf(SK_pr, IDr') where IDr' is the responder's ID payload excluding the fixed header. Note that neither the nonce Ni nor the value prf(SK_pr, IDr') are transmitted. | | BASIC | EN(responder) SGW(responder) | EN.R.1.1.1.2 EN.R.2.1.1.1 SGW.R.1.1.1.2 SGW.R.2.1.1.1 |
| 29 | 1619 | Similarly, the initiator signs the first message, starting with the first octet of the first SPI in the header and ending with the last octet of the last payload. Appended to this (for purposes of computing the signature) are the responder's nonce Nr, and the value prf(SK_pi, IDi'). | | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.1.2 EN.I.2.1.1.1 SGW.I.1.1.1.2 SGW.I.2.1.1.1 |
| 30 | 1630 | In the above calculation, IDi' and IDr' are the entire ID payloads excluding the fixed header. It is critical to the security of the exchange that each side sign the other side's nonce. | | Not support | | Explanation |
| 30 | 1635 | Note that all of the payloads are included under the signature, including any payload types not defined in this document. If the first message of the exchange is sent twice (the second time with a responder cookie and/or a different Diffie-Hellman group), it is the second version of the message that is signed. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason | |
|------|------|---|-----------------|------------------|--|--|--|
| 30 | 1641 | Optionally, messages 3 and 4 MAY include a certificate, or certificate chain providing evidence that the key used to compute a digital signature belongs to the name in the ID payload. The signature or MAC will be computed using algorithms dictated by the type of key used by the signer, and specified by the Auth Method field in the Authentication payload. There is no requirement that the initiator and responder sign with the same cryptographic algorithms. The choice of cryptographic algorithms depends on the type of key each has. In particular, the initiator may be using a shared key while the responder may have a public signature key and certificate. It will commonly be the case (but it is not required) that if a shared secret is used for authentication that the same key is used in both directions. Note that it is a common but typically insecure practice to have a shared key derived solely from a user-chosen password without incorporating another source of randomness. | MAY | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.1 EN.I.1.1.10.2 EN.I.1.1.10.3 EN.R.1.1.10.1 EN.R.1.1.10.2 EN.R.1.1.10.3 SGW.I.1.1.10.1 SGW.I.1.1.10.2 SGW.I.1.1.10.3 SGW.R.1.1.10.1 SGW.R.1.1.10.2 SGW.R.1.1.10.3 | |
| 30 | 1657 | This is typically insecure because user-chosen passwords are unlikely to have sufficient unpredictability to resist dictionary attacks and these attacks are not prevented in this authentication method. (Applications using password-based authentication for bootstrapping and IKE_SA should use the authentication method in section 2.16, which is designed to prevent off-line dictionary attacks.) The pre-shared key SHOULD contain as much unpredictability as the strongest key being negotiated. | SHOULD | Not support | | Difficult to test | |
| 30 | 1664 | In the case of a pre-shared key, the AUTH value is computed as: AUTH = prf(prf(Shared Secret,"Key Pad for IKEv2"), <msg octets>) where the string "Key Pad for IKEv2" is 17 ASCII characters without null termination. The shared secret can be variable length. The pad string is added so that if the shared secret is derived from a password, the IKE implementation need not store the password in cleartext, but rather can store the value prf(Shared Secret,"Key Pad for IKEv2"), which could not be used as a password equivalent for protocols other than IKEv2. As noted above, deriving the shared secret from a password is not secure. This construction is used because it is anticipated that people will do it anyway . | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.2.1.1.1 EN.R.1.1.1.2 EN.R.2.1.1.1 SGW.I.1.1.1.2 SGW.I.2.1.1.1 SGW.R.1.1.1.2 SGW.R.2.1.1.1 | |
| 31 | 1686 | The management interface by which the Shared Secret is provided MUST accept ASCII strings of at least 64 octets | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.2.1.1.1 EN.R.1.1.1.2 EN.R.2.1.1.1 SGW.I.1.1.1.2 SGW.I.2.1.1.1 SGW.R.1.1.1.2 SGW.R.2.1.1.1 | |
| 31 | 1687 | and MUST NOT add a null terminator before using them as shared secrets. | MUST NOT | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.2.1.1.1 EN.R.1.1.1.2 EN.R.2.1.1.1 SGW.I.1.1.1.2 SGW.I.2.1.1.1 SGW.R.1.1.1.2 SGW.R.2.1.1.1 | |
| 31 | 1688 | It MUST also accept a HEX encoding of the Shared Secret. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.4 EN.R.1.1.10.4 SGW.I.1.1.10.4 SGW.R.1.1.10.4 | |
| 31 | 1689 | The management interface MAY accept other encodings if the algorithm for translating the encoding to a binary string is specified. | MAY | Not support | | Not need to test | |
| 31 | 1691 | If the negotiated prf takes a fixed-size key, the shared secret MUST be of that fixed size. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.2.1.1.1 EN.R.1.1.1.2 EN.R.2.1.1.1 SGW.I.1.1.1.2 SGW.I.2.1.1.1 SGW.R.1.1.1.2 SGW.R.2.1.1.1 | |
| 31 | 1694 | 2.16. Extensible Authentication Protocol Methods | | | | | |
| 31 | 1696 | In addition to authentication using public key signatures and shared secrets, IKE supports authentication using methods defined in RFC 3748 [EAP]. Typically, these methods are asymmetric (designed for a user authenticating to a server), and they may not be mutual. | | Not support | | Explanation | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|------------------|------------------|--------|-----------------------|
| 31 | 1699 | For this reason, these protocols are typically used to authenticate the initiator to the responder and MUST be used in conjunction with a public key signature based authentication of the responder to the initiator. | MUST | Not support | | EAP is "Not support." |
| 31 | 1703 | These methods are often associated with mechanisms referred to as "Legacy Authentication" mechanisms. | | Not support | | Explanation |
| 31 | 1706 | While this memo references [EAP] with the intent that new methods can be added in the future without updating this specification, some simpler variations are documented here and in section 3.16. [EAP] defines an authentication protocol requiring a variable number of messages. | | Not support | | Explanation |
| 31 | 1710 | Extensible Authentication is implemented in IKE as additional IKE_AUTH exchanges that MUST be completed in order to initialize the IKE_SA. | MUST | Not support | | EAP is "Not support." |
| 31 | 1714 | An initiator indicates a desire to use extensible authentication by leaving out the AUTH payload from message 3. By including an IDi payload but not an AUTH payload, the initiator has declared an identity but has not proven it. If the responder is willing to use an extensible authentication method, it will place an Extensible Authentication Protocol (EAP) payload in message 4 and defer sending SAr2, TSi, and TSr until initiator authentication is complete in a subsequent IKE_AUTH exchange. In the case of a minimal extensible authentication, the initial SA establishment will appear as follows: | | Not support | | Explanation |
| 32 | 1742 | <pre> Initiator Responder ----- - HDR, SAi1, KEi, Ni --> <-- HDR, SAr1, KEr, Nr, [CERTREQ] HDR, SK {IDi, [CERTREQ,] [IDr, SAi2, TSi, TSr} --> <-- HDR, SK {IDr, [CERT,] AUTH, EAP } HDR, SK {EAP} --> <-- HDR, SK {EAP (success)} HDR, SK {AUTH} --> <-- HDR, SK {AUTH, SAr2, TSi, TSr } </pre> | | Not support | | Explanation |
| 32 | 1762 | For EAP methods that create a shared key as a side effect of authentication, that shared key MUST be used by both the initiator and responder to generate AUTH payloads in messages 7 and 8 using the syntax for shared secrets specified in section 2.15. The shared key from EAP is the field from the EAP specification named MSK. The shared key generated during an IKE exchange MUST NOT be used for any other purpose. | MUST MUST NOT | Not support | | EAP is "Not support." |
| 32 | 1770 | EAP methods that do not establish a shared key SHOULD NOT be used, as they are subject to a number of man-in-the-middle attacks [EAPMITM] if these EAP methods are used in other protocols that do not use a server-authenticated tunnel. Please see the Security Considerations section for more details. | SHOULD NOT | Not support | | EAP is "Not support." |
| 32 | 1774 | If EAP methods that do not generate a shared key are used, the AUTH payloads in messages 7 and 8 MUST be generated using SK_pi and SK_pr, respectively. | MUST | Not support | | EAP is "Not support." |
| 32 | 1778 | The initiator of an IKE_SA using EAP SHOULD be capable of extending the initial protocol exchange to at least ten IKE_AUTH exchanges in the event the responder sends notification messages and/or retries the authentication prompt. | SHOULD | Not support | | EAP is "Not support." |
| 32 | 1781 | Once the protocol exchange defined by the chosen EAP authentication method has successfully terminated, the responder MUST send an EAP payload containing the Success message. | MUST | Not support | | EAP is "Not support." |
| 32 | 1784 | Similarly, if the authentication method has failed, the responder MUST send an EAP payload containing the Failure message. | MUST | Not support | | EAP is "Not support." |
| 32 | 1785 | The responder MAY at any time terminate the IKE exchange by sending an EAP payload containing the Failure message. | MAY | Not support | | EAP is "Not support." |
| 33 | 1798 | Following such an extended exchange, the EAP AUTH payloads MUST be included in the two messages following the one containing the EAP Success message. | MUST | Not support | | EAP is "Not support." |
| 33 | 1802 | 2.17. Generating Keying Material for CHILD_SAs | | | | |
| 33 | 1804 | A single CHILD_SA is created by the IKE_AUTH exchange, and additional CHILD_SAs can optionally be created in CREATE_CHILD_SA exchanges. Keying material for them is generated as follows: | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 33 | 1808 | KEYMAT = prf+(SK_d, Ni Nr) | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.3 EN.R.1.1.1.3 SGW.I.1.1.1.3 SGW.R.1.1.1.3 |
| 33 | 1810 | Where Ni and Nr are the nonces from the IKE_SA_INIT exchange if this request is the first CHILD_SA created or the fresh Ni and Nr from the CREATE_CHILD_SA exchange if this is a subsequent creation. | | Not support | | Explanation |
| 33 | 1814 | For CREATE_CHILD_SA exchanges including an optional Diffie-Hellman exchange, the keying material is defined as: | | Not support | | Explanation |
| 33 | 1817 | KEYMAT = prf+(SK_d, g^ir (new) Ni Nr) | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.3.7 EN.R.1.2.5.5 SGW.I.1.2.3.7 SGW.R.1.2.5.5 |
| 33 | 1819 | where g^ir (new) is the shared secret from the ephemeral Diffie-Hellman exchange of this CREATE_CHILD_SA exchange (represented as an octet string in big endian order padded with zeros in the high-order bits if necessary to make it the length of the modulus). | | Not support | | Explanation |
| 33 | 1824 | A single CHILD_SA negotiation may result in multiple security associations. ESP and AH SAs exist in pairs (one in each direction), and four SAs could be created in a single CHILD_SA negotiation if a combination of ESP and AH is being negotiated. | | Not support | | Explanation |
| 33 | 1829 | Keying material MUST be taken from the expanded KEYMAT in the following order: | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.3 EN.R.1.1.1.3 SGW.I.1.1.1.3 SGW.R.1.1.1.3 |
| 33 | 1832 | All keys for SAs carrying data from the initiator to the responder are taken before SAs going in the reverse direction. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.3 EN.R.1.1.1.3 SGW.I.1.1.1.3 SGW.R.1.1.1.3 |
| 33 | 1835 | If multiple IPsec protocols are negotiated, keying material is taken in the order in which the protocol headers will appear in the encapsulated packet. | | Not support | | Explanation |
| 33 | 1839 | If a single protocol has both encryption and authentication keys, the encryption key is taken from the first octets of KEYMAT and the authentication key is taken from the next octets. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.3 EN.R.1.1.1.3 SGW.I.1.1.1.3 SGW.R.1.1.1.3 |
| 33 | 1843 | Each cryptographic algorithm takes a fixed number of bits of keying material specified as part of the algorithm. | | Not support | | Explanation |
| 34 | 1854 | 2.18. Rekeying IKE SAs Using a CREATE_CHILD_SA exchange | | | | |
| 34 | 1856 | The CREATE_CHILD_SA exchange can be used to rekey an existing IKE_SA (see section 2.8). New initiator and responder SPIs are supplied in the SPI fields. The TS payloads are omitted when rekeying an IKE_SA. SKEYSEED for the new IKE_SA is computed using SK_d from the existing IKE_SA as follows: | | Not support | | Explanation |
| 34 | 1862 | SKEYSEED = prf(SK_d (old), [g^ir (new)] Ni Nr) | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.4.2 EN.R.1.2.6.3 SGW.I.1.2.4.2 SGW.R.1.2.6.3 |
| 34 | 1864 | where g^ir (new) is the shared secret from the ephemeral Diffie-Hellman exchange of this CREATE_CHILD_SA exchange (represented as an octet string in big endian order padded with zeros if necessary to make it the length of the modulus) and Ni and Nr are the two nonces stripped of any headers. | | Not support | | Explanation |
| 34 | 1870 | The new IKE_SA MUST reset its message counters to 0. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.4.2 EN.R.1.2.6.3 SGW.I.1.2.4.2 SGW.R.1.2.6.3 |
| 34 | 1872 | SK_d, SK_ai, SK_ar, SK_ei, and SK_er are computed from SKEYSEED as specified in section 2.14. | | Not support | | Explanation |
| 34 | 1875 | 2.19. Requesting an Internal Address on a Remote Network | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|---------------------------------|---|
| 34 | 1877 | Most commonly occurring in the endpoint-to-security-gateway scenario, an endpoint may need an IP address in the network protected by the security gateway and may need to have that address dynamically assigned. A request for such a temporary address can be included in any request to create a CHILD_SA (including the implicit request in message 3) by including a CP payload. | | Not support | | Explanation |
| 34 | 1884 | This function provides address allocation to an IPsec Remote Access Client (IRAC) trying to tunnel into a network protected by an IPsec Remote Access Server (IRAS). | | Not support | | Explanation |
| 34 | 1886 | Since the IKE_AUTH exchange creates an IKE_SA and a CHILD_SA, the IRAC MUST request the IRAS-controlled address (and optionally other information concerning the protected network) in the IKE_AUTH exchange. | MUST | ADVANCED | EN(initiator) | EN.I.2.1.2.1 |
| 34 | 1889 | The IRAS may procure an address for the IRAC from any number of sources such as a DHCP/BOOTP server or its own address pool. | | Not support | | Explanation |
| 34 | 1893 | <pre> Initiator Responder ----- ----- HDR, SK {IDi, [CERT.] [CERTREQ.] [IDr.] AUTH, CP(CFG_REQUEST), SAi2, TSi, TSr} --> <-- HDR, SK {IDr, [CERT.] AUTH, CP(CFG_REPLY), SAr2, TSi, TSr} </pre> | | ADVANCED | EN(initiator) SGW(responder) | EN.I.2.1.2.1 EN.I.2.1.2.2 SGW.R.2.1.2.1 |
| 35 | 1910 | In all cases, the CP payload MUST be inserted before the SA payload. In variations of the protocol where there are multiple IKE_AUTH exchanges, the CP payloads MUST be inserted in the messages containing the SA payloads. | MUST MUST | ADVANCED | EN(initiator) SGW(responder) | EN.I.2.1.2.1 SGW.R.2.1.2.1 |
| 35 | 1915 | CP(CFG_REQUEST) MUST contain at least an INTERNAL_ADDRESS attribute (either IPv4 or IPv6) | MUST | ADVANCED | EN(initiator) | EN.I.2.1.2.1 |
| 35 | 1916 | but MAY contain any number of additional attributes the initiator wants returned in the response. | MAY | Not support | | Not need to test |
| 35 | 1919 | For example, message from initiator to responder: CP(CFG_REQUEST)= INTERNAL_ADDRESS(0.0.0.0) INTERNAL_NETMASK(0.0.0.0) INTERNAL_DNS(0.0.0.0) TSi = (0, 0-65535,0.0.0.0-255.255.255.255) TSr = (0, 0-65535,0.0.0.0-255.255.255.255) | | Not support | | Explanation |
| 35 | 1927 | NOTE: Traffic Selectors contain (protocol, port range, address range). | | Not support | | Explanation |
| 35 | 1930 | Message from responder to initiator: | | Not support | | Explanation |
| 35 | 1932 | CP(CFG_REPLY)= INTERNAL_ADDRESS(192.0.2.202) INTERNAL_NETMASK(255.255.255.0) INTERNAL_SUBNET(192.0.2.0/255.255.255.0) TSi = (0, 0-65535,192.0.2.202-192.0.2.202) TSr = (0, 0-65535,192.0.2.0-192.0.2.255) | | Not support | | Explanation |
| 35 | 1939 | All returned values will be implementation dependent. As can be seen in the above example, the IRAS MAY also send other attributes that were not included in CP(CFG_REQUEST) and MAY ignore the non-mandatory attributes that it does not support. | MAY | Not support | | Not need to test |
| 35 | 1944 | The responder MUST NOT send a CFG_REPLY without having first received a CP(CFG_REQUEST) from the initiator, because we do not want the IRAS to perform an unnecessary configuration lookup if the IRAC cannot process the REPLY. | MUST NOT | ADVANCED | SGW(responder) | SGW.R.2.1.2.3 |
| 35 | 1947 | In the case where the IRAS's configuration requires that CP be used for a given identity IDi, but IRAC has failed to send a CP(CFG_REQUEST), IRAS MUST fail the request, and terminate the IKE exchange with a FAILED_CP_REQUIRED error. | MUST | ADVANCED | SGW(responder) | SGW.R.2.1.2.3 |
| 35 | 1952 | 2.20. Requesting the Peer's Version | | | | |
| 35 | 1954 | An IKE peer wishing to inquire about the other peer's IKE software version information MAY use the method below. This is an example of a configuration request within an INFORMATIONAL exchange, after the IKE_SA and first CHILD_SA have been created. | MAY | Not support | | Not need to test |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|----------------------------|------------------|--|---|
| 36 | 1966 | An IKE implementation MAY decline to give out version information prior to authentication or even after authentication to prevent trolling in case some implementation is known to have some security weakness. In that case, it MUST either return an empty string or no CP payload if CP is not supported. | MAY MUST | Not support | | Not need to test |
| 36 | 1972 | <pre> Initiator Responder ----- ----- HDR, SK{CP(CFG_REQUEST)} --> <-- HDR, SK{CP(CFG_REPLY)} CP(CFG_REQUEST)= APPLICATION_VERSION("") CP(CFG_REPLY) APPLICATION_VERSION("foobar v1.3beta, (c) Foo Bar Inc.") </pre> | | Not support | | Explanation |
| 36 | 1983 | 2.21. Error Handling | | | | |
| 36 | 1985 | There are many kinds of errors that can occur during IKE processing. | | Not support | | Explanation |
| 36 | 1986 | If a request is received that is badly formatted or unacceptable for reasons of policy (e.g., no matching cryptographic algorithms), the response MUST contain a Notify payload indicating the error. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.7 EN.R.1.1.4.2 EN.R.1.1.4.4 EN.R.1.1.6.7 EN.R.1.1.7.2 EN.R.1.2.4.1 EN.R.1.2.8.1 SGW.I.1.1.6.7 SGW.R.1.1.4.2 SGW.R.1.1.4.4 SGW.R.1.1.6.7 SGW.R.1.1.7.2 SGW.R.1.2.4.1 SGW.R.1.2.8.1 |
| 36 | 1988 | If an error occurs outside the context of an IKE request (e.g., the node is getting ESP messages on a nonexistent SPD), the node SHOULD initiate an INFORMATIONAL exchange with a Notify payload describing the problem. | SHOULD | Not support | | untestable |
| 36 | 1994 | Errors that occur before a cryptographically protected IKE_SA is established must be handled very carefully. There is a trade-off between wanting to be helpful in diagnosing a problem and responding to it and wanting to avoid being a dupe in a denial of service attack based on forged messages. | | Not support | | Explanation |
| 36 | 2000 | If a node receives a message on UDP port 500 or 4500 outside the context of an IKE_SA known to it (and not a request to start one), it may be the result of a recent crash of the node. | | Not support | | Explanation |
| 36 | 2002 | If the message is marked as a response, the node MAY audit the suspicious event but MUST NOT respond. | MAY MUST NOT | Not support | | untestable |
| 36 | 2004 | If the message is marked as a request, the node MAY audit the suspicious event and MAY send a response. | MAY MAY | Not support | | Not need to test |
| 36 | 2005 | If a response is sent, the response MUST be sent to the IP address and port from whence it came with the same IKE SPIs and the Message ID copied. The response MUST NOT be cryptographically protected and MUST contain a Notify payload indicating INVALID_IKE_SPI. | MUST MUST NOT MUST | Not support | | untestable |
| 36 | 2011 | A node receiving such an unprotected Notify payload MUST NOT respond and MUST NOT change the state of any existing SAs. The message might be a forgery or might be a response the genuine correspondent was tricked into sending. | MUST NOT MUST NOT | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.1 EN.I.1.1.3.2 EN.R.1.1.3.1 EN.R.1.1.3.2 SGW.I.1.1.3.1 SGW.I.1.1.3.2 SGW.R.1.1.3.1 SGW.R.1.1.3.2 |
| 37 | 2022 | A node SHOULD treat such a message (and also a network message like ICMP destination unreachable) as a hint that there might be problems with SAs to that IP address and SHOULD initiate a liveness test for any such IKE_SA. An implementation SHOULD limit the frequency of such tests to avoid being tricked into participating in a denial of service attack. | SHOULD SHOULD SHOULD | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.1 EN.I.1.1.3.2 EN.R.1.1.3.1 EN.R.1.1.3.2 SGW.I.1.1.3.1 SGW.I.1.1.3.2 SGW.R.1.1.3.1 SGW.R.1.1.3.2 |
| 37 | 2029 | A node receiving a suspicious message from an IP address with which it has an IKE_SA MAY send an IKE Notify payload in an IKE INFORMATIONAL exchange over that SA. | MAY | Not support | | Not need to test |
| 37 | 2031 | The recipient MUST NOT change the state of any SA's as a result but SHOULD audit the event to aid in diagnosing malfunctions. | MUST NOT SHOULD | Not support | | Internal process |
| 37 | 2033 | A node MUST limit the rate at which it will send messages in response to unprotected messages. | MUST | Not support | | Internal process |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|----------------------------------|------------------|--------|--------------------------|
| 37 | 2036 | 2.22. IPComp | | | | |
| 37 | 2038 | Use of IP compression [IPCOMP] can be negotiated as part of the setup of a CHILD_SA. While IP compression involves an extra header in each packet and a compression parameter index (CPI), the virtual "compression association" has no life outside the ESP or AH SA that contains it. Compression associations disappear when the corresponding ESP or AH SA goes away. It is not explicitly mentioned in any DELETE payload. | | Not support | | Explanation |
| 37 | 2046 | Negotiation of IP compression is separate from the negotiation of cryptographic parameters associated with a CHILD_SA. | | Not support | | Explanation |
| 37 | 2047 | A node requesting a CHILD_SA MAY advertise its support for one or more compression algorithms through one or more Notify payloads of type IPCOMP_SUPPORTED. | MAY | Not support | | Not need to test |
| 37 | 2050 | The response MAY indicate acceptance of a single compression algorithm with a Notify payload of type IPCOMP_SUPPORTED. | MAY | Not support | | Not need to test |
| 37 | 2052 | These payloads MUST NOT occur in messages that do not contain SA payloads. | MUST NOT | Not support | | IPComp is "Not support." |
| 37 | 2055 | Although there has been discussion of allowing multiple compression algorithms to be accepted and to have different compression algorithms available for the two directions of a CHILD_SA, | | Not support | | Explanation |
| 37 | 2058 | implementations of this specification MUST NOT accept an IPComp algorithm that was not proposed, MUST NOT accept more than one, and MUST NOT compress using an algorithm other than one proposed and accepted in the setup of the CHILD_SA. | MUST NOT MUST NOT MUST NOT | Not support | | IPComp is "Not support." |
| 37 | 2063 | A side effect of separating the negotiation of IPComp from cryptographic parameters is that it is not possible to propose multiple cryptographic suites and propose IP compression with some of them but not others. | | Not support | | Explanation |
| 38 | 2078 | 2.23. NAT Traversal | | | | |
| 38 | 2080 | Network Address Translation (NAT) gateways are a controversial subject. This section briefly describes what they are and how they are likely to act on IKE traffic. Many people believe that NATs are evil and that we should not design our protocols so as to make them work better. IKEv2 does specify some unintuitive processing rules in order that NATs are more likely to work. | | Not support | | Explanation |
| 38 | 2087 | NATs exist primarily because of the shortage of IPv4 addresses, though there are other rationales. IP nodes that are "behind" a NAT have IP addresses that are not globally unique, but rather are assigned from some space that is unique within the network behind the NAT but that are likely to be reused by nodes behind other NATs. Generally, nodes behind NATs can communicate with other nodes behind the same NAT and with nodes with globally unique addresses, but not with nodes behind other NATs. There are exceptions to that rule. When those nodes make connections to nodes on the real Internet, the NAT gateway "translates" the IP source address to an address that will be routed back to the gateway. Messages to the gateway from the Internet have their destination addresses "translated" to the internal address that will route the packet to the correct endnode. | | Not support | | Explanation |
| 38 | 2101 | NATs are designed to be "transparent" to endnodes. Neither software on the node behind the NAT nor the node on the Internet requires modification to communicate through the NAT. Achieving this transparency is more difficult with some protocols than with others. Protocols that include IP addresses of the endpoints within the payloads of the packet will fail unless the NAT gateway understands the protocol and modifies the internal references as well as those in the headers. Such knowledge is inherently unreliable, is a network layer violation, and often results in subtle problems. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|----------------------|------------------|--------|--------------------------------|
| 38 | 2111 | Opening an IPsec connection through a NAT introduces special problems. If the connection runs in transport mode, changing the IP addresses on packets will cause the checksums to fail and the NAT cannot correct the checksums because they are cryptographically protected. Even in tunnel mode, there are routing problems because transparently translating the addresses of AH and ESP packets requires special logic in the NAT and that logic is heuristic and unreliable in nature. For that reason, IKEv2 can negotiate UDP encapsulation of IKE and ESP packets. This encoding is slightly less efficient but is easier for NATs to process. In addition, firewalls may be configured to pass IPsec traffic over UDP but not ESP/AH or vice versa. | | Not support | | Explanation |
| 39 | 2134 | It is a common practice of NATs to translate TCP and UDP port numbers as well as addresses and use the port numbers of inbound packets to decide which internal node should get a given packet. For this reason, even though IKE packets MUST be sent from and to UDP port 500, they MUST be accepted coming from any port and responses MUST be sent to the port from whence they came. This is because the ports may be modified as the packets pass through NATs. Similarly, IP addresses of the IKE endpoints are generally not included in the IKE payloads because the payloads are cryptographically protected and could not be transparently modified by NATs. | MUST MUST MUST | Not support | | NAT Traversal in "Not support" |
| 39 | 2145 | Port 4500 is reserved for UDP-encapsulated ESP and IKE. When working through a NAT, it is generally better to pass IKE packets over port 4500 because some older NATs handle IKE traffic on port 500 cleverly in an attempt to transparently establish IPsec connections between endpoints that don't handle NAT traversal themselves. Such NATs may interfere with the straightforward NAT traversal envisioned by this document, so an IPsec endpoint that discovers a NAT between it and its correspondent MUST send all subsequent traffic to and from port 4500, which NATs should not treat specially (as they might with port 500). | MUST | Not support | | NAT Traversal in "Not support" |
| 39 | 2156 | The specific requirements for supporting NAT traversal [RFC3715] are listed below. Support for NAT traversal is optional. In this section only, requirements listed as MUST apply only to implementations supporting NAT traversal. | MUST | Not support | | NAT Traversal in "Not support" |
| 39 | 2161 | IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived. | MUST | Not support | | NAT Traversal in "Not support" |
| 39 | 2164 | Both IKE initiator and responder MUST include in their IKE_SA_INIT packets Notify payloads of type NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP. Those payloads can be used to detect if there is NAT between the hosts, and which end is behind the NAT. The location of the payloads in the IKE_SA_INIT packets are just after the Ni and Nr payloads (before the optional CERTREQ payload). | | Not support | | Explanation |
| 39 | 2172 | If none of the NAT_DETECTION_SOURCE_IP payload(s) received matches the hash of the source IP and port found from the IP header of the packet containing the payload, it means that the other end is behind NAT (i.e., someone along the route changed the source address of the original packet to match the address of the NAT box). In this case, this end should allow dynamic update of the other ends IP address, as described later. | | Not support | | Explanation |
| 40 | 2190 | If the NAT_DETECTION_DESTINATION_IP payload received does not match the hash of the destination IP and port found from the IP header of the packet containing the payload, it means that this end is behind a NAT. In this case, this end SHOULD start sending keepalive packets as explained in [Hutt05]. | SHOULD | Not support | | NAT Traversal in "Not support" |
| 40 | 2196 | The IKE initiator MUST check these payloads if present and if they do not match the addresses in the outer packet MUST tunnel all future IKE and ESP packets associated with this IKE_SA over UDP port 4500. | MUST MUST | Not support | | NAT Traversal in "Not support" |
| 40 | 2201 | To tunnel IKE packets over UDP port 4500, the IKE header has four octets of zero prepended and the result immediately follows the UDP header. To tunnel ESP packets over UDP port 4500, the ESP header immediately follows the UDP header. Since the first four bytes of the ESP header contain the SPI, and the SPI cannot validly be zero, it is always possible to distinguish ESP and IKE messages. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-------------------------|------------------|--------|---|
| 40 | 2209 | The original source and destination IP address required for the transport mode TCP and UDP packet checksum fixup (see [Hutt05]) are obtained from the Traffic Selectors associated with the exchange. In the case of NAT traversal, the Traffic Selectors MUST contain exactly one IP address, which is then used as the original IP address. | MUST | Not support | | NAT Traversal in "Not support" |
| 40 | 2216 | There are cases where a NAT box decides to remove mappings that are still alive (for example, the keepalive interval is too long, or the NAT box is rebooted). To recover in these cases, hosts that are not behind a NAT SHOULD send all packets (including retransmission packets) to the IP address and port from the last valid authenticated packet from the other end (i.e., dynamically update the address). A host behind a NAT SHOULD NOT do this because it opens a DoS attack possibility. Any authenticated IKE packet or any authenticated UDP-encapsulated ESP packet can be used to detect that the IP address or the port has changed. | SHOULD SHOULD NOT | Not support | | NAT Traversal in "Not support" |
| 40 | 2227 | Note that similar but probably not identical actions will likely be needed to make IKE work with Mobile IP, but such processing is not addressed by this document. | | Not support | | Explanation |
| 40 | 2231 | 2.24. Explicit Congestion Notification (ECN) | | | | |
| 40 | 2233 | When IPsec tunnels behave as originally specified in [RFC2401], ECN usage is not appropriate for the outer IP headers because tunnel decapsulation processing discards ECN congestion indications to the detriment of the network. ECN support for IPsec tunnels for IKEv1-based IPsec requires multiple operating modes and negotiation (see[RFC3168]). | | Not support | | Explanation |
| 41 | 2246 | IKEv2 simplifies this situation by requiring that ECN be usable in the outer IP headers of all tunnel-mode IPsec SAs created by IKEv2. | | Not support | | Explanation |
| 41 | 2248 | Specifically, tunnel encapsulators and decapsulators for all tunnel-mode SAs created by IKEv2 MUST support the ECN full-functionality option for tunnels specified in [RFC3168] and MUST implement the tunnel encapsulation and decapsulation processing specified in [RFC4301] to prevent discarding of ECN congestion indications. | MUST MUST | Not support | | ECN is "Not support." |
| 41 | 2255 | 3. Header and Payload Formats | | | | |
| 41 | 2257 | 3.1. The IKE Header | | | | |
| 41 | 2259 | IKE messages use UDP ports 500 and/or 4500, with one IKE message per UDP datagram. Information from the beginning of the packet through the UDP header is largely ignored except that the IP addresses and UDP ports from the headers are reversed and used for return packets. | | Not support | | Explanation |
| 41 | 2263 | When sent on UDP port 500, IKE messages begin immediately following the UDP header. | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 41 | 2264 | When sent on UDP port 4500, IKE messages have prepended four octets of zero. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 41 | 2265 | These four octets of zero are not part of the IKE message and are not included in any of the length fields or checksums defined by IKE. Each IKE message begins with the IKE header, denoted HDR in this memo. Following the header are one or more IKE payloads each identified by a "Next Payload" field in the preceding payload. Payloads are processed in the order in which they appear in an IKE message by invoking the appropriate processing routine according to 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. | | Not support | | Explanation |
| 41 | 2275 | If a payload of type "Encrypted" is found, that payload is decrypted and its contents parsed as additional payloads. | | Not support | | Explanation |
| 41 | 2277 | An Encrypted payload MUST be the last payload in a packet | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.R.1.1.1.2 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 SGW.I.1.1.1.2 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.R.1.1.1.2 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 |
| 41 | 2277 | and an Encrypted payload MUST NOT contain another Encrypted payload. | MUST NOT | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.R.1.1.1.2 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 SGW.I.1.1.1.2 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.R.1.1.1.2 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 |
| 41 | 2280 | The Recipient SPI in the header identifies an instance of an IKE security association. It is therefore possible for a single instance of IKE to multiplex distinct sessions with multiple peers. | | Not support | | Explanation |
| 41 | 2284 | All multi-octet fields representing integers are laid out in big endian order (aka most significant byte first, or network byte order). | | Not support | | Explanation |
| 41 | 2288 | The format of the IKE header is shown in Figure 4. | | Not support | | Explanation |
| 42 | 2302 | <pre> 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 +++++ ! IKE_SA Initiator's SPI ! ! +++++ ! IKE_SA Responder's SPI ! ! +++++ ! Next Payload ! MjVer ! MnVer ! Exchange Type ! Flags ! +++++ ! Message ID ! +++++ ! Length ! +++++ </pre> <p>Figure 4: IKE Header Format</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 42 | 2320 | 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. | MUST NOT | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 42 | 2324 | 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 (including repeats of that message including a cookie) | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 42 | 2324 | and MUST NOT be zero in any other message. | MUST NOT | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|---------------------------------|---|
| 42 | 2330 | 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. | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 42 | 2334 | 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. | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 42 | 2334 | Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. | MUST | Not support | | Explanation |
| 42 | 2334 | Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.4.2 SGW.R.1.1.4.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 42 | 2342 | 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. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 42 | 2342 | They MUST ignore the minor version number of received messages. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.4.1 SGW.R.1.1.4.1 |
| 42 | 2347 | o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 43 | 2358 | Exchange Type Value | | Not support | | Explanation |
| 43 | 2360 | RESERVED 0-33 | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|---|
| 43 | 2361 | IKE_SA_INIT 34 IKE_AUTH 35 CREATE_CHILD_SA 36 INFORMATIONAL 37 | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 43 | 2365 | RESERVED TO IANA 38-239 Reserved for private use 240-255 | | Not support | | Explanation |
| 43 | 2368 | o Flags (1 octet) - Indicates specific options that are set for the message. Presence of options are indicated by the appropriate bit in the flags field being set. The bits are defined LSB first, so bit 0 would be the least significant bit of the Flags octet. In the description below, a bit being 'set' means its value is '1', while 'cleared' means its value is '0'. | | Not support | | Explanation |
| 43 | 2376 | -- X(reserved) (bits 0-2) - These bits MUST be cleared when sending | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 43 | 2376 | and MUST be ignored on receipt. | | | | EN.I.1.1.11.1 EN.I.1.1.11.2 EN.I.1.2.7.1 EN.R.1.1.11.1 EN.R.1.1.11.2 EN.R.1.2.9.1 EN.R.1.3.3.1 SGW.I.1.1.11.1 SGW.I.1.1.11.2 SGW.I.1.2.7.1 SGW.R.1.1.11.1 SGW.R.1.1.11.2 SGW.R.1.2.9.1 SGW.R.1.3.3.1 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 43 | 2379 | -- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA | MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 |
| 43 | 2379 | 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. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 43 | 2386 | -- V(ersion) (bit 4 of Flags) - 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. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.3 EN.R.1.1.11.3 SGW.I.1.1.11.3 SGW.R.1.1.11.3 |
| 43 | 2393 | -- R(esponse) (bit 5 of Flags) - 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 | MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 |
| 43 | 2393 | and MUST be set in all responses. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 43 | 2393 | An IKE endpoint MUST NOT generate a response to a message that is marked as being a response. | MUST NOT | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 43 | 2400 | --- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 43 | 2400 | and MUST be ignored on receipt. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.1 EN.I.1.1.11.2 EN.I.1.2.7.1 EN.R.1.1.11.1 EN.R.1.1.11.2 EN.R.1.2.9.1 EN.R.1.3.3.1 SGW.I.1.1.11.1 SGW.I.1.1.11.2 SGW.I.1.2.7.1 SGW.R.1.1.11.1 SGW.R.1.1.11.2 SGW.R.1.2.9.1 SGW.R.1.3.3.1 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 44 | 2414 | o Message ID (4 octets) - 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. See sections 2.1 and 2.2. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 44 | 2420 | o Length (4 octets) - Length of total message (header + payloads) in octets. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 44 | 2423 | 3.2. Generic Payload Header | | | | |
| 44 | 2425 | Each IKE payload defined in sections 3.3 through 3.16 begins with a generic payload header, shown in Figure 5. Figures for each payload below will include the generic payload header, but for brevity the description of each field will be omitted. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 44 | 2456 | No Next Payload 0 | | Not support | | Explanation |
| 44 | 2458 | RESERVED 1-32 | | Not support | | Explanation |
| 44 | 2459 | 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 Vendor ID V 43 Traffic Selector - Initiator TSi 44 Traffic Selector - Responder TSr 45 Encrypted E 46 Configuration CP 47 Extensible Authentication EAP 48 | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 45 | 2483 | RESERVED TO IANA 49-127 PRIVATE USE 128-255 | | Not support | | Explanation |
| 45 | 2486 | Payload type values 1-32 should not be used so that there is no overlap with the code assignments for IKEv1. Payload type values 49-127 are reserved to IANA for future assignment in IKEv2 (see section 6). Payload type values 128-255 are for private use among mutually consenting parties. | | Not support | | Explanation |
| 45 | 2492 | 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 MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 |
| 45 | 2492 | MUST be ignored by the recipient if the recipient understands the payload type code. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.4.4 SGW.R.1.1.4.4 |
| 45 | 2492 | MUST be set to zero for payload types defined in this document. | MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 45 | 2492 | Note that the critical bit applies to the current payload rather than the "next" payload whose type code appears in the first octet. The reasoning behind not setting the critical bit for payloads defined in this document is that all implementations MUST understand all payload types defined in this document and therefore must ignore the Critical bit's value. Skipped payloads are expected to have valid Next Payload and Payload Length fields. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGWI.1.1.1.1 SGWI.1.1.1.2 SGWI.1.1.1.3 SGWI.1.2.1.1 SGWI.2.1.1.1 SGWI.2.1.1.2 SGWR.1.1.1.1 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 45 | 2508 | o RESERVED (7 bits) - MUST be sent as zero: | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGWI.1.1.1.1 SGWI.1.1.1.2 SGWI.1.1.1.3 SGWI.1.2.1.1 SGWI.2.1.1.1 SGWI.2.1.1.2 SGWR.1.1.1.1 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 45 | 2508 | MUST be ignored on receipt. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.1 EN.I.1.1.11.2 EN.I.1.2.7.1 EN.R.1.1.11.1 EN.R.1.1.11.2 EN.R.1.2.9.1 EN.R.1.3.3.1 SGWI.1.1.11.1 SGWI.1.1.11.2 SGWI.1.2.7.1 SGWR.1.1.11.1 SGWR.1.1.11.2 SGWR.1.2.9.1 SGWR.1.3.3.1 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|---------------------------|------------------|---------------------------------|---|
| 45 | 2511 | o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header. | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 46 | 2526 | 3.3. Security Association Payload | | | | |
| 46 | 2528 | The Security Association Payload, denoted SA in this memo, is used to negotiate attributes of a security association. Assembly of Security Association Payloads requires great peace of mind. | | Not support | | Explanation |
| 46 | 2530 | An SA payload MAY contain multiple proposals. If there is more than one, they MUST be ordered from most preferred to least preferred. Each proposal may contain multiple IPsec protocols (where a protocol is IKE, ESP, or AH), each protocol MAY contain multiple transforms, and each transform MAY contain multiple attributes. | MAY MUST MAY MAY | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.1.6.4 EN.I.1.1.6.6 EN.I.1.2.3.4 EN.I.1.2.3.5 EN.I.1.2.4.4 EN.I.1.2.4.5 SGW.I.1.1.6.4 SGW.I.1.1.6.6 SGW.I.1.2.3.4 SGW.I.1.2.3.5 SGW.I.1.2.4.4 SGW.I.1.2.4.5 |
| 46 | 2535 | When parsing an SA, an implementation MUST check that the total Payload Length is consistent with the payload's internal lengths and counts. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.6.4 EN.R.1.1.6.6 EN.R.1.2.5.4 EN.R.1.2.6.5 EN.R.1.2.6.6 SGW.R.1.1.6.4 SGW.R.1.1.6.6 SGW.R.1.2.5.4 SGW.R.1.2.6.5 SGW.R.1.2.6.6 |
| 46 | 2537 | 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. | | Not support | | Explanation |
| 46 | 2545 | The syntax of Security Associations, Proposals, Transforms, and Attributes is based on ISAKMP; however, the semantics are somewhat different. The reason for the complexity and the hierarchy is to allow for multiple possible combinations of algorithms to be encoded in a single SA. Sometimes there is a choice of multiple algorithms, whereas other times there is a combination of algorithms. For example, an initiator might want to propose using (AH w/MD5 and ESP w/3DES) OR (ESP w/MD5 and 3DES). | | Not support | | Explanation |
| 46 | 2554 | One of the reasons the semantics of the SA payload has changed from ISAKMP and IKEv1 is to make the encodings more compact in common cases. | | Not support | | Explanation |
| 46 | 2558 | The Proposal structure contains within it a Proposal # and an IPsec protocol ID. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|------------------|------------------|--|--|
| 46 | 2559 | Each structure MUST have the same Proposal # as the previous one or be one (1) greater. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.4 EN.I.1.1.6.6 EN.I.1.2.3.5 EN.I.1.2.4.4 EN.R.1.1.6.4 EN.R.1.1.6.6 EN.R.1.2.5.4 EN.R.1.2.6.6 SGW.I.1.1.6.4 SGW.I.1.1.6.6 SGW.I.1.2.3.5 SGW.I.1.2.4.4 SGW.R.1.1.6.4 SGW.R.1.1.6.6 SGW.R.1.2.5.4 SGW.R.1.2.6.6 |
| 46 | 2560 | The first Proposal MUST have a Proposal # of one (1). | MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 |
| 46 | 2561 | If two successive structures have the same Proposal number, it means that the proposal consists of the first structure AND the second. So a proposal of AH AND ESP would have two proposal structures, one for AH and one for ESP and both would have Proposal #1. A proposal of AH OR ESP would have two proposal structures, one for AH with Proposal #1 and one for ESP with Proposal #2. | | Not support | | Explanation |
| 46 | 2569 | 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 a single transform: an integrity check algorithm. ESP generally has two: an encryption algorithm and an integrity check algorithm. IKE generally has four transforms: a Diffie-Hellman group, an integrity check algorithm, a prf algorithm, and an encryption algorithm. | | Not support | | Explanation |
| 47 | 2583 | If an algorithm that combines encryption and integrity protection is proposed, it MUST be proposed as an encryption algorithm and an integrity protection algorithm MUST NOT be proposed. | MUST MUST NOT | Not support | | Combined encryption and integrity protection algorithms are out of scope |
| 47 | 2586 | For each Protocol, the set of permissible transforms is assigned transform ID numbers, which appear in the header of each transform. | | Not support | | Explanation |
| 47 | 2590 | 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. | | Not support | | Explanation |
| 47 | 2590 | For example, to propose ESP with (3DES or IDEA) and (HMAC_MD5 or HMAC_SHA), the ESP proposal would contain two Transform Type 1 candidates (one for 3DES and one for IDEA) and two Transform Type 2 candidates (one for HMAC_MD5 and one for HMAC_SHA). This effectively proposes four combinations of algorithms. If the initiator wanted to propose only a subset of those, for example (3DES and HMAC_MD5) or (IDEA and HMAC_SHA), there is no way to encode that as multiple transforms within a single Proposal. Instead, the initiator would have to construct two different Proposals, each with two transforms. | | Not support | | Explanation |
| 47 | 2604 | 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. Most transforms do not have attributes. | MAY | Not support | | Not need to test |
| 47 | 2608 | A transform MUST NOT have multiple attributes of the same type. | MUST NOT | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.6.1 SGW.I.1.1.6.1 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 47 | 2609 | To propose alternate values for an attribute (for example, multiple key sizes for the AES encryption algorithm), and implementation MUST include multiple Transforms with the same Transform Type each with a single Attribute. | MUST | Not support | | Only AES with 128bit key is "BASIC". AES with other length keys are out of scope. |
| 47 | 2614 | Note that the semantics of Transforms and Attributes are quite different from those in IKEv1. In IKEv1, a single Transform carried multiple algorithms for a protocol with one carried in the Transform and the others carried in the Attributes. | | Not support | | Explanation |
| 47 | 2619 | <pre> 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 ! +++++ ! ~ <Proposals> ! +++++ </pre> <p>Figure 6: Security Association Payload</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 48 | 2638 | o Proposals (variable) - One or more proposal substructures. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 48 | 2640 | The payload type for the Security Association Payload is thirty three (33). | | Not support | | Explanation |
| 48 | 2643 | 3.3.1. Proposal Substructure | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 48 | 2645 | <pre> 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 +++++ ! 0 (last) or 2! RESERVED ! Proposal Length ! +++++ ! Proposal # ! Protocol ID ! SPI Size !# of Transforms! +++++ ~ ~ SPI (variable) ~ ~ ~ <Transforms> ~ ~ +++++ </pre> <p>Figure 7: Proposal Substructure</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 48 | 2661 | o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. | | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.6.4 EN.I.1.1.6.6 EN.I.1.2.3.5 EN.I.1.2.4.4 SGW.I.1.1.6.4 SGW.I.1.1.6.6 SGW.I.1.2.3.5 SGW.I.1.2.4.4 |
| 48 | 2662 | This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. The value (2) corresponds to a Payload Type of Proposal in IKEv1, and the first 4 octets of the Proposal structure are designed to look somewhat like the header of a Payload. | | Not support | | Explanation |
| 48 | 2669 | o RESERVED (1 octet) - MUST be sent as zero; | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 48 | 2669 | MUST be ignored on receipt. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.1 EN.I.1.1.11.2 EN.I.1.2.7.1 EN.R.1.1.11.1 EN.R.1.1.11.2 EN.R.1.2.9.1 EN.R.1.3.3.1 SGW.I.1.1.11.1 SGW.I.1.1.11.2 SGW.I.1.2.7.1 SGW.R.1.1.11.1 SGW.R.1.1.11.2 SGW.R.1.2.9.1 SGW.R.1.3.3.1 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 48 | 2672 | o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 48 | 2675 | o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 48 | 2675 | and subsequent proposals MUST either be the same as the previous proposal (indicating an AND of the two proposals) or one more than the previous proposal (indicating an OR of the two proposals). | MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.6.4 EN.I.1.1.6.6 EN.I.1.2.3.5 EN.I.1.2.4.4 SGW.I.1.1.6.4 SGW.I.1.1.6.6 SGW.I.1.2.3.5 SGW.I.1.2.4.4 |
| 48 | 2679 | When a proposal is accepted, all of the proposal numbers in the SA payload MUST be the same and MUST match the number on the proposal sent that was accepted. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.6.4 EN.R.1.1.6.6 EN.R.1.2.5.4 EN.R.1.2.6.6 SGW.R.1.1.6.4 SGW.R.1.1.6.6 SGW.R.1.2.5.4 SGW.R.1.2.6.6 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 49 | 2694 | o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are: | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 49 | 2697 | Protocol Protocol ID | | Not support | | Explanation |
| 49 | 3698 | RESERVED 0 | | Not support | | Explanation |
| 49 | 2699 | IKE 1 AH 2 ESP 3 | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 49 | 2702 | RESERVED TO IANA 4-200 PRIVATE USE 201-255 | | Not support | | Explanation |
| 49 | 2705 | o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero: | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 SGW.I.1.1.1.1 SGW.R.1.1.1.1 |
| 49 | 2706 | 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). | | | | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 49 | 2711 | o # of Transforms (1 octet) - Specifies the number of transforms in this proposal. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.3 EN.I.1.1.6.6 EN.R.1.1.6.4 EN.R.1.1.6.6 SGW.I.1.1.6.3 SGW.I.1.1.6.6 SGW.R.1.1.6.4 SGW.R.1.1.6.6 |
| 49 | 2714 | o SPI (variable) - The sending entity's SPI. Even if the SPI Size is not a multiple of 4 octets, there is no padding applied to the payload. When the SPI Size field is zero, this field is not present in the Security Association payload. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 49 | 2719 | o Transforms (variable) - One or more transform substructures. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.1 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 49 | 2721 | 3.3.2. Transform Substructure | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 49 | 2723 | <pre> 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 +++++ ! 0 (last) or 3! RESERVED ! Transform Length ! +++++ !Transform Type! RESERVED ! Transform ID ! +++++ ! ~ Transform Attributes ~ ! +++++ </pre> <p>Figure 8: Transform Substructure</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 49 | 2737 | o 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 49 | 2738 | This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. | | Not support | | Explanation |
| 50 | 2750 | The value (3) corresponds to a Payload Type of Transform in IKEv1, and the first 4 octets of the Transform structure are designed to look somewhat like the header of a Payload. | | Not support | | Explanation |
| 50 | 2754 | o RESERVED - MUST be sent as zero; | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 50 | 2754 | MUST be ignored on receipt. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.1 EN.I.1.1.11.2 EN.I.1.2.7.1 EN.R.1.1.11.1 EN.R.1.1.11.2 EN.R.1.2.9.1 EN.R.1.3.3.1 SGW.I.1.1.11.1 SGW.I.1.1.11.2 SGW.I.1.2.7.1 SGWR.1.1.11.1 SGWR.1.1.11.2 SGWR.1.2.9.1 SGWR.1.3.3.1 |
| 50 | 2756 | o Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGWR.1.1.6.1 SGWR.1.1.6.2 |
| 50 | 2759 | 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. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGWR.1.1.6.1 SGWR.1.1.6.2 |
| 50 | 2759 | 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. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.1 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|--|
| 50 | 2759 | 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. | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 50 | 2769 | o Transform ID (2 octets) - The specific instance of the transform type being proposed. | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 50 | 2772 | Transform Type Values | | Not support | | Explanation |
| 50 | 2774 | Transform Type Used In | | Not support | | Explanation |
| 50 | 2776 | RESERVED 0 | | Not support | | Explanation |
| 50 | 2777 | Encryption Algorithm (ENCR) 1 (IKE and ESP) Pseudo-random 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) | | | | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|------------------|------------------|-------------|-----------------------|
| 50 | 2782 | RESERVED TO IANA PRIVATE USE | 6-240 241-255 | | Not support | Explanation |
| 50 | 2785 | For Transform Type 1 (Encryption Algorithm), defined Transform IDs are: | | | Not support | Explanation |
| 50 | 2788 | Name | Number | Defined In | | Explanation |
| 50 | 2789 | RESERVED | 0 | | Not support | Explanation |
| 50 | 2790 | ENCR_DES_IV64 | 1 | (RFC1827) | | Explanation |
| 50 | 2791 | ENCR_DES [DES] | 2 | (RFC2405), | | Explanation |
| 50 | 2792 | ENCR_3DES | 3 | (RFC2451) | | Explanation |
| 50 | 2793 | ENCR_RC5 | 4 | (RFC2451) | | Explanation |
| 50 | 2794 | ENCR_IDEA [IDEA] | 5 | (RFC2451), | | Explanation |
| 50 | 2795 | ENCR_CAST | 6 | (RFC2451) | | Explanation |
| 50 | 2796 | ENCR_BLOWFISH | 7 | (RFC2451) | | Explanation |
| 50 | 2806 | ENCR_3IDEA | 8 | (RFC2451) | | Explanation |
| 51 | 2807 | ENCR_DES_IV32 | 9 | | | Explanation |
| 51 | 2808 | RESERVED | 10 | | | Explanation |
| 51 | 2809 | ENCR_NULL | 11 | (RFC2410) | | Explanation |
| 51 | 2810 | ENCR_AES_CBC | 12 | (RFC3602) | | Explanation |
| 51 | 2811 | ENCR_AES_CTR | 13 | (RFC3664) | | Explanation |
| 51 | 2812 | values 14-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties. | | | Not support | Explanation |
| 51 | 2815 | For Transform Type 2 (Pseudo-random Function), defined Transform IDs are: | | | Not support | Explanation |
| 51 | 2818 | Name | Number | Defined In | | Explanation |
| 51 | 2819 | RESERVED | 0 | | Not support | Explanation |
| 51 | 2820 | PRF_HMAC_MD5 (RFC2104), [MD5] | 1 | | Not support | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason | |
|------|------|--|-----------------|---|--|--|--|
| 51 | 2821 | PRF_HMAC_SHA1 (RFC2104), [SHA] | 2 | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 51 | 2822 | PRF_HMAC_TIGER (RFC2104) | 3 | | Not support | Explanation | |
| 51 | 2823 | PRF_AES128_XCBC (RFC3664) | 4 | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 51 | 2825 | values 5-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties. | | ADVANCED Only value 5(PRF_HMAC_SHA2_256) defined in RFC 4868 | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.R.1.1.6.1 SGW.I.1.1.6.1 SGW.R.1.1.6.1 | |
| 51 | 2828 | For Transform Type 3 (Integrity Algorithm), defined Transform IDs are: | | Not support | | Explanation | |
| 51 | 2831 | Name Number Defined In | | Not support | | Explanation | |
| 51 | 2832 | NONE 0 | | Not support | | Explanation | |
| 51 | 2833 | AUTH_HMAC_MD5_96 (RFC2403) 1 | | Not support | | Explanation | |
| 51 | 2834 | AUTH_HMAC_SHA1_96 (RFC2404) 2 | | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 51 | 2835 | AUTH_DES_MAC 3 | | Not support | | Explanation | |
| 51 | 2836 | AUTH_KPDK_MD5 (RFC1826) 4 | | Not support | | Explanation | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|--|--|--|
| 51 | 2837 | AUTH_AES_XCBC_96 (RFC3566) 5 | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 51 | 2839 | values 6-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties. | | ADVANCED *Only value 12(AUTH_H MAC_SHA2_256_128) defined in RFC4868 | | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 51 | 2842 | For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are: | | Not support | | Explanation |
| 51 | 2845 | Name Number | | Not support | | Explanation |
| 51 | 2846 | NONE 0 | | Not support | | Explanation |
| 51 | 2847 | Defined in Appendix B 1 - 2 | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 51 | 2848 | RESERVED 3 - 4 | | Not support | | Explanation |
| 51 | 2849 | Defined in [ADDGROUP] 5 | | Not support | | Explanation |
| 51 | 2850 | RESERVED TO IANA 6 - 13 | | Not support | | Explanation |
| 51 | 2851 | Defined in [ADDGROUP] 14 - 18 | | ADVANCED *Only value 14(2048 MODP) defined in RFC 5114 | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|---|--|--|
| 51 | 2852 | RESERVED TO IANA PRIVATE USE 19 - 1023 1024-65535 | | ADVANCED *Only value 24(2048-bit MODP Group with 256-bit Prime Order Subgroup) defined in RFC 5114 | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.R.1.1.6.1 SGW.I.1.1.6.1 SGWR.1.1. EN.I.1.1.6.3 EN.I.1.1.6.4 EN.I.1.1.6.7 EN.I.1.1.6.11 EN.I.1.2.4.4 EN.I.1.2.4.5 EN.R.1.1.6.3 EN.R.1.1.6.4 EN.R.1.1.6.7 EN.R.1.1.6.8 EN.R.1.2.6.5 EN.R.1.2.6.6 SGW.I.1.1.6.3 SGW.I.1.1.6.4 SGW.I.1.1.6.7 SGW.I.1.1.6.11 SGW.I.1.2.4.4 SGW.I.1.2.4.5 SGWR.1.1.6.3 SGWR.1.1.6.4 SGWR.1.1.6.7 SGWR.1.1.6.8 SGWR.1.2.6.5 SGWR.1.2.6.6 |
| 52 | 2862 | For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are: | | Not support | | Explanation |
| 52 | 2865 | Name Number | | Not support | | Explanation |
| 52 | 2866 | No Extended Sequence Numbers 0 | | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.1 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 52 | 2867 | Extended Sequence Numbers 1 | | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 52 | 2868 | RESERVED 2 - 65535 | | Not support | | Explanation |
| 52 | 2870 | 3.3.3. Valid Transform Types by Protocol | | | | |
| 52 | 2872 | 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. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason | | | | | | | | | | | | |
|----------|-----------------------|--|-----------------|------------------|--|---|-----------------------|--|-----|-----------|------------|----|------------|-----|--|-------|--|---|
| 52 | 2872 | A compliant implementation MUST understand all mandatory and optional types for each protocol it supports (though it need not accept proposals with unacceptable suites). | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 | | | | | | | | | | | | |
| 52 | 2872 | A proposal MAY omit the optional types if the only value for them it will accept is NONE. | MAY | Not support | | Not need to test | | | | | | | | | | | | |
| 52 | 2881 | <table border="0"> <tr> <td>Protocol</td> <td>Mandatory Types</td> <td>Optional Types</td> </tr> <tr> <td>IKE</td> <td>ENCR, PRF, INTEG, D-H</td> <td></td> </tr> <tr> <td>ESP</td> <td>ENCR, ESN</td> <td>INTEG, D-H</td> </tr> <tr> <td>AH</td> <td>INTEG, ESN</td> <td>D-H</td> </tr> </table> | Protocol | Mandatory Types | Optional Types | IKE | ENCR, PRF, INTEG, D-H | | ESP | ENCR, ESN | INTEG, D-H | AH | INTEG, ESN | D-H | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| Protocol | Mandatory Types | Optional Types | | | | | | | | | | | | | | | | |
| IKE | ENCR, PRF, INTEG, D-H | | | | | | | | | | | | | | | | | |
| ESP | ENCR, ESN | INTEG, D-H | | | | | | | | | | | | | | | | |
| AH | INTEG, ESN | D-H | | | | | | | | | | | | | | | | |
| 52 | 2886 | 3.3.4. Mandatory Transform IDs | | | | | | | | | | | | | | | | |
| 52 | 2888 | The specification of suites that MUST and SHOULD be supported for interoperability has been removed from this document because they are likely to change more rapidly than this document evolves. | | Not support | | Explanation | | | | | | | | | | | | |
| 52 | 2892 | An important lesson learned from IKEv1 is that no system should only implement the mandatory algorithms and expect them to be the best choice for all customers. | | Not support | | Explanation | | | | | | | | | | | | |
| 52 | 2892 | For example, at the time that this document was written, many IKEv1 implementers were starting to migrate to AES in Cipher Block Chaining (CBC) mode for Virtual Private Network (VPN) applications. Many IPsec systems based on IKEv2 will implement AES, additional Diffie-Hellman groups, and additional hash algorithms, and some IPsec customers already require these algorithms in addition to the ones listed above. | | Not support | | Explanation | | | | | | | | | | | | |
| 52 | 2902 | It is likely that IANA will add additional transforms in the future, and some users may want to use private suites, especially for IKE where implementations should be capable of supporting different parameters, up to certain size limits. | | Not support | | Explanation | | | | | | | | | | | | |
| 52 | 2905 | In support of this goal, all implementations of IKEv2 SHOULD include a management facility that allows specification (by a user or system administrator) of Diffie-Hellman (DH) parameters (the generator, modulus, and exponent lengths and values) for new DH groups. | SHOULD | Not support | | No need to test | | | | | | | | | | | | |
| 53 | 2918 | Implementations SHOULD provide a management interface via which these parameters and the associated transform IDs may be entered (by a user or system administrator), to enable negotiating such groups. | SHOULD | Not support | | No need to test | | | | | | | | | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 53 | 2922 | All implementations of IKEv2 MUST include a management facility that enables a user or system administrator to specify the suites that are acceptable for use with IKE. | MUST | Not support | | No need to test |
| 53 | 2924 | Upon receipt of a payload with a set of transform IDs, the implementation MUST compare the transmitted transform IDs against those locally configured via the management controls, to verify that the proposed suite is acceptable based on local policy. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.2.4.1 SGW.R.1.2.4.1 |
| 53 | 2924 | The implementation MUST reject SA proposals that are not authorized by these IKE suite controls. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.2.4.1 SGW.R.1.2.4.1 |
| 53 | 2924 | Note that cryptographic suites that MUST be implemented need not be configured as acceptable to local policy. | MUST | Not support | | Internal process |
| 53 | 2933 | 3.3.5. Transform Attributes | | | | |
| 53 | 2935 | Each transform in a Security Association payload may include attributes that modify or complete the specification of the transform. These attributes are type/value pairs and are defined below. | | Not support | | Explanation |
| 53 | 2935 | For example, if an encryption algorithm has a variable-length key, the key length to be used may be specified as an attribute. Attributes can have a value with a fixed two octet length or a variable-length value. For the latter, the attribute is encoded as type/length/value. | | Not support | | Explanation |
| 53 | 2944 | <pre> 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 +++++ !A! Attribute Type ! AF=0 Attribute Length ! !F! ! AF=1 Attribute Value ! +++++ ! AF=0 Attribute Value ! ! AF=1 Not Transmitted ! +++++ </pre> <p>Figure 9: Data Attributes</p> | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 53 | 2956 | o Attribute Type (2 octets) - Unique identifier for each type of attribute (see below). | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 53 | 2959 | The most significant bit of this field is the Attribute Format bit (AF). It indicates whether the data attributes follow the Type/Length/Value (TLV) format or a shortened Type/Value (TV) format. If the AF bit is zero (0), then the Data Attributes are of the Type/Length/Value (TLV) form. If the AF bit is a one (1), then the Data Attributes are of the Type/Value form. | | Not support | | Explanation |
| 54 | 2974 | o Attribute Length (2 octets) - Length in octets of the Attribute Value. When the AF bit is a one (1), the Attribute Value is only 2 octets and the Attribute Length field is not present. | | Not support | | Explanation |
| 54 | 2978 | o Attribute Value (variable length) - Value of the Attribute associated with the Attribute Type. If the AF bit is a zero (0), this field has a variable length defined by the Attribute Length field. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 54 | 2978 | If the AF bit is a one (1), the Attribute Value has a length of 2 octets. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 54 | 2984 | Note that only a single attribute type (Key Length) is defined, and it is fixed length. The variable-length encoding specification is included only for future extensions. The only algorithms defined in this document that accept attributes are the AES-based encryption, integrity, and pseudo-random functions, which require a single attribute specifying key width. | | Not support | | Explanation |
| 54 | 2991 | Attributes described as basic MUST NOT be encoded using the variable-length encoding. | MUST NOT | Not support | | Internal process |
| 54 | 2991 | Variable-length attributes MUST NOT be encoded as basic even if their value can fit into two octets. | MUST NOT | Not support | | Internal process |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 54 | 2991 | NOTE: This is a change from IKEv1, where increased flexibility may have simplified the composer of messages but certainly complicated the parser. | | Not support | | Explanation |
| 54 | 2998 | Attribute Type Value Attribute Format ----- | | Not support | | Explanation |
| 54 | 3000 | RESERVED 0-13 | | Not support | | Explanation |
| 54 | 3001 | Key Length (in bits) 14 TV | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 54 | 3002 | RESERVED 15-17 | | Not support | | Explanation |
| 54 | 3003 | RESERVED TO IANA 18-16383 PRIVATE USE 16384-32767 | | Not support | | Explanation |
| 54 | 3005 | Values 0-13 and 15-17 were used in a similar context in IKEv1 and should not be assigned except to matching values. Values 18-16383 are reserved to IANA. Values 16384-32767 are for private use among mutually consenting parties. | | Not support | | Explanation |
| 54 | 3010 | - Key Length When using an Encryption Algorithm that has a variable-length key, this attribute specifies the key length in bits (MUST use network byte order). | MUST | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.1 EN.I.1.1.6.2 EN.R.1.1.6.1 EN.R.1.1.6.2 SGW.I.1.1.6.1 SGW.I.1.1.6.2 SGW.R.1.1.6.1 SGW.R.1.1.6.2 |
| 54 | 3010 | This attribute MUST NOT be used when the specified Encryption Algorithm uses a fixed-length key. | MUST NOT | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 55 | 3030 | 3.3.6. Attribute Negotiation | | | | |
| 55 | 3032 | During security association negotiation, initiators present offers to responders. | | Not support | | Explanation |
| 55 | 3033 | Responders MUST select a single complete set of parameters from the offers (or reject all offers if none are acceptable). | MUST | BASIC | responder | EN.R.1.1.6.3 EN.R.1.1.6.4 EN.R.1.1.6.5 EN.R.1.1.6.6 EN.R.1.2.5.3 EN.R.1.2.5.4 SGW.R.1.1.6.3 SGW.R.1.1.6.4 SGW.R.1.1.6.5 SGW.R.1.1.6.6 SGW.R.1.2.5.3 SGW.R.1.2.5.4 |
| 55 | 3035 | If there are multiple proposals, the responder MUST choose a single proposal number and return all of the Proposal substructures with that Proposal number. | MUST | BASIC | responder | EN.R.1.1.6.4 EN.R.1.1.6.6 SGW.R.1.1.6.4 SGW.R.1.1.6.6 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|---|--|--|
| 55 | 3037 | If there are multiple Transforms with the same type, the responder MUST choose a single one. | MUST | BASIC | responder | EN.R.1.1.6.3 EN.R.1.1.6.5 SGW.R.1.1.6.3 SGW.R.1.1.6.5 |
| 55 | 3039 | Any attributes of a selected transform MUST be returned unmodified. | MUST | BASIC | responder | EN.R.1.1.6.3 EN.R.1.1.6.5 SGW.R.1.1.6.3 SGW.R.1.1.6.5 |
| 55 | 3040 | The initiator of an exchange MUST check that the accepted offer is consistent with one of its proposals, and if not that response MUST be rejected. | MUST MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.6.9 EN.I.1.1.6.10 SGW.I.1.1.6.9 SGW.I.1.1.6.10 |
| 55 | 3044 | Negotiating Diffie-Hellman groups presents some special challenges. SA offers include proposed attributes and a Diffie-Hellman public number (KE) in the same message. | | Not support | | Explanation |
| 55 | 3046 | If in the initial exchange the initiator offers to use one of several Diffie-Hellman groups, it SHOULD pick the one the responder is most likely to accept and include a KE corresponding to that group. | SHOULD | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.1.1 SGW.I.1.1.1.1 |
| 55 | 3049 | If the guess turns out to be wrong, the responder will indicate the correct group in the response and the initiator SHOULD pick an element of that group for its KE value when retrying the first message. | SHOULD | ADVANCED *Because DH#14 is ADVANCED group. | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.7 EN.R.1.1.6.7 SGW.I.1.1.6.7 SGW.R.1.1.6.7 |
| 55 | 3052 | It SHOULD , however, continue to propose its full supported set of groups in order to prevent a man-in-the-middle downgrade attack. | SHOULD | Not support | | difficult to test |
| 55 | 3056 | Implementation Note: | | Not support | | Explanation |
| 55 | 3058 | Certain negotiable attributes can have ranges or could have multiple acceptable values. These include the key length of a variable key length symmetric cipher. To further interoperability and to support upgrading endpoints independently, implementers of this protocol SHOULD accept values that they deem to supply greater security. | SHOULD | Not support | | Internal process |
| 55 | 3058 | For instance, if a peer is configured to accept a variable-length cipher with a key length of X bits and is offered that cipher with a larger key length, the implementation SHOULD accept the offer if it supports use of the longer key. | SHOULD | Not support | | Internal process |
| 55 | 3068 | Support of this capability allows an implementation to express a concept of "at least" a certain level of security -- "a key length of at least X bits for cipher Y". | | Not support | | Explanation |
| 56 | 3086 | 3.4. Key Exchange Payload | | | | |
| 56 | 3088 | The Key Exchange Payload, denoted KE in this memo, is used to exchange Diffie-Hellman public numbers as part of a Diffie-Hellman key exchange. The Key Exchange Payload consists of the IKE generic payload header followed by the Diffie-Hellman public value itself. | | Not support | | Explanation |
| 56 | 3093 | <pre> 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 ! +++++ ! DH Group # ! RESERVED ! +++++ ! ~ Key Exchange Data ~ ! +++++ </pre> <p>Figure 10: Key Exchange Payload Format</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 SGW.I.1.1.1.1 SGW.R.1.1.1.1 |
| 56 | 3107 | A key exchange payload is constructed by copying one's Diffie-Hellman public value into the "Key Exchange Data" portion of the payload. | | Not support | | Explanation |
| 56 | 3109 | The length of the Diffie-Hellman public value MUST be equal to the length of the prime modulus over which the exponentiation was performed, prepending zero bits to the value if necessary. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 SGW.I.1.1.1.1 SGW.R.1.1.1.1 |
| 56 | 3113 | The DH Group # identifies the Diffie-Hellman group in which the Key Exchange Data was computed (see section 3.3.2). | | Not support | | Explanation |
| 56 | 3114 | If the selected proposal uses a different Diffie-Hellman group, the message MUST be rejected with a Notify payload of type INVALID_KE_PAYLOAD. | MUST | ADVANCED *Because DH#14 is ADVANCED group. | EN(responder) SGW(responder) | EN.R.1.1.6.7 SGW.R.1.1.6.7 |
| 56 | 3118 | The payload type for the Key Exchange payload is thirty four (34). | | Not support | | |
| 56 | 3120 | 3.5. Identification Payloads | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 56 | 3122 | The Identification Payloads, denoted IDi and IDr in this memo, allow peers to assert an identity to one another. This identity may be used for policy lookup, but does not necessarily have to match anything in the CERT payload; both fields may be used by an implementation to perform access control decisions. | | Not support | | Explanation |
| 56 | 3128 | NOTE: In IKEv1, two ID payloads were used in each direction to hold Traffic Selector (TS) information for data passing over the SA. In IKEv2, this information is carried in TS payloads (see section 3.13). | | Not support | | Explanation |
| 57 | 3142 | The Identification Payload consists of the IKE generic payload header followed by identification fields as follows: | | Not support | | Explanation |
| 57 | 3145 | <pre> 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 ! +++++ ! ID Type ! RESERVED +++++ ! ~ Identification Data ~ ! +++++ </pre> <p>Figure 11: Identification Payload Format</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 57 | 3159 | o ID Type (1 octet) - Specifies the type of Identification being used. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 57 | 3162 | o RESERVED - MUST be sent as zero; | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 57 | 3162 | MUST be ignored on receipt. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.2 EN.R.1.1.11.2 SGW.I.1.1.11.2 SGW.R.1.1.11.2 |
| 57 | 3164 | 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. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.R.1.1.1.2 SGW.I.1.1.1.2 SGW.R.1.1.1.2 |
| 57 | 3168 | The payload types for the Identification Payload are thirty five (35) for IDi | | Not support | | Explanation |
| 57 | 3168 | and thirty six (36) for IDr. | | Not support | | Explanation |
| 57 | 3171 | The following table lists the assigned values for the Identification Type field, followed by a description of the Identification Data which follows: | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|---|--|--|
| 57 | 3175 | ID Type ----- Value ----- | | Not support | | Explanation |
| 57 | 3177 | RESERVED 0 | | Not support | | Explanation |
| 57 | 3179 | ID_IPV4_ADDR 1 A single four (4) octet IPv4 address. | | Not support | | Not support except ID_IPV6_ADDR, FQDN and RFC822_ADDR, |
| 57 | 3183 | ID_FQDN 2 A fully-qualified domain name string. An example of a ID_FQDN is, "example.com". The string MUST not contain any terminators (e.g., NULL, CR, etc.). | | ADVANCED *Only with RSA-DSS auth. | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 EN.R.1.2.8.1 SGW.I.1.1.10.3 SGW.R.1.1.10.3 SGW.R.1.2.8.1 |
| 58 | 3198 | ID_RFC822_ADDR 3 A fully-qualified RFC822 email address string, An example of a ID_RFC822_ADDR is, "jsmith@example.com". The string MUST not contain any terminators. | | ADVANCED *Only with RSA-DSS auth. | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 EN.R.1.2.8.1 SGW.I.1.1.10.3 SGW.R.1.1.10.3 SGW.R.1.2.8.1 |
| 58 | 3204 | Reserved to IANA 4 | | Not support | | Not support except ID_IPV6_ADDR, FQDN and RFC822_ADDR, |
| 58 | 3206 | ID_IPV6_ADDR 5 A single sixteen (16) octet IPv6 address. | | BASIC (receiving) | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.2.1.1.1 EN.R.1.1.1.2 EN.R.2.1.1.1 SGW.I.1.1.1.2 SGW.I.2.1.1.1 SGW.R.1.1.1.2 SGW.R.2.1.1.1 |
| | | | | BASIC (sending)*1 | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.2.1.1.1 EN.R.1.1.1.2 EN.R.2.1.1.1 SGW.I.1.1.1.2 SGW.I.2.1.1.1 SGW.R.1.1.1.2 SGW.R.2.1.1.1 |
| 58 | 3210 | Reserved to IANA 6 - 8 | | Not support | | Explanation |
| 58 | 3212 | ID_DER_ASN1_DN 9 The binary Distinguished Encoding Rules (DER) encoding of an ASN.1 X.500 Distinguished Name [X.501]. | | Not support | | Not support except ID_IPV6_ADDR, FQDN and RFC822_ADDR, |
| 58 | 3217 | ID_DER_ASN1_GN 10 The binary DER encoding of an ASN.1 X.500 GeneralName[X.509]. | | Not support | | Not support except ID_IPV6_ADDR, FQDN and RFC822_ADDR, |
| 58 | 3222 | ID_KEY_ID 11 An opaque octet stream which may be used to pass vendor-specific information necessary to do certain proprietary types of identification. | | Not support | | Not support except ID_IPV6_ADDR, FQDN and RFC822_ADDR, |
| 58 | 3228 | Reserved to IANA 12-200 | | Not support | | Explanation |
| 58 | 3230 | Reserved for private use 201-255 | | Not support | | Explanation |
| 58 | 3232 | Two implementations will interoperate only if each can generate a type of ID acceptable to the other. | | Not support | | Explanation |
| 58 | 3233 | To assure maximum interoperability, implementations MUST be configurable to send at least one of ID_IPV4_ADDR, ID_FQDN, ID_RFC822_ADDR, or ID_KEY_ID, | MUST | Not support *However ID_FQDN and RFC822_ADDR are available only with RSA-DSS auth. | | Not support except ID_IPV6_ADDR, FQDN and RFC822_ADDR, |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason | |
|------|------|--|-----------------|---|--|--|--|
| 58 | 3235 | and MUST be configurable to accept all of these types. | MUST | Not support *However ID_FQDN and RFC822_ADDR are available only with RSA-DSS auth. | | Not support except ID_IPV6_ADDR, FQDN and RFC822_ADDR, | |
| 58 | 3236 | Implementations SHOULD be capable of generating and accepting all of these types. | SHOULD | Not support *However ID_FQDN and RFC822_ADDR are available only with RSA-DSS auth. | | Not support except ID_IPV6_ADDR, FQDN and RFC822_ADDR, | |
| 58 | 3238 | IPv6-capable implementations MUST additionally be configurable to accept ID_IPV6_ADDR. IPv6-only implementations MAY be configurable to send only ID_IPV6_ADDR. | MUST MAY | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.2.1.1.1 EN.R.1.1.1.2 EN.R.2.1.1.1 SGW.I.1.1.1.2 SGW.I.2.1.1.1 SGW.R.1.1.1.2 SGW.R.2.1.1.1 | |
| 59 | 3254 | 3.6. Certificate Payload | | | | | |
| 59 | 3256 | The Certificate Payload, denoted CERT in this memo, provides a means to transport certificates or other authentication-related information via IKE. | | Not support | | Explanation | |
| 59 | 3256 | Certificate payloads SHOULD be included in an exchange if certificates are available to the sender unless the peer has indicated an ability to retrieve this information from elsewhere using an HTTP_CERT_LOOKUP_SUPPORTED Notify payload. | SHOULD | Not support | | HTTP_CERT_LOOKUP_SUPPORTED is "not support" | |
| 59 | 3256 | Note that the term "Certificate Payload" is somewhat misleading, because not all authentication mechanisms use certificates and data other than certificates may be passed in this payload. | | Not support | | Explanation | |
| 59 | 3266 | The Certificate Payload is defined as follows: | | Not support | | Explanation | |
| 59 | 3268 | <pre> 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 ! +++++ ! Cert Encoding ! +++++ ~ Certificate Data ~ ! +++++ </pre> | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 | |
| 59 | 3281 | o Certificate Encoding (1 octet) - This field indicates the type of certificate or certificate-related information contained in the Certificate Data field. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 | |
| 59 | 3285 | Certificate Encoding Value | | Not support | | Explanation | |
| 59 | 3287 | RESERVED 0 | | Not support | | Explanation | |
| 59 | 3288 | PKCS #7 wrapped X.509 certificate 1 | | Not support | | Explanation | |
| 59 | 3289 | PGP Certificate 2 | | Not support | | Explanation | |
| 59 | 3290 | DNS Signed Key 3 | | Not support | | Explanation | |
| 59 | 3291 | X.509 Certificate - Signature 4 | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 | |
| 59 | 3292 | Kerberos Token 6 | | Not support | | Explanation | |
| 59 | 3293 | Certificate Revocation List (CRL) 7 | | Not support | | Explanation | |
| 59 | 3294 | Authority Revocation List (ARL) 8 | | Not support | | Explanation | |
| 59 | 3295 | SPKI Certificate 9 | | Not support | | Explanation | |
| 59 | 3296 | X.509 Certificate - Attribute 10 | | Not support | | Explanation | |
| 59 | 3297 | Raw RSA Key 11 | | Not support | | Explanation | |
| 59 | 3298 | Hash and URL of X.509 certificate 12 | | Not support | | Explanation | |
| 59 | 3299 | Hash and URL of X.509 bundle 13 | | Not support | | Explanation | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------------|------------------|--|--|
| 59 | 3300 | RESERVED to IANA PRIVATE USE | 14 - 200 201 - 255 | | Not support | Explanation |
| 60 | 3310 | o Certificate Data (variable length) - Actual encoding of certificate data. The type of certificate is indicated by the Certificate Encoding field. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 |
| 60 | 3314 | The payload type for the Certificate Payload is thirty seven (37). | | | Not support | Explanation |
| 60 | 3316 | Specific syntax is for some of the certificate type codes above is not defined in this document. The types whose syntax is defined in this document are: | | | Not support | Explanation |
| 60 | 3320 | X.509 Certificate - Signature (4) contains a DER encoded X.509 certificate whose public key is used to validate the sender's AUTH payload. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 |
| 60 | 3324 | Certificate Revocation List (7) contains a DER encoded X.509 certificate revocation list. | | | Not support | Explanation |
| 60 | 3327 | Raw RSA Key (11) contains a PKCS #1 encoded RSA key (see [RSA] and [PKCS1]). | | | Not support | Explanation |
| 60 | 3330 | Hash and URL encodings (12-13) allow IKE messages to remain short by replacing long data structures with a 20 octet SHA-1 hash (see [SHA]) of the replaced value followed by a variable-length URL that resolves to the DER encoded data structure itself. | | | Not support | Explanation |
| 60 | 3333 | This improves efficiency when the endpoints have certificate data cached and makes IKE less subject to denial of service attacks that become easier to mount when IKE messages are large enough to require IP fragmentation [KPS03]. | | | Not support | Explanation |
| 60 | 3339 | Use the following ASN.1 definition for an X.509 bundle: | | | Not support | Explanation |
| 60 | 3341 | CertBundle { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-cert-bundle(34) } DEFINITIONS EXPLICIT TAGS ::= BEGIN IMPORTS Certificate, CertificateList FROM PKIX1Explicit88 { iso(1) identified-organization(3) dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18) }; CertificateOrCRL ::= CHOICE { cert [0] Certificate, crl [1] CertificateList } CertificateBundle ::= SEQUENCE OF CertificateOrCRL END | | Not support | | Explanation |
| 61 | 3374 | Implementations MUST be capable of being configured to send and accept up to four X.509 certificates in support of authentication, and also MUST be capable of being configured to send and accept the first two Hash and URL formats (with HTTP URLs). | MUST MUST | Not support | | Internal process |
| 61 | 3376 | Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. | SHOULD | Not support | | Internal process |
| 61 | 3379 | If multiple certificates are sent, the first certificate MUST contain the public key used to sign the AUTH payload. The other certificates may be sent in any order. | MUST | Not support | | Difficult to test |
| 61 | 3383 | 3.7. Certificate Request Payload | | | | |
| 61 | 3385 | The Certificate Request Payload, denoted CERTREQ in this memo, provides a means to request preferred certificates via IKE and can appear in the IKE_INIT_SA response and/or the IKE_AUTH request. Certificate Request payloads MAY be included in an exchange when the sender needs to get the certificate of the receiver. | MAY | Not support | | Not need to test |
| 61 | 3385 | If multiple CAs are trusted and the cert encoding does not allow a list, then multiple Certificate Request payloads SHOULD be transmitted. | SHOULD | Not support | | Difficult to test |
| 61 | 3393 | The Certificate Request Payload is defined as follows: | | | Not support | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 61 | 3395 | <pre> 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 ! +++++ ! Cert Encoding ! +++++ ~ Certification Authority ~ ! +++++ </pre> <p>Figure 13: Certificate Request Payload Format</p> | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.1.10.3 SGW.I.1.1.1.10.3 SGW.R.1.1.1.10.3 |
| 61 | 3408 | o Certificate Encoding (1 octet) - Contains an encoding of the type or format of certificate requested. Values are listed in section 3.6. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.1.10.3 SGW.I.1.1.1.10.3 SGW.R.1.1.1.10.3 |
| 62 | 3422 | o Certification Authority (variable length) - Contains an encoding of an acceptable certification authority for the type of certificate requested. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.1.10.3 SGW.I.1.1.1.10.3 SGW.R.1.1.1.10.3 |
| 62 | 3426 | The payload type for the Certificate Request Payload is thirty eight (38). | | Not support | | Explanation |
| 62 | 3429 | The Certificate Encoding field has the same values as those defined in section 3.6. 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 [RFC3280]) from each Trust Anchor certificate. The twenty-octet hashes are concatenated and included with no other formatting. | | Not support | | Explanation |
| 62 | 3439 | Note that the term "Certificate Request" is somewhat misleading, in that values other than certificates are defined in a "Certificate" payload and requests for those values can be present in a Certificate Request Payload. The syntax of the Certificate Request payload in such cases is not defined in this document. | | Not support | | Explanation |
| 62 | 3445 | The Certificate Request Payload is processed by inspecting the "Cert Encoding" field to determine whether the processor has any certificates of this type. If so, the "Certification Authority" field is inspected to determine if the processor has any certificates that can be validated up to one of the specified certification authorities. This can be a chain of certificates. | | Not support | | Explanation |
| 62 | 3452 | If an end-entity certificate exists that satisfies the criteria specified in the CERTREQ, a certificate or certificate chain SHOULD be sent back to the certificate requestor if the recipient of the CERTREQ: - is configured to use certificate authentication, - is allowed to send a CERT payload, - has matching CA trust policy governing the current negotiation, and - has at least one time-wise and usage appropriate end-entity certificate chaining to a CA provided in the CERTREQ. | SHOULD | Not support | | CERTREQ is "not support" |
| 62 | 3466 | Certificate revocation checking must be considered during the chaining process used to select a certificate. Note that even if two peers are configured to use two different CAs, cross-certification relationships should be supported by appropriate selection logic. | | Not support | | Explanation |
| 63 | 3478 | The intent is not to prevent communication through the strict adherence of selection of a certificate based on CERTREQ, when an alternate certificate could be selected by the sender that would still enable the recipient to successfully validate and trust it through trust conveyed by cross-certification, CRLs, or other out-of-band configured means. Thus, the processing of a CERTREQ should be seen as a suggestion for a certificate to select, not a mandated one. If no certificates exist, then the CERTREQ is ignored. This is not an error condition of the protocol. There may be cases where there is a preferred CA sent in the CERTREQ, but an alternate might be acceptable (perhaps after prompting a human operator). | | Not support | | Explanation |
| 63 | 3490 | 3.8 Authentication Payload | | | | |
| 63 | 3492 | The Authentication Payload, denoted AUTH in this memo, contains data used for authentication purposes. The syntax of the Authentication data varies according to the Auth Method as specified below. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 63 | 3496 | The Authentication Payload is defined as follows: | | Not support | | Explanation |
| 63 | 3498 | <pre> 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 +++++ ! Next Payload !C! RESERVED ! Payload Length ! +++++ ! Auth Method ! RESERVED ! +++++ ! ~ Authentication Data ~ ! +++++ </pre> <p>Figure 14: Authentication Payload Format</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 63 | 3512 | o Auth Method (1 octet) - Specifies the method of authentication used. Values defined are: | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.I.1.1.10.3 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.1.1.10.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.I.1.1.10.3 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.2.1.1.1 SGWR.2.1.1.2 SGWR.1.1.10.3 |
| 63 | 3515 | RSA Digital Signature (1) - Computed as specified in section 2.15 using an RSA private key over a PKCS#1 padded hash (see [RSA] and [PKCS1]). | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGWR.1.1.10.3 |
| 63 | 3519 | Shared Key Message Integrity Code (2) - Computed as specified in section 2.15 using the shared key associated with the identity in the ID payload and the negotiated prf function | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 63 | 3523 | DSS Digital Signature (3) - Computed as specified in section 2.15 using a DSS private key (see [DSS]) over a SHA-1 hash. | | Not support | | Explanation |
| 64 | 3534 | The values 0 and 4-200 are reserved to IANA. The values 201-255 are available for private use. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 64 | 3537 | o Authentication Data (variable length) - see section 2.15. | | | | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.I.1.1.10.3 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.1.1.10.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.I.1.1.10.3 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.2.1.1.1 SGWR.2.1.1.2 SGWR.1.1.10.3 |
| 64 | 3539 | The payload type for the Authentication Payload is thirty nine (39). | | Not support | | Explanation |
| 64 | 3541 | 3.9. Nonce Payload | | | | |
| 64 | 3543 | The Nonce Payload, denoted Ni and Nr in this memo for the initiator's and responder's nonce respectively, contains random data used to guarantee liveness during an exchange and protect against replay attacks. | | Not support | | Explanation |
| 64 | 3548 | The Nonce Payload is defined as follows: | | Not support | | Explanation |
| 64 | 3550 | <pre> 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 ! +++++ ! ~ Nonce Data ! ! +++++ </pre> <p>Figure 15: Nonce Payload Format</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 SGW.I.1.1.1.1 SGW.R.1.1.1.1 |
| 64 | 3562 | o Nonce Data (variable length) - Contains the random data generated by the transmitting entity. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 SGW.I.1.1.1.1 SGW.R.1.1.1.1 |
| 64 | 3565 | The payload type for the Nonce Payload is forty (40). | | Not support | | |
| 64 | 3567 | The size of a Nonce MUST be between 16 and 256 octets inclusive. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 SGW.I.1.1.1.1 SGW.R.1.1.1.1 |
| 64 | 3567 | Nonce values MUST NOT be reused. | MUST NOT | Not support | | Difficult to test |
| 64 | 3570 | 3.10. Notify Payload | | | | |
| 64 | 3572 | 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. | | Not support | | Explanation |
| 65 | 3590 | The Notify Payload is defined as follows: | | Not support | | Explanation |
| 65 | 3592 | <pre> 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 ! Notify Message Type ! +++++ ! ~ Security Parameter Index (SPI) ! ! +++++ ! ~ Notification Data ! ! +++++ </pre> <p>Figure 16: Notify Payload Format</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 65 | 3610 | o Protocol ID (1 octet) - If this notification concerns an existing SA, this field indicates the type of that SA. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 65 | 3611 | For IKE_SA notifications, this field MUST be one (1). | MUST | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 65 | 3612 | For notifications concerning IPsec SAs this field MUST contain either | MUST | Not support | | Explanation |
| 65 | 3613 | (2) to indicate AH or | MUST | Not support | | Explanation |
| 65 | 3614 | (3) to indicate ESP. | MUST | Not support | | Explanation |
| 65 | 3614 | For notifications that do not relate to an existing SA, this field MUST be sent as zero | MUST | Not support | | Explanation |
| 65 | 3614 | and MUST be ignored on receipt. | MUST | Not support | | Explanation |
| 65 | 3616 | All other values for this field are reserved to IANA for future assignment. | | Not support | | Explanation |
| 65 | 3619 | 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. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 65 | 3623 | o Notify Message Type (2 octets) - Specifies the type of notification message. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 65 | 3626 | o SPI (variable length) - Security Parameter Index. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 65 | 3628 | o Notification Data (variable length) - Informational or error data transmitted in addition to the Notify Message Type. Values for this field are type specific (see elow). | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 65 | 3632 | The payload type for the Notify Payload is forty one (41). | | Not support | | Explanation |
| 66 | 3646 | 3.10.1. Notify Message Types | | | | |
| 66 | 3648 | 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. The table below lists the Notification messages and their corresponding values. The number of different error statuses was greatly reduced from IKEv1 both for simplification and to avoid giving configuration information to probers. | | Not support | | Explanation |
| 66 | 3656 | Types in the range 0 - 16383 are intended for reporting errors. | | Not support | | Explanation |
| 66 | 3656 | 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. | MUST | Not support | EN(initiator) SGW(initiator) | EN.I.1.1.11.4 SGW.I.1.1.11.4 |
| 66 | 3659 | Unrecognized error types in a request and status types in a request or response MUST be ignored except that they SHOULD be logged. | MUST SHOULD | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.5 EN.R.1.1.11.5 SGW.I.1.1.11.5 SGW.R.1.1.11.5 |
| 66 | 3663 | 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. | MAY MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.5 EN.R.1.1.11.5 SGW.I.1.1.11.5 SGW.R.1.1.11.5 |
| 66 | 3668 | NOTIFY MESSAGES - ERROR TYPES ----- Value | | Not support | | Explanation |
| 66 | 3670 | RESERVED 0 | | Not support | | Explanation |
| 66 | 3672 | UNSUPPORTED_CRITICAL_PAYLOAD 1 Sent if the payload has the "critical" bit set and the payload type is not recognized. Notification Data contains the one-octet payload type. | | BASIC | EN(responder) SGW(responder) | EN.R.1.1.4.4 SGW.R.1.1.4.4 |
| 66 | 3678 | INVALID_IKE_SPI 4 Indicates an IKE message was received with an unrecognized destination SPI. | | Not support | | untestable |
| 66 | 3681 | This usually indicates that the recipient has rebooted and forgotten the existence of an IKE_SA. | | Not support | | Explanation |
| 66 | 3684 | INVALID_MAJOR_VERSION 5 Indicates the recipient cannot handle the version of IKE specified in the header. The closest version number that the recipient can support will be in the reply header. | | BASIC | SGW(initiator) EN(responder) | EN.R.1.1.4.2 SGW.I.1.1.4.2 |
| 66 | 3690 | 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. | | Not support | | untestable |
| 66 | 3694 | To avoid a denial of service 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. | | Not support | | Explanation |
| 67 | 3705 | 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. | MUST | Not support | | untestable |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|--|--|--|
| 67 | 3708 | To aid debugging, more detailed error information SHOULD be written to a console or log. | SHOULD | Not support | | Internal process |
| 67 | 3712 | INVALID_MESSAGE_ID 9 Sent when an IKE message ID outside the supported window is received. | | Not support | | Explanation |
| 67 | 3715 | This Notify MUST NOT be sent in a response; | MUST NOT | Not support | | INVALID_MESS AGE_ID is "Not support" |
| 67 | 3715 | the invalid request MUST NOT be acknowledged. | MUST NOT | Not support | | INVALID_MESS AGE_ID is "Not support" |
| 67 | 3715 | Instead, inform the other side by initiating an INFORMATIONAL exchange with Notification data containing the four octet invalid message ID. | | Not support | | Explanation |
| 67 | 3715 | Sending this notification is optional, and notifications of this type MUST be rate limited. | MUST | Not support | | INVALID_MESS AGE_ID is "Not support" |
| 67 | 3722 | INVALID_SPI 11 MAY be sent in an IKE INFORMATIONAL exchange when a node receives an ESP or AH packet with an invalid SPI. The Notification Data contains the SPI of the invalid packet. | MAY | Not support | | untestable |
| 67 | 3725 | This usually indicates a node has rebooted and forgotten an SA. If this Informational Message is sent outside the context of an IKE_SA, it should be used by the recipient only as a "hint" that something might be wrong (because it could easily be forged). | | Not support | | Explanation |
| 67 | 3733 | NO_PROPOSAL_CHOSEN 14 None of the proposed crypto suites was acceptable. | | BASIC | EN(responder) SGW(responder) | EN.R.1.2.4.1 EN.R.1.2.6.9 SGW.R.1.2.4.1 SGW.R.1.2.6.9 |
| 67 | 3737 | INVALID_KE_PAYLOAD 17 The D-H Group # field in the KE payload is not the group # selected by the responder for this exchange. There are two octets of data associated with this notification: the accepted D-H Group # in big endian order. | | ADVANCED *Because DH#14 is ADVANCED group. | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.7 EN.R.1.1.6.7 SGW.I.1.1.6.7 SGW.R.1.1.6.7 |
| 67 | 3744 | AUTHENTICATION_FAILED 24 Sent in the response to an IKE_AUTH message when for some reason the authentication failed. There is no associated data. | | BASIC | EN(responder) SGW(responder) | EN.R.1.2.8.1 SGW.R.1.2.8.1 |
| 68 | 3758 | SINGLE_PAIR_REQUIRED 34 This error indicates that a CREATE_CHILD_SA request is unacceptable because its sender is only willing to accept traffic selectors specifying a single pair of addresses. | | Not support | | Explanation |
| 68 | 3762 | The requestor is expected to respond by requesting an SA for only the specific traffic it is trying to forward. | | Not support | | Explanation |
| 68 | 3766 | NO_ADDITIONAL_SAS 35 This error indicates that a CREATE_CHILD_SA request is unacceptable because the responder is unwilling to accept any more CHILD_SAs on this IKE_SA. | | Not support | | Explanation |
| 68 | 3770 | Some minimal implementations may only accept a single CHILD_SA setup in the context of an initial IKE exchange and reject any subsequent attempts to add more. | | Not support | | Explanation |
| 68 | 3774 | INTERNAL_ADDRESS_FAILURE 36 Indicates an error assigning an internal address (i.e., INTERNAL_IP4_ADDRESS or INTERNAL_IP6_ADDRESS) during the processing of a Configuration Payload by a responder. | | Not support | | Explanation |
| 68 | 3778 | If this error is generated within an IKE_AUTH exchange, no CHILD_SA will be created. | | Not support | | Explanation |
| 68 | 3782 | FAILED_CP_REQUIRED 37 Sent by responder in the case where CP(CFG_REQUEST) was expected but not received, and so is a conflict with locally configured policy. There is no associated data. | | ADVANCED | SGW(responder) | SGW.R.2.1.2.3 |
| 68 | 3788 | TS_UNACCEPTABLE 38 Indicates that none of the addresses/protocols/ports in the supplied traffic selectors is acceptable. | | BASIC | EN(responder) SGW(responder) | EN.R.1.1.7.2 SGW.R.1.1.7.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|----------------------------------|
| 68 | 3793 | INVALID_SELECTORS 39 MAY be sent in an IKE INFORMATIONAL exchange when a node receives an ESP or AH packet whose selectors do not match those of the SA on which it was delivered (and that caused the packet to be dropped). | MAY | Not support | | untestable |
| 68 | 3798 | The Notification Data contains the start of the offending packet (as in ICMP messages) and the SPI field of the notification is set to match the SPI of the IPsec SA. | | Not support | | Explanation |
| 68 | 3803 | RESERVED TO IANA - Error types 40 - 8191 | | Not support | | Explanation |
| 68 | 3805 | Private Use - Errors 8192 - 16383 | | Not support | | Explanation |
| 69 | 3814 | NOTIFY MESSAGES - STATUS TYPES Value ----- | | Not support | | Explanation |
| 69 | 3817 | INITIAL_CONTACT 16384 This notification asserts that this IKE_SA is the only IKE_SA currently active between the authenticated identities. | | Not support | | untestable |
| 69 | 3821 | It MAY be sent when an IKE_SA is established after a crash, and the recipient MAY use this information to delete any other IKE_SAs it has to the same authenticated identity without waiting for a timeout. | MAY MAY | Not support | | Explanation |
| 69 | 3824 | This notification MUST NOT be sent by an entity that may be replicated (e.g., a roaming user's credentials where the user is allowed to connect to the corporate firewall from two remote systems at the same time). | MUST NOT | Not support | | difficult to test |
| 69 | 3830 | SET_WINDOW_SIZE 16385 This notification asserts that the sending endpoint is capable of keeping state for multiple outstanding exchanges, permitting the recipient to send multiple requests before getting a response to the first. | | Not support | | Explanation |
| 69 | 3835 | The data associated with a SET_WINDOW_SIZE notification MUST be 4 octets long and contain the big endian representation of the number of messages the sender promises to keep. | MUST | Not support | | SET_WINDOW_SIZE is "Not support" |
| 69 | 3838 | Window size is always one until the initial exchanges complete. | | Not support | | Explanation |
| 69 | 3841 | ADDITIONAL_TS_POSSIBLE 16386 This notification asserts that the sending endpoint narrowed the proposed traffic selectors but that other traffic selectors would also have been acceptable, though only in a separate SA (see section 2.9). There is no data associated with this Notify type. | | Not support | | Explanation |
| 69 | 3846 | It may be sent only as an additional payload in a message including accepted TSs. | | Not support | | Explanation |
| 69 | 3850 | IPCOMP_SUPPORTED 16387 This notification may be included only in a message containing an SA payload negotiating a CHLD_SA and indicates a willingness by its sender to use IPComp on this SA. | | Not support | | Explanation |
| 69 | 3855 | The data associated with this notification includes a two-octet IPComp CPI followed by a one-octet transform ID optionally followed by attributes whose length and format are defined by that transform ID. A message proposing an SA may contain multiple IPCOMP_SUPPORTED notifications to indicate multiple supported algorithms. A message accepting an SA may contain at most one. | | Not support | | Explanation |
| 70 | 3870 | The transform IDs currently defined are: | | Not support | | Explanation |
| 70 | 3872 | NAME NUMBER DEFINED IN ----- | | Not support | | Explanation |
| 70 | 3874 | RESERVED 0 | | Not support | | Explanation |
| 70 | 3875 | IPCOMP_OUI 1 | | Not support | | Explanation |
| 70 | 3876 | IPCOMP_DEFLATE 2 RFC 2394 | | Not support | | Explanation |
| 70 | 3877 | IPCOMP_LZS 3 RFC 2395 | | Not support | | Explanation |
| 70 | 3878 | IPCOMP_LZJH 4 RFC 3051 | | Not support | | Explanation |
| 70 | 3880 | values 5-240 are reserved to IANA. Values 241-255 are for private use among mutually consenting parties. | | Not support | | Explanation |
| 70 | 3883 | NAT_DETECTION_SOURCE_IP 16388 This notification is used by its recipient to determine whether the source is behind a NAT box. | | Not support | | Explanation |
| 70 | 3886 | The data associated with this notification is a SHA-1 digest of the SPIs (in the order they appear in the header), IP address, and port on which this packet was sent. | | Not support | | Explanation |
| 70 | 3886 | There MAY be multiple Notify payloads of this type in a message if the sender does not know which of several network attachments will be used to send the packet. | MAY | Not support | | NAT Traversal is "Not support" |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 70 | 3886 | The recipient of this notification MAY compare the supplied value to a SHA-1 hash of the SPIs, source IP address, and port, and if they don't match it SHOULD enable NAT traversal (see section 2.23). | MAY SHOULD | Not support | | NAT Traversal is "Not support" |
| 70 | 3886 | Alternately, it MAY reject the connection attempt if NAT traversal is not supported. | MAY | Not support | | Not need to test |
| 70 | 3899 | NAT_DETECTION_DESTINATION_IP 16389 This notification is used by its recipient to determine whether it is behind a NAT box. | | Not support | | Explanation |
| 70 | 3902 | The data associated with this notification is a SHA-1 digest of the SPIs (in the order they appear in the header), IP address, and port to which this packet was sent. | | Not support | | Explanation |
| 70 | 3902 | The recipient of this notification MAY compare the supplied value to a hash of the SPIs, destination IP address, and port, and if they don't match it SHOULD invoke NAT traversal (see section 2.23). | MAY SHOULD | Not support | | NAT Traversal is "not support" |
| 70 | 3902 | If they don't match, it means that this end is behind a NAT and this end SHOULD start sending keepalive packets as defined in [Hutt05]. | | Not support | | Explanation |
| 70 | 3902 | Alternately, it MAY reject the connection attempt if NAT traversal is not supported. | MAY | Not support | | Not need to test |
| 71 | 3926 | COOKIE 16390 This notification MAY be included in an IKE_SA_INIT response. | MAY | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.1.5.1 SGW.I.1.1.5.1 |
| 71 | 3929 | It indicates that the request should be retried with a copy of this notification as the first payload. | | Not support | | Explanation |
| 71 | 3930 | This notification MUST be included in an IKE_SA_INIT request retry if a COOKIE notification was included in the initial response. | MUST | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.1.5.1 SGW.I.1.1.5.1 |
| 71 | 3933 | The data associated with this notification MUST be between 1 and 64 octets in length (inclusive). | MUST | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.1.5.1 SGW.I.1.1.5.1 |
| 71 | 3936 | USE_TRANSPORT_MODE 16391 This notification MAY be included in a request message that also includes an SA payload requesting a CHILD_SA. | MAY | BASIC | EN(initiator) EN(responder) | EN.I.1.1.1.2 EN.R.1.1.1.2 |
| 71 | 3939 | It requests that the CHILD_SA use transport mode rather than tunnel mode for the SA created. | | Not support | | Explanation |
| 71 | 3941 | If the request is accepted, the response MUST also include a notification of type USE_TRANSPORT_MODE. If the responder declines the request, the CHILD_SA will be established in tunnel mode. | MUST | BASIC | EN(responder) | EN.R.1.1.1.2 |
| 71 | 3944 | If this is unacceptable to the initiator, the initiator MUST delete the SA. | MUST | Not support | | Difficult to test |
| 71 | 3946 | Note: Except when using this option to negotiate transport mode, all CHILD_SAs will use tunnel mode. | | Not support | | Explanation |
| 71 | 3949 | Note: The ECN decapsulation modifications specified in [RFC4301] MUST be performed for every tunnel mode SA created by IKEv2. | MUST | Not support | | ECN is "Not support" |
| 71 | 3953 | HTTP_CERT_LOOKUP_SUPPORTED 16392 This notification MAY be included in any message that can include a CERTREQ payload and indicates that the sender is capable of looking up certificates based on an HTTP-based URL (and hence presumably would prefer to receive certificate specifications in that format). | MAY | Not support | | Not need to test |
| 71 | 3961 | REKEY_SA 16393 This notification MUST be included in a CREATE_CHILD_SA exchange if the purpose of the exchange is to replace an existing ESP or AH SA. The SPI field identifies the SA being rekeyed. There is no data. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 71 | 3968 | ESP_TFC_PADDING_NOT_SUPPORTED 16394 This notification asserts that the sending endpoint will NOT accept packets that contain Flow Confidentiality (TFC) padding. | | Not support | | Explanation |
| 72 | 3982 | NON_FIRST_FRAGMENTS_ALSO 16395 Used for fragmentation control. See [RFC4301] for Explanation. | | Not support | | Explanation |
| 72 | 3987 | RESERVED TO IANA - STATUS TYPES 16396 - 40959 | | Not support | | Explanation |
| 72 | 3989 | Private Use - STATUS TYPES 40960 - 65535 | | Not support | | Explanation |
| 72 | 3991 | 3.11. Delete Payload | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 72 | 3993 | The Delete Payload, denoted D in this memo, contains a protocol-specific security association identifier that the sender has removed from its security association database and is, therefore, no longer valid. Figure 17 shows the format of the Delete Payload. | | Not support | | Explanation |
| 72 | 3996 | It is possible to send multiple SPIs in a Delete payload; however, each SPI MUST be for the same protocol. | MUST | Not support | EN(responder) SGW(responder) | EN.R.1.2.3.1 SGW.R.1.2.3.1 |
| 72 | 3998 | Mixing of protocol identifiers MUST NOT be performed in a Delete payload. | MUST NOT | Not support | | Not need to test |
| 72 | 3999 | It is permitted, however, to include multiple Delete payloads in a single INFORMATIONAL exchange where each Delete payload lists SPIs for a different protocol. | | Not support | | Explanation |
| 72 | 4003 | Deletion of the IKE_SA is indicated by a protocol ID of 1 (IKE) but no SPIs. Deletion of a CHILD_SA, such as ESP or AH, will contain the IPsec protocol ID of that protocol (2 for AH, 3 for ESP), and the SPI is the SPI the sending endpoint would expect in inbound ESP or AH packets. | | Not support | | Explanation |
| 72 | 4009 | The Delete Payload is defined as follows: | | Not support | | Explanation |
| 72 | 4011 | <pre> 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 ! # of SPIs ! +++++ ! ! Security Parameter Index(es) (SPI) ! ! +++++ </pre> <p>Figure 17: Delete Payload Format</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.8 EN.I.1.1.3.9 EN.I.1.1.3.10 EN.R.1.1.3.6 EN.R.1.1.3.7 EN.R.1.1.3.8 EN.R.1.1.3.9 EN.R.1.1.3.10 EN.R.1.1.3.10 SGW.R.1.1.3.6 SGW.R.1.1.3.7 SGW.R.1.1.3.8 SGW.R.1.1.3.9 |
| 72 | 4025 | o Protocol ID (1 octet) - Must be 1 for an IKE_SA, 2 for AH, or 3 for ESP. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.8 EN.I.1.1.3.9 EN.I.1.1.3.10 EN.R.1.1.3.6 EN.R.1.1.3.7 EN.R.1.1.3.8 EN.R.1.1.3.9 EN.R.1.1.3.9 SGW.I.1.1.3.9 SGW.I.1.1.3.10 SGW.R.1.1.3.6 SGW.R.1.1.3.7 SGW.R.1.1.3.8 SGW.R.1.1.3.9 |
| 73 | 4038 | 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) or four for AH and ESP. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.8 EN.I.1.1.3.9 EN.I.1.1.3.10 EN.R.1.1.3.6 EN.R.1.1.3.7 EN.R.1.1.3.8 EN.R.1.1.3.9 EN.R.1.1.3.9 SGW.I.1.1.3.9 SGW.I.1.1.3.10 SGW.R.1.1.3.6 SGW.R.1.1.3.7 SGW.R.1.1.3.8 SGW.R.1.1.3.9 |
| 73 | 4042 | o # of SPIs (2 octets) - The number of SPIs contained in the Delete payload. The size of each SPI is defined by the SPI Size field. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.8 EN.I.1.1.3.9 EN.I.1.1.3.10 EN.R.1.1.3.6 EN.R.1.1.3.7 EN.R.1.1.3.8 EN.R.1.1.3.9 EN.R.1.1.3.9 SGW.I.1.1.3.9 SGW.I.1.1.3.10 SGW.R.1.1.3.6 SGW.R.1.1.3.7 SGW.R.1.1.3.8 SGW.R.1.1.3.9 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--------|--|
| 73 | 4045 | 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 # of SPIs fields. | | | | EN.I.1.1.3.8 EN.I.1.1.3.9 EN.I.1.1.3.10 EN.R.1.1.3.6 EN.R.1.1.3.7 EN.R.1.1.3.8 EN.R.1.1.3.9 SGW.I.1.1.3.9 SGW.I.1.1.3.10 SGW.R.1.1.3.6 SGW.R.1.1.3.7 SGW.R.1.1.3.8 SGW.R.1.1.3.9 |
| 73 | 4049 | The payload type for the Delete Payload is forty two (42). | | Not support | | Explanation |
| 73 | 4051 | 3.12. Vendor ID Payload | | | | |
| 73 | 4053 | The Vendor ID Payload, denoted V in this memo, contains a vendor defined constant. The constant is used by vendors to identify and recognize remote instances of their implementations. This mechanism allows a vendor to experiment with new features while maintaining backward compatibility. | | Not support | | Explanation |
| 73 | 4059 | A Vendor ID payload MAY announce that the sender is capable to accepting certain extensions to the protocol, or it MAY simply identify the implementation as an aid in debugging. | MAY MAY | Not support | | Not need to test |
| 73 | 4059 | A Vendor ID payload MUST NOT change the interpretation of any information defined in this specification | MUST NOT | Not support | | Not need to test |
| 73 | 4059 | (i.e., the critical bit MUST be set to 0). | MUST | Not support | | Not need to test |
| 73 | 4059 | Multiple Vendor ID payloads MAY be sent. | MAY | Not support | | Not need to test |
| 73 | 4059 | An implementation is NOT REQUIRED to send any Vendor ID payload at all. | | Not support | | Explanation |
| 73 | 4067 | A Vendor ID payload may be sent as part of any message. Reception of a familiar Vendor ID payload allows an implementation to make use of Private USE numbers described throughout this memo -- private payloads, private exchanges, private notifications, etc. Unfamiliar Vendor IDs MUST be ignored. | MUST | Not support | | Not need to test |
| 73 | 4073 | Writers of Internet-Drafts who wish to extend this protocol MUST define a Vendor ID payload to announce the ability to implement the extension in the Internet-Draft. It is expected that Internet-Drafts that gain acceptance and are standardized will be given "magic numbers" out of the Future Use range by IANA, and the requirement to use a Vendor ID will go away. | MUST | Not support | | Not need to test |
| 74 | 4094 | The Vendor ID Payload fields are defined as follows: | | Not support | | Explanation |
| 74 | 4096 | <pre> 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 +++++ ! Next Payload !C! RESERVED ! Payload Length ! +++++ ! ~ Vendor ID (VID) ~ ! +++++ </pre> <p>Figure 18: Vendor ID Payload Format</p> | | Not support | | Explanation |
| 74 | 4108 | o Vendor ID (variable length) - It is the responsibility of the person choosing the Vendor ID to assure its uniqueness in spite of the absence of any central registry for IDs. Good practice is to include a company name, a person name, or some such. If you want to show off, you might include the latitude and longitude and time where you were when you chose the ID and some random input. A message digest of a long unique string is preferable to the long unique string itself. | | Not support | | Explanation |
| 74 | 4117 | The payload type for the Vendor ID Payload is forty three (43). | | Not support | | Explanation |
| 74 | 4119 | 3.13. Traffic Selector Payload | | | | |
| 74 | 4121 | The Traffic Selector Payload, denoted TS in this memo, allows peers to identify packet flows for processing by IPsec security services. The Traffic Selector Payload consists of the IKE generic payload header followed by individual traffic selectors as follows: | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 74 | 4126 | <pre> 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 ! +++++ ! Number of TSs ! RESERVED ! +++++ ! ~ <Traffic Selectors> ~ ! +++++ </pre> <p>Figure 19: Traffic Selectors Payload Format</p> | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 74 | 4140 | o Number of TSs (1 octet) - Number of traffic selectors being provided. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 75 | 4150 | o RESERVED - This field MUST be sent as zero | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 75 | 4150 | and MUST be ignored on receipt. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.11.2 EN.R.1.1.11.2 SGW.I.1.1.11.2 SGW.R.1.1.11.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 75 | 4153 | o Traffic Selectors (variable length) - One or more individual traffic selectors. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 75 | 4156 | The length of the Traffic Selector payload includes the TS header and all the traffic selectors. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 75 | 4159 | The payload type for the Traffic Selector payload is forty four (44) for addresses at the initiator's end of the SA | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|---|
| 75 | 4159 | and forty five (45) for addresses at the responder's end. | | | | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 75 | 4163 | 3.13.1. Traffic Selector | | | | |
| 75 | 4165 | <pre> 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 +++++ ! TS Type IP Protocol ID* Selector Length +++++ Start Port* End Port* +++++ ! ~ Starting Address* ~ ! +++++ ! ~ Ending Address* ~ ! +++++ </pre> <p>Figure 20: Traffic Selector</p> | | | | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 75 | 4183 | * Note: All fields other than TS Type and Selector Length depend on the TS Type. The fields shown are for TS Types 7 and 8, the only two values currently defined. | | | | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 75 | 4187 | o TS Type (one octet) - Specifies the type of traffic selector. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 75 | 4189 | o IP protocol ID (1 octet) - Value specifying an associated IP protocol ID (e.g., UDP/TCP/ICMP). A value of zero means that the protocol ID is not relevant to this traffic selector -- the SA can carry all protocols. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 75 | 4194 | o Selector Length - Specifies the length of this Traffic Selector Substructure including the header. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 76 | 4206 | o Start Port (2 octets) - Value specifying the smallest port number allowed by this Traffic Selector. For protocols for which port is undefined, or if all ports are allowed, this field MUST be zero. For the ICMP protocol, the two one-octet fields Type and Code are treated as a single 16-bit integer (with Type in the most significant eight bits and Code in the least significant eight bits) port number for the purposes of filtering based on this field. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 76 | 4215 | o End Port (2 octets) - Value specifying the largest port number allowed by this Traffic Selector. For protocols for which port is undefined, or if all ports are allowed, this field MUST be 65535. For the ICMP protocol, the two one-octet fields Type and Code are treated as a single 16-bit integer (with Type in the most significant eight bits and Code in the least significant eight bits) port number for the purposed of filtering based on this field. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 76 | 4224 | o Starting Address - The smallest address included in this Traffic Selector (length determined by TS type). | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.3 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 77 | 4276 | The Encrypted Payload, if present in a message, MUST be the last payload in the message. Often, it is the only payload in the message. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 77 | 4280 | The algorithms for encryption and integrity protection are negotiated during IKE_SA setup, and the keys are computed as specified in sections 2.14 and 2.18. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 77 | 4284 | The encryption and integrity protection algorithms are modeled after the ESP algorithms described in RFCs 2104 [KBC96], 4303 [RFC4303], and 2451 [ESPCBC]. This document completely specifies the cryptographic processing of IKE data, but those documents should be consulted for design rationale. We require a block cipher with a fixed block size and an integrity check algorithm that computes a fixed-length checksum over a variable size message. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 77 | 4292 | The payload type for an Encrypted payload is forty six (46). The Encrypted Payload consists of the IKE generic payload header followed by individual fields as follows: | | | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 78 | 4318 | <pre> 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 ! +++++ ! Initialization Vector ! ! (length is block size for encryption algorithm) ! +++++ ~ Encrypted IKE Payloads ~ + ! Padding (0-255 octets) ! +++++ ! Pad Length ! +++++ ~ Integrity Checksum Data ~ +++++ </pre> <p>Figure 21: Encrypted Payload Format</p> | | | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 78 | 4337 | 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. | | | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 78 | 4345 | o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 78 | 4348 | o Initialization Vector - A randomly chosen value whose length is equal to the block length of the underlying encryption algorithm. Recipients MUST accept any value. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 78 | 4348 | Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. | SHOULD | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|---|
| 78 | 4348 | Senders MUST NOT use the same value for each message, use a sequence of values with low hamming distance (e.g., a sequence number), or use ciphertext from a received message. | MUST NOT | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 78 | 4357 | o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |
| 78 | 4360 | 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. | MAY MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGWR.1.1.1.2 SGWR.1.1.1.3 SGWR.1.2.1.1 SGWR.1.3.1.1 SGWR.2.1.1.1 SGWR.2.1.1.2 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 79 | 4374 | 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, | SHOULD | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 79 | 4374 | but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 79 | 4381 | 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. | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 EN.R.2.1.1.2 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 79 | 4386 | 3.15. Configuration Payload | | | | |
| 79 | 4388 | The Configuration payload, denoted CP in this document, is used to exchange configuration information between IKE peers. The exchange is for an IRAC to request an internal IP address from an IRAS and to exchange other information of the sort that one would acquire with Dynamic Host Configuration Protocol (DHCP) if the IRAC were directly connected to a LAN. | | Not support | | Explanation |
| 79 | 4395 | Configuration payloads are of type CFG_REQUEST/CFG_REPLY or CFG_SET/CFG_ACK (see CFG Type in the payload description below). CFG_REQUEST and CFG_SET payloads may optionally be added to any IKE request. | | ADVANCED | EN(initiator) SGW(responder) | EN.I.2.1.2.1 SGW.R.2.1.2.1 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-------------------------|------------------|---------------------------------|--|
| 79 | 4398 | The IKE response MUST include either a corresponding CFG_REPLY or CFG_ACK or a Notify payload with an error type indicating why the request could not be honored. | MUST | ADVANCED | SGW(responder) | SGW.R.2.1.2.2 |
| 79 | 4400 | An exception is that a minimal implementation MAY ignore all CFG_REQUEST and CFG_SET payloads, so a response message without a corresponding CFG_REPLY or CFG_ACK MUST be accepted as an indication that the request was not supported. | MAY MUST | ADVANCED | EN(initiator) | EN.I.2.1.2.4 |
| 79 | 4406 | "CFG_REQUEST/CFG_REPLY" allows an IKE endpoint to request information from its peer. | | Not support | | Explanation |
| 79 | 4406 | If an attribute in the CFG_REQUEST Configuration Payload is not zero-length, it is taken as a suggestion for that attribute. | | Not support | | Explanation |
| 79 | 4406 | The CFG_REPLY Configuration Payload MAY return that value, or a new one. It MAY also add new attributes and not include some requested ones. | MAY MAY | Not support | | Not need to test |
| 79 | 4406 | Requestors MUST ignore returned attributes that they do not recognize. | MUST | ADVANCED | EN(initiator) | EN.I.2.1.2.5 |
| 79 | 4414 | Some attributes MAY be multi-valued, in which case multiple attribute values of the same type are sent and/or returned. Generally, all values of an attribute are returned when the attribute is requested. | MAY | Not support | | Not need to test |
| 79 | 4414 | For some attributes (in this version of the specification only internal addresses), multiple requests indicates a request that multiple values be assigned. For these attributes, the number of values returned SHOULD NOT exceed the number requested. | SHOULD NOT | ADVANCED | SGW(responder) | SGW.R.2.1.2.5 |
| 80 | 4430 | If the data type requested in a CFG_REQUEST is not recognized or not supported, the responder MUST NOT return an error type but rather MUST either send a CFG_REPLY that MAY be empty or a reply not containing a CFG_REPLY payload at all. Error returns are reserved for cases where the request is recognized but cannot be performed as requested or the request is badly formatted. | MUST NOT MUST MAY | ADVANCED | SGW(responder) | SGW.R.2.1.2.6 |
| 80 | 4437 | "CFG_SET/CFG_ACK" allows an IKE endpoint to push configuration data to its peer. | | Not support | | Explanation |
| 80 | 4438 | In this case, the CFG_SET Configuration Payload contains attributes the initiator wants its peer to alter. The responder MUST return a Configuration Payload if it accepted any of the configuration data | MUST | Not support | | CFG-SET/ACK is "Not support." |
| 80 | 4438 | and it MUST contain the attributes that the responder accepted with zero-length data. | MUST | Not support | | CFG-SET/ACK is "Not support." |
| 80 | 4438 | Those attributes that it did not accept MUST NOT be in the CFG_ACK Configuration Payload. | MUST NOT | Not support | | CFG-SET/ACK is "Not support." |
| 80 | 4438 | If no attributes were accepted, the responder MUST return either an empty CFG_ACK payload or a response message without a CFG_ACK payload. | MUST | Not support | | CFG-SET/ACK is "Not support." |
| 80 | 4446 | There are currently no defined uses for the CFG_SET/CFG_ACK exchange, though they may be used in connection with extensions based on Vendor IDs. | | Not support | | Explanation |
| 80 | 4446 | An minimal implementation of this specification MAY ignore CFG_SET payloads. | MAY | Not support | | CFG-SET/ACK is "Not support." |
| 80 | 4451 | Extensions via the CP payload SHOULD NOT be used for general purpose management. Its main intent is to provide a bootstrap mechanism to exchange information within IPsec from IRAS to IRAC. | SHOULD NOT | Not support | | Not need to test |
| 80 | 4451 | While it MAY be useful to use such a method to exchange information between some Security Gateways (SGW) or small networks, existing management protocols such as DHCP [DHCP], RADIUS [RADIUS], SNMP, or LDAP [LDAP] should be preferred for enterprise management as well as subsequent information exchanges. | MAY | Not support | | Not need to test |
| 80 | 4460 | The Configuration Payload is defined as follows: | | ADVANCED | EN(initiator) SGW(responder) | EN.I.2.1.2.1 EN.I.2.1.2.2 SGW.R.2.1.2.1 SGW.R.2.1.2.2 |
| 80 | 4462 | <pre> 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 ! +++++ ! CFG Type ! RESERVED ! +++++ ! ~ Configuration Attributes ~ ! +++++ </pre> | | ADVANCED | EN(initiator) SGW(responder) | EN.I.2.1.2.1 EN.I.2.1.2.2 SGW.R.2.1.2.1 SGW.R.2.1.2.2 |
| | | Figure 22: Configuration Payload Format | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|---------------------------------|--|
| 82 | 4555 | INTERNAL_IP6_ADDRESS 8 YES* 0 or 17 octets | | ADVANCED | EN(initiator) SGW(responder) | EN.I.2.1.2.1 EN.I.2.1.2.2 SGW.R.2.1.2.1 SGW.R.2.1.2.2 |
| 82 | 4556 | RESERVED 9 | | Not support | | Explanation |
| 82 | 4557 | INTERNAL_IP6_DNS 10 YES 0 or 16 octets | | Not support | | Explanation |
| 82 | 4558 | INTERNAL_IP6_NBNS 11 YES 0 or 16 octets | | Not support | | Explanation |
| 82 | 4559 | INTERNAL_IP6_DHCP 12 YES 0 or 16 octets | | Not support | | Explanation |
| 82 | 4560 | INTERNAL_IP4_SUBNET 13 YES 0 or 8 octets | | Not support | | Explanation |
| 82 | 4561 | SUPPORTED_ATTRIBUTES 14 NO Multiple of 2 | | Not support | | Explanation |
| 82 | 4562 | INTERNAL_IP6_SUBNET 15 YES 17 octets | | Not support | | Explanation |
| 82 | 4564 | * These attributes may be multi-valued on return only if multiple values were requested. | | Not support | | Explanation |
| 82 | 4567 | Types 16-16383 are reserved to IANA. Values 16384-32767 are for private use among mutually consenting parties. | | Not support | | Explanation |
| 82 | 4570 | o INTERNAL_IP4_ADDRESS, INTERNAL_IP6_ADDRESS - An address on the internal network, sometimes called a red node address or private address and MAY be a private address on the Internet. In a request message, the address specified is a requested address (or zero if no specific address is requested). If a specific address is requested, it likely indicates that a previous connection existed with this address and the requester would like to reuse that address. | MAY | Not support | | Not need to test |
| 82 | 4577 | With IPv6, a requestor MAY supply the low-order address bytes it wants to use. Multiple internal addresses MAY be requested by requesting multiple internal address attributes. | MAY MAY | Not support | | Not need to test |
| 82 | 4580 | The responder MAY only send up to the number of addresses requested. The INTERNAL_IP6_ADDRESS is made up of two fields: the first is a sixteen-octet IPv6 address and the second is a one-octet prefix-length as defined in [ADDRIPV6]. | MAY | Not support | | Not need to test |
| 82 | 4586 | The requested address is valid until the expiry time defined with the INTERNAL_ADDRESS_EXPIRY attribute or there are no IKE_SAs between the peers. | | Not support | | Explanation |
| 83 | 4598 | o INTERNAL_IP4_NETMASK - The internal network's netmask. Only one netmask is allowed in the request and reply messages (e.g., 255.255.255.0), and it MUST be used only with an INTERNAL_IP4_ADDRESS attribute. | MUST | Not support | | IPv4 is out of scope |
| 83 | 4603 | o INTERNAL_IP4_DNS, INTERNAL_IP6_DNS - Specifies an address of a DNS server within the network. Multiple DNS servers MAY be requested. | MAY | Not support | | Not need to test |
| 83 | 4603 | The responder MAY respond with zero or more DNS server attributes. | MAY | Not support | | Not need to test |
| 83 | 4608 | o INTERNAL_IP4_NBNS, INTERNAL_IP6_NBNS - Specifies an address of a NetBios Name Server (WINS) within the network. Multiple NBNS servers MAY be requested. | MAY | Not support | | Not need to test |
| 83 | 4608 | The responder MAY respond with zero or more NBNS server attributes. | MAY | Not support | | Not need to test |
| 83 | 4613 | o INTERNAL_ADDRESS_EXPIRY - Specifies the number of seconds that the host can use the internal IP address. The host MUST renew the IP address before this expiry time. | MUST | Not support | | INTERNAL_ADD RESS_EXPIRY is "Not support" |
| 83 | 4613 | Only one of these attributes MAY be present in the reply. | MAY | Not support | | Not need to test |
| 83 | 4618 | o INTERNAL_IP4_DHCP, INTERNAL_IP6_DHCP - Instructs the host to send any internal DHCP requests to the address contained within the attribute. Multiple DHCP servers MAY be requested. | MAY | Not support | | Not need to test |
| 83 | 4618 | The responder MAY respond with zero or more DHCP server attributes. | MAY | Not support | | Not need to test |
| 83 | 4623 | o APPLICATION_VERSION - The version or application information of the IPsec host. This is a string of printable ASCII characters that is NOT null terminated. | | Not support | | Explanation |
| 83 | 4627 | o INTERNAL_IP4_SUBNET - The protected sub-networks that this edge-device protects. This attribute is made up of two fields: the first is an IP address and the second is a netmask. Multiple sub-networks MAY be requested. | MAY | Not support | | Not need to test |
| 83 | 4627 | The responder MAY respond with zero or more sub-network attributes. | MAY | Not support | | Not need to test |
| 83 | 4633 | o SUPPORTED_ATTRIBUTES - When used within a Request, this attribute MUST be zero-length and specifies a query to the responder to reply back with all of the attributes that it supports. | MUST | Not support | | SUPPORTED_AT TRIBUTES is "Not support" |
| 83 | 4633 | The response contains an attribute that contains a set of attribute identifiers each in 2 octets. The length divided by 2 (octets) would state the number of supported attributes contained in the response. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|------------------|------------------|--------|-----------------------|
| 84 | 4654 | o INTERNAL_IP6_SUBNET - The protected sub-networks that this edge-device protects. This attribute is made up of two fields: the first is a sixteen-octet IPv6 address and the second is a one-octet prefix-length as defined in [ADDRIPV6]. Multiple sub-networks MAY be requested. | MAY | Not support | | Not need to test |
| 84 | 4654 | The responder MAY respond with zero or more sub-network attributes. | MAY | Not support | | Not need to test |
| 84 | 4661 | Note that no recommendations are made in this document as to how an implementation actually figures out what information to send in a reply. That is, we do not recommend any specific method of an IRAS determining which DNS server should be returned to a requesting IRAC. | | Not support | | Explanation |
| 84 | 4667 | 3.16. Extensible Authentication Protocol (EAP) Payload | | | | |
| 84 | 4669 | The Extensible Authentication Protocol Payload, denoted EAP in this memo, allows IKE_SAs to be authenticated using the protocol defined in RFC 3748 [EAP] and subsequent extensions to that protocol. The full set of acceptable values for the payload is defined elsewhere, but a short summary of RFC 3748 is included here to make this document stand alone in the common cases. | | Not support | | Explanation |
| 84 | 4676 | <pre> 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 ! +++++ ! ~ EAP Message ~ ! +++++ </pre> <p>Figure 24: EAP Payload Format</p> | | Not support | | Explanation |
| 84 | 4688 | The payload type for an EAP Payload is forty eight (48). | | Not support | | Explanation |
| 84 | 4690 | <pre> 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 +++++ ! Code ! Identifier ! Length ! +++++ ! Type ! Type_Data... +++++ </pre> <p>Figure 25: EAP Message Format</p> | | Not support | | Explanation |
| 84 | 4700 | o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4). | | Not support | | Explanation |
| 85 | 4710 | o Identifier (1 octet) is used in PPP to distinguish replayed messages from repeated ones. Since in IKE, EAP runs over a reliable protocol, it serves no function here. In a response message, this octet MUST be set to match the identifier in the corresponding request. | MUST | Not support | | EAP is "Not support." |
| 85 | 4710 | In other messages, this field MAY be set to any value. | MAY | Not support | | EAP is "Not support." |
| 85 | 4717 | o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload. | MUST | Not support | | EAP is "Not support." |
| 85 | 4720 | o Type (1 octet) is present only if the Code field is Request (1) or Response (2). | | Not support | | Explanation |
| 85 | 4720 | For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. | MUST MUST NOT | Not support | | EAP is "Not support." |
| 85 | 4720 | In a Request (1) message, Type indicates the data being requested. | | Not support | | Explanation |
| 85 | 4720 | In a Response (2) message, Type MUST either be Nak or match the type of the data requested. The following types are defined in RFC 3748: | MUST | Not support | | EAP is "Not support." |
| 85 | 4728 | 1 Identity | | Not support | | Explanation |
| 85 | 4729 | 2 Notification | | Not support | | Explanation |
| 85 | 4730 | 3 Nak (Response Only) | | Not support | | Explanation |
| 85 | 4731 | 4 MD5-Challenge | | Not support | | Explanation |
| 85 | 4732 | 5 One-Time Password (OTP) | | Not support | | Explanation |
| 85 | 4733 | 6 Generic Token Card | | Not support | | Explanation |
| 85 | 4735 | o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP]. | | Not support | | Explanation |
| 85 | 4739 | Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. | SHOULD NOT | Not support | | EAP is "Not support." |
| 85 | 4739 | The initiator SHOULD , however, respond to such requests if it receives them. | SHOULD | Not support | | EAP is "Not support." |
| 85 | 4744 | 4. Conformance Requirements | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|---|
| 85 | 4746 | In order to assure that all implementations of IKEv2 can interoperate, there are "MUST support" requirements in addition to those listed elsewhere. Of course, IKEv2 is a security protocol, and one of its major functions is to allow only authorized parties to successfully complete establishment of SAs. So a particular implementation may be configured with any of a number of restrictions concerning algorithms and trusted authorities that will prevent universal interoperability. | | Not support | | Explanation |
| 86 | 4766 | IKEv2 is designed to permit minimal implementations that can interoperate with all compliant implementations. There are a series of optional features that can easily be ignored by a particular implementation if it does not support that feature. Those features include: | | Not support | | Explanation |
| 86 | 4772 | Ability to negotiate SAs through a NAT and tunnel the resulting ESP SA over UDP. | | Not support | | Explanation |
| 86 | 4775 | Ability to request (and respond to a request for) a temporary IP address on the remote end of a tunnel. | | Not support | | Explanation |
| 86 | 4778 | Ability to support various types of legacy authentication. | | Not support | | Explanation |
| 86 | 4780 | Ability to support window sizes greater than one. | | Not support | | Explanation |
| 86 | 4782 | Ability to establish multiple ESP and/or AH SAs within a single IKE_SA. | | Not support | | Explanation |
| 86 | 4785 | Ability to rekey SAs. | | Not support | | Explanation |
| 86 | 4787 | To assure interoperability, all implementations MUST be capable of parsing all payload types (if only to skip over them) and to ignore payload types that it does not support unless the critical bit is set in the payload header. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.4.3 SGW.R.1.1.4.3 |
| 86 | 4787 | If the critical bit is set in an unsupported payload header, all implementations MUST reject the messages containing those payloads. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.1.4.4 SGW.R.1.1.4.4 |
| 86 | 4794 | Every implementation MUST be capable of doing four-message IKE_SA_INIT and IKE_AUTH exchanges establishing two SAs (one for IKE, one for ESP and/or AH). | MUST | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.I.1.1.1.2 EN.I.1.1.1.3 EN.I.1.2.1.1 EN.I.2.1.1.1 EN.I.2.1.1.2 EN.R.1.1.1.1 EN.R.1.1.1.2 EN.R.1.1.1.3 EN.R.1.2.1.1 EN.R.1.3.1.1 EN.R.2.1.1.1 EN.R.2.1.1.2 SGW.I.1.1.1.1 SGW.I.1.1.1.2 SGW.I.1.1.1.3 SGW.I.1.2.1.1 SGW.I.2.1.1.1 SGW.I.2.1.1.2 SGW.R.1.1.1.1 SGW.R.1.1.1.2 SGW.R.1.1.1.3 SGW.R.1.2.1.1 SGW.R.1.3.1.1 SGW.R.2.1.1.1 SGW.R.2.1.1.2 |
| 86 | 4796 | Implementations MAY be initiate-only or respond-only if appropriate for their platform. | MAY | Not support | | Not need to test |
| 86 | 4797 | Every implementation MUST be capable of responding to an INFORMATIONAL exchange. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.3.1.1 SGW.R.1.3.1.1 |
| 86 | 4798 | , but a minimal implementation MAY respond to any INFORMATIONAL message with an empty INFORMATIONAL reply (note that within the context of an IKE_SA, an "empty" message consists of an IKE header followed by an Encrypted payload with no payloads contained in it). | MAY | Not support | | Not need to test |
| 86 | 4802 | A minimal implementation MAY support the CREATE_CHILD_SA exchange only in so far as to recognize requests and reject them with a Notify payload of type NO_ADDITIONAL_SAS. | MAY | Not support | | Not need to test |
| 86 | 4805 | A minimal implementation need not be able to initiate CREATE_CHILD_SA or INFORMATIONAL exchanges. When an SA expires (based on locally configured values of either lifetime or octets passed), and implementation MAY either try to renew it with a CREATE_CHILD_SA exchange or it MAY delete (close) the old SA and create a new one. | MAY MAY | Not support | | Not need to test |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|----------------|-----------------------|
| 86 | 4810 | If the responder rejects the CREATE_CHILD_SA request with a NO_ADDITIONAL_SAS notification, the implementation MUST be capable of instead closing the old SA and creating a new one. | MUST | Not support | | Difficult to test |
| 87 | 4822 | Implementations are not required to support requesting temporary IP addresses or responding to such requests. | | Not support | | Explanation |
| 87 | 4822 | If an implementation does support issuing such requests, it MUST include a CP payload in message 3 containing at least a field of type INTERNAL_IP4_ADDRESS or INTERNAL_IP6_ADDRESS. All other fields are optional. | MUST | ADVANCED | EN(initiator) | EN.I.2.1.2.1 |
| 87 | 4823 | If an implementation supports responding to such requests, it MUST parse the CP payload of type CFG_REQUEST in message 3 and recognize a field of type INTERNAL_IP4_ADDRESS or INTERNAL_IP6_ADDRESS. | MUST | ADVANCED | SGW(initiator) | SGW.I.2.1.2.1 |
| 87 | 4829 | If it supports leasing an address of the appropriate type, it MUST return a CP payload of type CFG_REPLY containing an address of the requested type. | MUST | ADVANCED | SGW(initiator) | SGW.I.2.1.2.1 |
| 87 | 4832 | The responder SHOULD include all of the other related attributes if it has them. | SHOULD | ADVANCED | SGW(initiator) | SGW.I.2.1.2.1 |
| 87 | 4835 | A minimal IPv4 responder implementation will ignore the contents of the CP payload except to determine that it includes an INTERNAL_IP4_ADDRESS attribute and will respond with the address and other related attributes regardless of whether the initiator requested them. | | Not support | | Explanation |
| 87 | 4841 | A minimal IPv4 initiator will generate a CP payload containing only an INTERNAL_IP4_ADDRESS attribute and will parse the response ignoring attributes it does not know how to use. The only attribute it MUST be able to process is INTERNAL_ADDRESS_EXPIRY, which it must use to bound the lifetime of the SA unless it successfully renews the lease before it expires. Minimal initiators need not be able to request lease renewals and minimal responders need not respond to them. | MUST | Not support | | IPv4 is out of scope |
| 87 | 4850 | For an implementation to be called conforming to this specification, it MUST be possible to configure it to accept the following: | MUST | Not support | | Explanation |
| 87 | 4853 | 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. | | Not support | | Explanation |
| 87 | 4857 | Shared key authentication where the ID passes is any of ID_KEY_ID, ID_FQDN, or ID_RFC822_ADDR. | | Not support | | Explanation |
| 87 | 4860 | Authentication where the responder is authenticated using PKIX Certificates and the initiator is authenticated using shared key authentication. | | Not support | | Explanation |
| 88 | 4878 | 5. Security Considerations | | | | |
| 88 | 4880 | While this protocol is designed to minimize disclosure of configuration information to unauthenticated peers, some such disclosure is unavoidable. One peer or the other must identify itself first and prove its identity first. To avoid probing, the initiator of an exchange is required to identify itself first, and usually is required to authenticate itself first. | | Not support | | Explanation |
| 88 | 4880 | The initiator can, however, learn that the responder supports IKE and what cryptographic protocols it supports. | | Not support | | Explanation |
| 88 | 4880 | The responder (or someone impersonating the responder) can probe the initiator not only for its identity, but using CERTREQ payloads may be able to determine what certificates the initiator is willing to use. | | Not support | | Explanation |
| 88 | 4892 | Use of EAP authentication changes the probing possibilities somewhat. When EAP authentication is used, the responder proves its identity before the initiator does, so an initiator that knew the name of a valid initiator could probe the responder for both its name and certificates. | | Not support | | Explanation |
| 88 | 4898 | Repeated rekeying using CREATE_CHILD_SA without additional Diffie-Hellman exchanges leaves all SAs vulnerable to cryptanalysis of a single key or overrun of either endpoint. Implementers should take note of this fact and set a limit on CREATE_CHILD_SA exchanges between exponentiations. This memo does not prescribe such a limit. | | Not support | | Explanation |
| 88 | 4904 | The strength of a key derived from a Diffie-Hellman exchange using any of the groups defined here depends on the inherent strength of the group, the size of the exponent used, and the entropy provided by the random number generator used. Due to these inputs, it is difficult to determine the strength of a key for any of the defined groups. Diffie-Hellman group number two, when used with a strong random number generator and an exponent no less than 200 bits, is common for use with 3DES. | | Not support | | Explanation |
| 88 | 4904 | Group five provides greater security than group two. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|-----------------------|
| 88 | 4904 | Group one is for historic purposes only and does not provide sufficient strength except for use with DES, which is also for historic use only. Implementations should make note of these estimates when establishing policy and negotiating security parameters. | | Not support | | Explanation |
| 88 | 4918 | Note that these limitations are on the Diffie-Hellman groups themselves. There is nothing in IKE that prohibits using stronger groups nor is there anything that will dilute the strength obtained from stronger groups (limited by the strength of the other algorithms negotiated including the prf function). In fact, the extensible framework of IKE encourages the definition of more groups; use of elliptical curve groups may greatly increase strength using much smaller numbers. | | Not support | | Explanation |
| 89 | 4934 | It is assumed that all Diffie-Hellman exponents are erased from memory after use. In particular, these exponents MUST NOT be derived from long-lived secrets like the seed to a pseudo-random generator that is not erased after use. | MUST NOT | Not support | | Internal process |
| 89 | 4939 | The strength of all keys is limited by the size of the output of the negotiated prf function. For this reason, a prf function whose output is less than 128 bits (e.g., 3DES-CBC) MUST NOT be used with this protocol. | MUST NOT | Not support | | Difficult to test |
| 89 | 4944 | The security of this protocol is critically dependent on the randomness of the randomly chosen parameters. These should be generated by a strong random or properly seeded pseudo-random source (see [RFC4086]). | | Not support | | Explanation |
| 89 | 4944 | Implementers should take care to ensure that use of random numbers for both keys and nonces is engineered in a fashion that does not undermine the security of the keys. | | Not support | | Explanation |
| 89 | 4951 | For information on the rationale of many of the cryptographic design choices in this protocol, see [SIGMA] and [SKEME]. Though the security of negotiated CHILD_SAs does not depend on the strength of the encryption and integrity protection negotiated in the IKE_SA, implementations MUST NOT negotiate NONE as the IKE integrity protection algorithm or ENCR_NULL as the IKE encryption algorithm. | MUST NOT | Not support | | Difficult to test |
| 89 | 4958 | When using pre-shared keys, a critical consideration is how to assure the randomness of these secrets. The strongest practice is to ensure that any pre-shared key contain as much randomness as the strongest key being negotiated. Deriving a shared secret from a password, name, or other low-entropy source is not secure. These sources are subject to dictionary and social engineering attacks, among others. | | Not support | | Explanation |
| 89 | 4965 | The NAT_DETECTION_*_IP notifications contain a hash of the addresses and ports in an attempt to hide internal IP addresses behind a NAT. Since the IPv4 address space is only 32 bits, and it is usually very sparse, it would be possible for an attacker to find out the internal address used behind the NAT box by trying all possible IP addresses and trying to find the matching hash. The port numbers are normally fixed to 500, and the SPIs can be extracted from the packet. This reduces the number of hash calculations to 2^32. With an educated guess of the use of private address space, the number of hash calculations is much smaller. Designers should therefore not assume that use of IKE will not leak internal address information. | | Not support | | Explanation |
| 89 | 4977 | When using an EAP authentication method that does not generate a shared key for protecting a subsequent AUTH payload, certain man-in-the-middle and server impersonation attacks are possible [EAPMITM]. These vulnerabilities occur when EAP is also used in protocols that are not protected with a secure tunnel. | | Not support | | Explanation |
| 90 | 4990 | Since EAP is a general-purpose authentication protocol, which is often used to provide single-signon facilities, a deployed IPsec solution that relies on an EAP authentication method that does not generate a shared key (also known as a non-key-generating EAP method) can become compromised due to the deployment of an entirely unrelated application that also happens to use the same non-key-generating EAP method, but in an unprotected fashion. Note that this vulnerability is not limited to just EAP, but can occur in other scenarios where an authentication infrastructure is reused. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--------|-----------------------|
| 90 | 4990 | For example, if the EAP mechanism used by IKEv2 utilizes a token authenticator, a man-in-the-middle attacker could impersonate the web server, intercept the token authentication exchange, and use it to initiate an IKEv2 connection. For this reason, use of non-key-generating EAP methods SHOULD be avoided where possible. | SHOULD | Not support | | EAP is "Not support." |
| 90 | 4990 | Where they are used, it is extremely important that all usages of these EAP methods SHOULD utilize a protected tunnel, where the initiator validates the responder's certificate before initiating the EAP exchange. | SHOULD | Not support | | EAP is "Not support." |
| 90 | 4990 | Implementers SHOULD describe the vulnerabilities of using non-key-generating EAP methods in the documentation of their implementations so that the administrators deploying IPsec solutions are aware of these dangers. | SHOULD | Not support | | EAP is "Not support." |
| 90 | 5011 | An implementation using EAP MUST also use a public-key-based authentication of the server to the client before the EAP exchange begins, even if the EAP method offers mutual authentication. This avoids having additional IKEv2 protocol variations and protects the EAP data from active attackers. | MUST | Not support | | EAP is "Not support." |
| 90 | 5017 | If the messages of IKEv2 are long enough that IP-level fragmentation is necessary, it is possible that attackers could prevent the exchange from completing by exhausting the reassembly buffers. The chances of this can be minimized by using the Hash and URL encodings instead of sending certificates (see section 3.6). Additional mitigations are discussed in [KPS03]. | | Not support | | Explanation |

Table 5-3 IKEv2 functions and its classifications for RFC4718

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|---|---------------------------------|---------------------------------|
| 4 | 174 | 1. Introduction | | | | |
| 4 | 176 | This document clarifies many areas of the IKEv2 specification that may be difficult to understand to developers not intimately familiar with the specification and its history. The clarifications in this document come from the discussion on the IPsec WG mailing list, from experience in interoperability testing, and from implementation issues that have been brought to the editors' attention. | | Not support | | Explanation |
| 4 | 183 | IKEv2/IPsec can be used for several different purposes, including IPsec-based remote access (sometimes called the "road warrior" case), site-to-site virtual private networks (VPNs), and host-to-host protection of application traffic. While this document attempts to consider all of these uses, the remote access scenario has perhaps received more attention here than the other uses. | | Not support | | Explanation |
| 4 | 190 | This document does not place any requirements on anyone and does not use [RFC2119] keywords such as "MUST" and "SHOULD", except in quotations from the original IKEv2 documents. The requirements are given in the IKEv2 specification [IKEv2] and IKEv2 cryptographic algorithms document [IKEv2ALG]. | | Not support | | Explanation |
| 4 | 196 | In this document, references to a numbered section (such as "Section 2.15") mean that section in [IKEv2]. References to mailing list messages or threads refer to the IPsec WG mailing list at ipsec@ietf.org. Archives of the mailing list can be found at < http://www.ietf.org/mail-archive/web/ipsec/index.html >. | | Not support | | Explanation |
| 4 | 202 | 2. Creating the IKE_SA | | | | |
| 4 | 204 | 2.1. SPI Values in IKE_SA_INIT Exchange | | | | |
| 4 | 206 | Normal IKE messages include the initiator's and responder's Security Parameter Indexes (SPIs), both of which are non-zero, in the IKE header. However, there are some corner cases where the IKEv2 specification is not fully consistent about what values should be used. | | Not support | | Explanation |
| 4 | 212 | First, Section 3.1 says that the Responder's SPI "...MUST NOT be zero in any other message" (than the first message of the IKE_SA_INIT exchange). However, the figure in Section 2.6 shows the second IKE_SA_INIT message as "HDR(A,0), N(COOKIE)", contradicting the text in 3.1. | | Not support | | Explanation |
| 4 | 218 | Since the responder's SPI identifies security-related state held by the responder, and in this case no state is created, sending a zero value seems reasonable. | | Not support | | untestable |
| 5 | 230 | Second, in addition to cookies, there are several other cases when the IKE_SA_INIT exchange does not result in the creation of an IKE_SA (for instance, INVALID_KEY_PAYLOAD or NO_PROPOSAL_CHOSEN). What responder SPI value should be used in the IKE_SA_INIT response in this case? | | Not support | | Explanation |
| 5 | 236 | Since the IKE_SA_INIT request always has a zero responder SPI, the value will not be actually used by the initiator. Thus, we think sending a zero value is correct also in this case. | | BASIC | EN(responder) SGW(responder) | EN.R.1.1.6.8 SGW.R.1.1.6.8 |
| 5 | 240 | If the responder sends a non-zero responder SPI, the initiator should not reject the response only for that reason. | | ADVANCED *Because DH#14 is ADVANCED group. | EN(initiator) SGW(initiator) | EN.I.1.1.6.11 SGW.I.1.1.6.11 |
| 5 | 240 | However, when retrying the IKE_SA_INIT request, the initiator will use a zero responder SPI, as described in Section 3.1: "Responder's SPI [...] This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) [...]". We believe the intent was to cover repeats of that message due to other reasons, such as INVALID_KEY_PAYLOAD, as well. | | ADVANCED *Because DH#14 is ADVANCED group. | EN(initiator) SGW(initiator) | EN.I.1.1.6.11 SGW.I.1.1.6.11 |
| 5 | 249 | (References: "INVALID_KEY_PAYLOAD and clarifications document" thread, Sep-Oct 2005.) | | Not support | | Explanation |
| 5 | 252 | 2.2. Message IDs for IKE_SA_INIT Messages | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|---|
| 5 | 254 | The Message ID for IKE_SA_INIT messages is always zero. This includes retries of the message due to responses such as COOKIE and INVALID_KE_PAYLOAD. | | | | EN.I.1.1.1.1 EN.I.1.1.2.1 EN.I.1.1.5.1 EN.I.1.1.6.11 EN.R.1.1.1.1 EN.R.1.2.1.1 EN.R.1.1.6.8 SGW.I.1.1.1.1 SGW.I.1.1.2.1 SGW.I.1.1.5.1 SGW.I.1.1.6.11 SGW.R.1.1.1.1 SGW.R.1.2.1.1 SGW.R.1.1.6.8 |
| 5 | 258 | This is because Message IDs are part of the IKE_SA state, and when the responder replies to IKE_SA_INIT request with N(COOKIE) or N(INVALID_KE_PAYLOAD), the responder does not allocate any state. | | Not support | | Explanation |
| 5 | 262 | (References: "Question about N(COOKIE) and N(INVALID_KE_PAYLOAD) combination" thread, Oct 2004. Tero Kivinen's mail "Comments of draft-eronen-ipsec-ikev2-clarifications-02.txt", 2005-04-05.) | | Not support | | Explanation |
| 5 | 266 | 2.3. Retransmissions of IKE_SA_INIT Requests | | | | |
| 5 | 268 | When a responder receives an IKE_SA_INIT request, it has to determine whether the packet is a retransmission belonging to an existing "half-open" IKE_SA (in which case the responder retransmits the same response), or a new request (in which case the responder creates a new IKE_SA and sends a fresh response). | | Not support | | Explanation |
| 5 | 274 | The specification does not describe in detail how this determination is done. In particular, it is not sufficient to use the initiator's SPI and/or IP address for this purpose: two different peers behind a single NAT could choose the same initiator SPI (and the probability of this happening is not necessarily small, since IKEv2 does not require SPIs to be chosen randomly). | | Not support | | Explanation |
| 6 | 287 | Instead, the responder should do the IKE_SA lookup using the whole packet or its hash (or at the minimum, the Ni payload which is always chosen randomly). | | Not support | | Explanation |
| 6 | 291 | For all other packets than IKE_SA_INIT requests, looking up right IKE_SA is of course done based on the recipient's SPI (either the initiator or responder SPI depending on the value of the Initiator bit in the IKE header). | | Not support | | Explanation |
| 6 | 296 | 2.4. Interaction of COOKIE and INVALID_KE_PAYLOAD | | | | |
| 6 | 298 | There are two common reasons why the initiator may have to retry the IKE_SA_INIT exchange: the responder requests a cookie or wants a different Diffie-Hellman group than was included in the KEi payload. Both of these cases are quite simple alone, but it is not totally obvious what happens when they occur at the same time, that is, the IKE_SA_INIT exchange is retried several times. | | Not support | | Explanation |
| 6 | 305 | The main question seems to be the following: if the initiator receives a cookie from the responder, should it include the cookie in only the next retry of the IKE_SA_INIT request, or in all subsequent retries as well? Section 3.10.1 says that: | | Not support | | Explanation |
| 6 | 310 | "This notification MUST be included in an IKE_SA_INIT request retry if a COOKIE notification was included in the initial response." | | Not support | | Explanation |
| 6 | 314 | This could be interpreted as saying that when a cookie is received in the initial response, it is included in all retries. On the other hand, Section 2.6 says that: | | Not support | | Explanation |
| 6 | 318 | "Initiators who receive such responses MUST retry the IKE_SA_INIT with a Notify payload of type COOKIE containing the responder supplied cookie data as the first payload and all other payloads unchanged." | | Not support | | Explanation |
| 6 | 323 | Including the same cookie in later retries makes sense only if the "all other payloads unchanged" restriction applies only to the first retry, but not to subsequent retries. | | Not support | | Explanation |
| 6 | 327 | It seems that both interpretations can peacefully coexist. If the initiator includes the cookie only in the next retry, one additional roundtrip may be needed in some cases: | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 8 | 414 | (References: "INVALID_KE_PAYLOAD and clarifications document" thread, Sep-Oct 2005.) | | Not support | | Explanation |
| 8 | 417 | 2.5. Invalid Cookies | | | | |
| 8 | 419 | There has been some confusion what should be done when an IKE_SA_INIT request containing an invalid cookie is received ("invalid" in the sense that its contents do not match the value expected by the responder). | | Not support | | Explanation |
| 8 | 424 | The correct action is to ignore the cookie and process the message as if no cookie had been included (usually this means sending a response containing a new cookie). | | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.1.5.2 SGW.1.1.5.2 |
| 8 | 425 | This is shown in Section 2.6 when it says "The responder in that case MAY reject the message by sending another response with a new cookie [...]". | | Not support | | Explanation |
| 8 | 430 | Other possible actions, such as ignoring the whole request (or even all requests from this IP address for some time), create strange failure modes even in the absence of any malicious attackers and do not provide any additional protection against DoS attacks. | | Not support | | Explanation |
| 8 | 435 | (References: "Invalid Cookie" thread, Sep-Oct 2005.) | | Not support | | Explanation |
| 9 | 454 | 3. Authentication | | | | |
| 9 | 456 | 3.1. Data Included in AUTH Payload Calculation | | | | |
| 9 | 458 | Section 2.15 describes how the AUTH payloads are calculated; this calculation involves values $\text{prf}(\text{SK}_{\text{pi}}, \text{IDi})$ and $\text{prf}(\text{SK}_{\text{pr}}, \text{IDr})$. The text describes the method in words, but does not give clear definitions of what is signed or MACed (i.e., protected with a message authentication code). | | Not support | | Explanation |
| 9 | 464 | The initiator's signed octets can be described as: | | Not support | | Explanation |
| 9 | 466 | 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 RestOfIDPayload RestOfInitIDPayload = IDType RESERVED InitIDData MACedIDForI = $\text{prf}(\text{SK}_{\text{pi}}, \text{RestOfInitIDPayload})$ | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.R.1.1.1.2 SGW.I.1.1.1.2 SGW.R.1.1.1.2 |
| 9 | 475 | The responder's signed octets can be described as: | | Not support | | Explanation |
| 9 | 477 | 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 RestOfIDPayload RestOfRespIDPayload = IDType RESERVED InitIDData MACedIDForR = $\text{prf}(\text{SK}_{\text{pr}}, \text{RestOfRespIDPayload})$ | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.R.1.1.1.2 SGW.I.1.1.1.2 SGW.R.1.1.1.2 |
| 9 | 486 | 3.2. Hash Function for RSA Signatures | | | | |
| 9 | 488 | Section 3.8 says that RSA digital signature is "Computed as specified in section 2.15 using an RSA private key over a PKCS#1 padded hash." | | Not support | | Explanation |
| 9 | 491 | Unlike IKEv1, IKEv2 does not negotiate a hash function for the IKE_SA. The algorithm for signatures is selected by the signing party who, in general, may not know beforehand what algorithms the verifying party supports. Furthermore, [IKEv2ALG] does not say what algorithms implementations are required or recommended to support. This clearly has a potential for causing interoperability problems, since authentication will fail if the signing party selects an algorithm that is not supported by the verifying party, or not acceptable according to the verifying party's policy. | | Not support | | Explanation |
| 10 | 510 | This document recommends that all implementations support SHA-1 and use SHA-1 as the default hash function when generating the signatures, unless there are good reasons (such as explicit manual configuration) to believe that the peer supports something else. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 10 | 515 | Note that hash function collision attacks are not important for the AUTH payloads, since they are not intended for third-party verification, and the data includes fresh nonces. See [HashUse] for more discussion about hash function attacks and IPsec. | | Not support | | Explanation |
| 10 | 520 | Another reasonable choice would be to use the hash function that was used by the CA when signing the peer certificate. | | Not support | | Explanation |
| 10 | 521 | However, this does not guarantee that the IKEv2 peer would be able to validate the AUTH payload, because the same code might not be used to validate certificate signatures and IKEv2 message signatures, and these two routines may support a different set of hash algorithms. | | Not support | | Explanation |
| 10 | 525 | The peer could be configured with a fingerprint of the certificate, or certificate validation could be performed by an external entity using [SCVP]. | | Not support | | Explanation |
| 10 | 528 | Furthermore, not all CERT payloads types include a signature, and the certificate could be signed with some algorithm other than RSA. | | Not support | | Explanation |
| 10 | 532 | Note that unlike IKEv1, IKEv2 uses the PKCS#1 v1.5 [PKCS1v20] signature encoding method (see next section for details), which includes the algorithm identifier for the hash algorithm. Thus, when the verifying party receives the AUTH payload it can at least determine which hash function was used. | | Not support | | Explanation |
| 10 | 538 | (References: Magnus Alstrom's mail "RE:", 2005-01-03. Pasi Eronen's reply, 2005-01-04. Tero Kivinen's reply, 2005-01-04. "First draft of IKEv2.1" thread, Dec 2005/Jan 2006.) | | Not support | | Explanation |
| 10 | 542 | 3.3. Encoding Method for RSA Signatures | | | | |
| 10 | 544 | Section 3.8 says that the RSA digital signature is "Computed as specified in section 2.15 using an RSA private key over a PKCS#1 padded hash." | | Not support | | Explanation |
| 10 | 548 | The PKCS#1 specification [PKCS1v21] defines two different encoding methods (ways of "padding the hash") for signatures. However, the Internet-Draft approved by the IESG had a reference to the older PKCS#1 v2.0 [PKCS1v20]. That version has only one encoding method for signatures (EMSA-PKCS1-v1_5), and thus there is no ambiguity. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 |
| 11 | 566 | Note that this encoding method is different from the encoding method used in IKEv1. If future revisions of IKEv2 provide support for other encoding methods (such as EMSA-PSS), they will be given new Auth Method numbers. | | Not support | | Explanation |
| 11 | 571 | (References: Pasi Eronen's mail "RE:", 2005-01-04.) | | Not support | | Explanation |
| 11 | 573 | 3.4. Identification Type for EAP | | | | |
| 11 | 575 | Section 3.5 defines several different types for identification payloads, including, e.g., ID_FQDN, ID_RFC822_ADDR, and ID_KEY_ID. EAP [EAP] does not mandate the use of any particular type of identifier, but often EAP is used with Network Access Identifiers (NAIs) defined in [NAI]. Although NAIs look a bit like email addresses (e.g., "joe@example.com"), the syntax is not exactly the same as the syntax of email address in [RFC822]. This raises the question of which identification type should be used. | | Not support | | Explanation |
| 11 | 584 | This document recommends that ID_RFC822_ADDR identification type is used for those NAIs that include the realm component. Therefore, responder implementations should not attempt to verify that the contents actually conform to the exact syntax given in [RFC822] or [RFC2822], but instead should accept any reasonable looking NAI. | | Not support | | Explanation |
| 11 | 590 | For NAIs that do not include the realm component, this document recommends using the ID_KEY_ID identification type. | | Not support | | Explanation |
| 11 | 593 | (References: "need your help on this IKEv2/i18n/EAP issue" and "IKEv2 identifier issue with EAP" threads, Aug 2004.) | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 11 | 596 | 3.5. Identity for Policy Lookups When Using EAP | | | | |
| 11 | 598 | When the initiator authentication uses EAP, it is possible that the contents of the IDi payload is used only for AAA routing purposes and selecting which EAP method to use. This value may be different from the identity authenticated by the EAP method (see [EAP], Sections 5.1 and 7.3). | | Not support | | Explanation |
| 11 | 604 | It is important that policy lookups and access control decisions use the actual authenticated identity. | | Not support | | Explanation |
| 11 | 605 | Often the EAP server is implemented in a separate AAA server that communicates with the IKEv2 responder using, e.g., RADIUS [RADEAP]. In this case, the authenticated identity has to be sent from the AAA server to the IKEv2 responder. | | Not support | | Explanation |
| 11 | 611 | (References: Pasi Eronen's mail "RE: Reauthentication in IKEv2", 2004-10-28. "Policy lookups" thread, Oct/Nov 2004. RFC 3748, Section 7.3.) | | Not support | | Explanation |
| 12 | 622 | 3.6. Certificate Encoding Types | | | | |
| 12 | 624 | Section 3.6 defines a total of twelve different certificate encoding types, and continues that "Specific syntax is for some of the certificate type codes above is not defined in this document." However, the text does not provide references to other documents that would contain information about the exact contents and use of those values. | | Not support | | Explanation |
| 12 | 631 | Without this information, it is not possible to develop interoperable implementations. Therefore, this document recommends that the following certificate encoding values should not be used before new specifications that specify their use are available. | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 |
| 12 | 636 | PKCS #7 wrapped X.509 certificate 1 PGP Certificate 2 DNS Signed Key 3 Kerberos Token 6 SPKI Certificate 9 | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 |
| 12 | 642 | This document recommends that most implementations should use only those values that are "MUST"/"SHOULD" requirements in [IKEv2]; i.e., "X.509 Certificate - Signature" (4), "Raw RSA Key" (11), "Hash and URL of X.509 certificate" (12), and "Hash and URL of X.509 bundle" (13). | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.10.3 EN.R.1.1.10.3 SGW.I.1.1.10.3 SGW.R.1.1.10.3 |
| 12 | 648 | Furthermore, Section 3.7 says that the "Certificate Encoding" field for the Certificate Request payload uses the same values as for Certificate payload. | | Not support | | Explanation |
| 12 | 650 | However, the contents of the "Certification Authority" field are defined only for X.509 certificates (presumably covering at least types 4, 10, 12, and 13). This document recommends that other values should not be used before new specifications that specify their use are available. | | Not support | | Explanation |
| 12 | 656 | The "Raw RSA Key" type needs one additional clarification. Section 3.6 says it contains "a PKCS #1 encoded RSA key". What this means is a DER-encoded RSAPublicKey structure from PKCS#1 [PKCS1v21]. | | Not support | | Explanation |
| 12 | 660 | 3.7. Shared Key Authentication and Fixed PRF Key Size | | | | |
| 12 | 662 | Section 2.15 says that "If the negotiated prf takes a fixed-size key, the shared secret MUST be of that fixed size". This statement is correct: the shared secret must be of the correct size. If it is not, it cannot be used; there is no padding, truncation, or other processing involved to force it to that correct size. | | Not support | | Explanation |
| 13 | 678 | This requirement means that it is difficult to use these pseudo-random functions (PRFs) with shared key authentication. The authors think this part of the specification was very poorly thought out, and using PRFs with a fixed key size is likely to result in interoperability problems. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--------|-----------------------|
| 13 | 682 | Thus, we recommend that such PRFs should not be used with shared key authentication. | | Not support | | Explanation |
| 13 | 683 | PRF_AES128_XCBC [RFC3664] originally used fixed key sizes; that RFC has been updated to handle variable key sizes in [RFC4434]. | | Not support | | Explanation |
| 13 | 687 | Note that Section 2.13 also contains text that is related to PRFs with fixed key size: "When the key for the prf function has fixed length, the data provided as a key is truncated or padded with zeros as necessary unless exceptional processing is explained following the formula". However, this text applies only to the prf+ construction, so it does not contradict the text in Section 2.15. | | Not support | | Explanation |
| 13 | 694 | (References: Paul Hoffman's mail "Re: ikev2-07: last nits", 2003-05-02. Hugo Krawczyk's reply, 2003-05-12. Thread "Question about PRFs with fixed size key", Jan 2005.) | | Not support | | Explanation |
| 13 | 698 | 3.8. EAP Authentication and Fixed PRF Key Size | | | | |
| 13 | 700 | As described in the previous section, PRFs with a fixed key size require a shared secret of exactly that size. This restriction applies also to EAP authentication. For instance, a PRF that requires a 128-bit key cannot be used with EAP since [EAP] specifies that the MSK is at least 512 bits long. | | Not support | | Explanation |
| 13 | 706 | (References: Thread "Question about PRFs with fixed size key", Jan 2005.) | | Not support | | Explanation |
| 13 | 709 | 3.9. Matching ID Payloads to Certificate Contents | | | | |
| 13 | 711 | In IKEv1, there was some confusion about whether or not the identities in certificates used to authenticate IKE were required to match the contents of the ID payloads. The PKI4IPsec Working Group produced the document [PKI4IPsec] which covers this topic in much more detail. | | Not support | | Explanation |
| 13 | 715 | However, Section 3.5 of [IKEv2] explicitly says that the ID payload "does not necessarily have to match anything in the CERT payload". | | Not support | | Explanation |
| 14 | 734 | 3.10. Message IDs for IKE_AUTH Messages | | | | |
| 14 | 736 | According to Section 2.2, "The IKE_SA initial setup messages will always be numbered 0 and 1." That is true when the IKE_AUTH exchange does not use EAP. When EAP is used, each pair of messages has their message numbers incremented. The first pair of AUTH messages will have an ID of 1, the second will be 2, and so on. | | Not support | | Explanation |
| 14 | 742 | (References: "Question about MsgID in AUTH exchange" thread, April 2005.) | | Not support | | Explanation |
| 14 | 745 | 4. Creating CHILD_SAs | | | | |
| 14 | 747 | 4.1. Creating SAs with the CREATE_CHILD_SA Exchange | | | | |
| 14 | 749 | Section 1.3's organization does not lead to clear understanding of what is needed in which environment. The section can be reorganized with subsections for each use of the CREATE_CHILD_SA exchange (creating child SAs, rekeying IKE SAs, and rekeying child SAs.) | | Not support | | Explanation |
| 14 | 754 | The new Section 1.3 with subsections and the above changes might look like the following. | | Not support | | Explanation |
| 14 | 757 | NEW-1.3 The CREATE_CHILD_SA Exchange | | | | |
| 14 | 759 | The CREATE_CHILD_SA Exchange is used to create new CHILD_SAs and to rekey both IKE_SAs and CHILD_SAs. This exchange consists of a single request/response pair, and some of its function was referred to as a phase 2 exchange in IKEv1. | | Not support | | Explanation |
| 14 | 762 | It MAY be initiated by either end of the IKE_SA after the initial exchanges are completed. | MAY | Not support | | Explanation |
| 14 | 766 | All messages following the initial exchange are cryptographically protected using the cryptographic algorithms and keys negotiated in the first two messages of the IKE exchange. These subsequent messages use the syntax of the Encrypted Payload described in section 3.14. All subsequent messages include an Encrypted Payload, even if they are referred to in the text as "empty". | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|---------------------------------|-------------------------------|
| 14 | 774 | The CREATE_CHILD_SA is used for rekeying IKE_SAs and CHILD_SAs. This section describes the first part of rekeying, the creation of new SAs; Section 2.8 covers the mechanics of rekeying, including moving traffic from old to new SAs and the deletion of the old SAs. The two sections must be read together to understand the entire process of rekeying. | | Not support | | Explanation |
| 15 | 790 | Either endpoint may initiate a CREATE_CHILD_SA exchange, so in this section the term initiator refers to the endpoint initiating this exchange. An implementation MAY refuse all CREATE_CHILD_SA requests within an IKE_SA. | MAY | Not support | | Explanation |
| 15 | 795 | The CREATE_CHILD_SA request MAY optionally contain a KE payload for an additional Diffie-Hellman exchange to enable stronger guarantees of forward secrecy for the CHILD_SA or IKE_SA. | | Not support | | Explanation |
| 15 | 797 | The keying material for the SA is a function of SK_d established during the establishment of the IKE_SA, the nonces exchanged during the CREATE_CHILD_SA exchange, and the Diffie-Hellman value (if KE payloads are included in the CREATE_CHILD_SA exchange). The details are described in sections 2.17 and 2.18. | | Not support | | Explanation |
| 15 | 804 | If a CREATE_CHILD_SA exchange includes a KEi payload, at least one of the SA offers MUST include the Diffie-Hellman group of the KEi. | MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.3.7 SGW.I.1.2.3.7 |
| 15 | 806 | The Diffie-Hellman group of the KEi MUST be an element of the group the initiator expects the responder to accept (additional Diffie-Hellman groups can be proposed). | MUST | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.3.7 SGW.I.1.2.3.7 |
| 15 | 808 | If the responder rejects the Diffie-Hellman group of the KEi payload, the responder MUST reject the request and indicate its preferred Diffie-Hellman group in the INVALID_KE_PAYLOAD Notification payload. | MUST | BASIC | EN(responder) SGW(responder) | EN.R.1.2.5.7 SGW.R.1.2.5.7 |
| 15 | 812 | In the case of such a rejection, the CREATE_CHILD_SA exchange fails, and the initiator SHOULD retry the exchange with a Diffie-Hellman proposal and KEi in the group that the responder gave in the INVALID_KE_PAYLOAD. | SHOULD | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.3.9 SGW.I.1.2.3.9 |
| 15 | 817 | NEW-1.3.1 Creating New CHILD_SAs with the CREATE_CHILD_SA Exchange | | | | |
| 15 | 819 | A CHILD_SA may be created by sending a CREATE_CHILD_SA request. The CREATE_CHILD_SA request for creating a new CHILD_SA is: | | Not support | | Explanation |
| 15 | 822 | Initiator ----- HDR, SK {[N+], SA, Ni, [KEi], TSi, TSr} --> Responder ----- | | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.2.1.1 SGW.I.1.2.1.1 |
| 15 | 827 | The initiator sends SA offer(s) in the SA payload, a nonce in the Ni payload, optionally a Diffie-Hellman value in the KEi payload, and the proposed traffic selectors for the proposed CHILD_SA in the TSi and TSr payloads. The request can also contain Notify payloads that specify additional details for the CHILD_SA: these include IPCOMP_SUPPORTED, USE_TRANSPORT_MODE, ESP_TFC_PADDING_NOT_SUPPORTED, and NON_FIRST_FRAGMENTS_ALSO. | | Not support | | Explanation |
| 16 | 846 | The CREATE_CHILD_SA response for creating a new CHILD_SA is: | | Not support | | Explanation |
| 16 | 848 | <- HDR, SK {[N+], SA, Nr, [KEr], TSi, TSr} | | ADVANCED | EN(responder) SGW(responder) | EN.R.1.2.1.1 SGW.R.1.2.1.1 |
| 16 | 851 | The responder replies with the accepted offer in an SA payload, and a Diffie-Hellman value in the KEr payload if KEi was included in the request and the selected cryptographic suite includes that group. As with the request, optional Notification payloads can specify additional details for the CHILD_SA. | | Not support | | Explanation |
| 16 | 857 | The traffic selectors for traffic to be sent on that SA are specified in the TS payloads in the response, which may be a subset of what the initiator of the CHILD_SA proposed. | | Not support | | Explanation |
| 16 | 861 | The text about rekeying SAs can be found in Section 5.1 of this document. | | Not support | | Explanation |
| 16 | 864 | 4.2. Creating an IKE_SA without a CHILD_SA | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 16 | 866 | CHILD_SAs can be created either by being piggybacked on the IKE_AUTH exchange, or using a separate CREATE_CHILD_SA exchange. The specification is not clear about what happens if creating the CHILD_SA during the IKE_AUTH exchange fails for some reason. | | Not support | | Explanation |
| 16 | 871 | Our recommendation in this situation is that the IKE_SA is created as usual. This is also in line with how the CREATE_CHILD_SA exchange works: a failure to create a CHILD_SA does not close the IKE_SA. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.12 EN.R.1.1.6.9 SGW.I.1.1.6.12 SGW.R.1.1.6.9 |
| 16 | 875 | The list of responses in the IKE_AUTH exchange that do not prevent an IKE_SA from being set up include at least the following: NO_PROPOSAL_CHOSEN, TS_UNACCEPTABLE, SINGLE_PAIR_REQUIRED, INTERNAL_ADDRESS_FAILURE, and FAILED_CP_REQUIRED. | | Not support | | Explanation |
| 16 | 880 | (References: "Questions about internal address" thread, April 2005.) | | Not support | | Explanation |
| 16 | 882 | 4.3. Diffie-Hellman for First CHILD_SA | | | | |
| 16 | 884 | Section 1.2 shows that IKE_AUTH messages do not contain KEi/KER or Ni/Nr payloads. This implies that the SA payload in IKE_AUTH exchange cannot contain Transform Type 4 (Diffie-Hellman Group) with any other value than NONE. | | Not support | | Explanation |
| 16 | 887 | Implementations should probably leave the transform out entirely in this case. | | BASIC | EN(initiator) SGW(initiator) | EN.I.1.1.1.2 SGW.I.1.1.1.2 |
| 17 | 902 | 4.4. Extended Sequence Numbers (ESN) Transform | | | | |
| 17 | 904 | The description of the ESN transform in Section 3.3 has been proved difficult to understand. The ESN transform has the following meaning: | | Not support | | Explanation |
| 17 | 908 | o A proposal containing one ESN transform with value 0 means "do not use extended sequence numbers". | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.R.1.1.1.2 SGW.I.1.1.1.2 SGW.R.1.1.1.2 |
| 17 | 908 | o A proposal containing one ESN transform with value 1 means "use extended sequence numbers". | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.2 EN.R.1.1.6.2 SGW.I.1.1.6.2 SGW.R.1.1.6.2 |
| 17 | 908 | o A proposal containing two ESN transforms with values 0 and 1 means "I support both normal and extended sequence numbers, you choose". (Obviously this case is only allowed in requests; the response will contain only one ESN transform.) | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.6.5 EN.R.1.1.6.5 SGW.I.1.1.6.5 SGW.R.1.1.6.5 |
| 17 | 919 | In most cases, the exchange initiator will include either the first or third alternative in its SA payload. The second alternative is rarely useful for the initiator: it means that using normal sequence numbers is not acceptable (so if the responder does not support ESNs, the exchange will fail with NO_PROPOSAL_CHOSEN). | | Not support | | Explanation |
| 17 | 925 | Note that including the ESN transform is mandatory when creating ESP/AH SAs (it was optional in earlier drafts of the IKEv2 specification). | | ADVANCED | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.R.1.1.1.2 SGW.I.1.1.1.2 SGW.R.1.1.1.2 |
| 17 | 929 | (References: "Technical change needed to IKEv2 before publication", STRAW POLL: Dealing with the ESN negotiation interoper issue in IKEv2" and "Results of straw poll regarding: IKEv2 interoperability issue" threads, March-April 2005.) | | Not support | | Explanation |
| 17 | 934 | 4.5. Negotiation of ESP_TFC_PADDING_NOT_SUPPORTED | | | | |
| 17 | 936 | The description of ESP_TFC_PADDING_NOT_SUPPORTED notification in Section 3.10.1 says that "This notification asserts that the sending endpoint will NOT accept packets that contain Flow Confidentiality (TFC) padding". | | Not support | | Explanation |
| 17 | 941 | However, the text does not say in which messages this notification should be included, or whether the scope of this notification is a single CHILD_SA or all CHILD_SAs of the peer. | | Not support | | Explanation |
| 17 | 945 | Our interpretation is that the scope is a single CHILD_SA, and thus this notification is included in messages containing an SA payload negotiating a CHILD_SA. If neither endpoint accepts TFC padding, this notification will be included in both the request proposing an SA and the response accepting it. If this notification is included in only one of the messages, TFC padding can still be sent in one direction. | | Not support | | Explanation |
| 18 | 961 | 4.6. Negotiation of NON_FIRST_FRAGMENTS_ALSO | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|-----------------------|
| 18 | 963 | NON_FIRST_FRAGMENTS_ALSO notification is described in Section 3.10.1 simply as "Used for fragmentation control. See [RFC4301] for Explanation." | | Not support | | Explanation |
| 18 | 967 | [RFC4301] says "Implementations that will transmit non-initial fragments on a tunnel mode SA that makes use of non-trivial port (or ICMP type/code or MH type) selectors MUST notify a peer via the IKE NOTIFY_NON_FIRST_FRAGMENTS_ALSO payload. The peer MUST reject this proposal if it will not accept non-initial fragments in this context. If an implementation does not successfully negotiate transmission of non-initial fragments for such an SA, it MUST NOT send such fragments over the SA." | | Not support | | Explanation |
| 18 | 976 | However, it is not clear exactly how the negotiation works. Our interpretation is that the negotiation works the same way as for IPCOMP_SUPPORTED and USE_TRANSPORT_MODE: sending non-first fragments is enabled only if NON_FIRST_FRAGMENTS_ALSO notification is included in both the request proposing an SA and the response accepting it. | | Not support | | Explanation |
| 18 | 981 | In other words, if the peer "rejects this proposal", it only omits NON_FIRST_FRAGMENTS_ALSO notification from the response, but does not reject the whole CHILD_SA creation. | | Not support | | Explanation |
| 18 | 985 | 4.7. Semantics of Complex Traffic Selector Payloads | | | | |
| 18 | 987 | As described in Section 3.13, the TSi/TSr payloads can include one or more individual traffic selectors. | | Not support | | Explanation |
| 18 | 990 | 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. | | Not support | | Explanation |
| 18 | 996 | For instance, the following traffic selectors: TSi = ((17, 100, 192.0.1.66-192.0.1.66), (17, 200, 192.0.1.66-192.0.1.66)) TSr = ((17, 300, 0.0.0.0-255.255.255.255), (17, 400, 0.0.0.0-255.255.255.255)) would match UDP packets from 192.0.1.66 to anywhere, with any of the four combinations of source/destination ports (100,300), (100,400), (200,300), and (200, 400). | | Not support | | Explanation |
| 19 | 1014 | This implies that some types of policies may require several CHILD_SA pairs. For instance, a policy matching only source/destination ports (100,300) and (200,400), but not the other two combinations, cannot be negotiated as a single CHILD_SA pair using IKEv2. | | Not support | | Explanation |
| 19 | 1019 | (References: "IKEv2 Traffic Selectors?" thread, Feb 2005.) | | Not support | | Explanation |
| 19 | 1021 | 4.8. ICMP Type/Code in Traffic Selector Payloads | | | | |
| 19 | 1023 | The traffic selector types 7 and 8 can also refer to ICMP type and code fields. As described in Section 3.13.1, "For the ICMP protocol, the two one-octet fields Type and Code are treated as a single 16-bit integer (with Type in the most significant eight bits and Code in the least significant eight bits) port number for the purposes of filtering based on this field." | | Not support | | Explanation |
| 19 | 1030 | Since ICMP packets do not have separate source and destination port fields, there is some room for confusion what exactly the four TS payloads (two in the request, two in the response, each containing both start and end port fields) should contain. | | Not support | | Explanation |
| 19 | 1035 | The answer to this question can be found from [RFC4301] Section 4.4.1.3. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|---------------------------------|--|
| 19 | 1038 | To give a concrete example, if a host at 192.0.1.234 wants to create a transport mode SA for sending "Destination Unreachable" packets (ICMPv4 type 3) to 192.0.2.155, but is not willing to receive them over this SA pair, the CREATE_CHILD_SA exchange would look like this: <pre> Initiator Responder ----- - HDR, SK { N(USE_TRANSPORT_MODE), SA, Ni, TSi(1, 0x0300-0x03FF, 192.0.1.234-192.0.1.234), TSr(1, 65535-0, 192.0.2.155-192.0.2.155) } --> <- HDR, SK { N(USE_TRANSPORT_MODE), SA, Nr, TSi(1, 0x0300-0x03FF, 192.0.1.234-192.0.1.234), TSr(1, 65535-0, 192.0.2.155-192.0.2.155) } </pre> | | Not support | | Explanation |
| 19 | 1053 | Since IKEv2 always creates IPsec SAs in pairs, two SAs are also created in this case, even though the second SA is never used for data traffic. | | Not support | | Explanation |
| 19 | 1057 | An exchange creating an SA pair that can be used both for sending and receiving "Destination Unreachable" places the same value in all the port: | | Not support | | Explanation |
| 20 | 1070 | <pre> Initiator Responder ----- - HDR, SK { N(USE_TRANSPORT_MODE), SA, Ni, TSi(1, 0x0300-0x03FF, 192.0.1.234-192.0.1.234), TSr(1, 0x0300-0x03FF, 192.0.2.155-192.0.2.155) } --> <- HDR, SK { N(USE_TRANSPORT_MODE), SA, Nr, TSi(1, 0x0300-0x03FF, 192.0.1.234-192.0.1.234), TSr(1, 0x0300-0x03FF, 192.0.2.155-192.0.2.155) } </pre> | | Not support | | Explanation |
| 20 | 1080 | (References: "ICMP and MH TSs for IKEv2" thread, Sep 2005.) | | Not support | | Explanation |
| 20 | 1082 | 4.9. Mobility Header in Traffic Selector Payloads | | | | |
| 20 | 1084 | Traffic selectors can use IP Protocol ID 135 to match the IPv6 mobility header [MIPv6]. However, the IKEv2 specification does not define how to represent the "MH Type" field in traffic selectors. | | Not support | | Explanation |
| 20 | 1088 | At some point, it was expected that this will be defined in a separate document later. | | Not support | | Explanation |
| 20 | 1089 | However, [RFC4301] says that "For IKE, the IPv6 mobility header message type (MH type) is placed in the most significant eight bits of the 16 bit local "port" selector". The direction semantics of TSi/TSr port fields are the same as for ICMP and are described in the previous section. | | Not support | | Explanation |
| 20 | 1095 | (References: Tero Kivinen's mail "Issue #86: Add IPv6 mobility header message type as selector", 2003-10-14. "ICMP and MH TSs for IKEv2" thread, Sep 2005.) | | Not support | | Explanation |
| 20 | 1099 | 4.10. Narrowing the Traffic Selectors | | | | |
| 20 | 1101 | Section 2.9 describes how traffic selectors are negotiated when creating a CHILD_SA. A more concise summary of the narrowing process is presented below. | | Not support | | Explanation |
| 20 | 1105 | o If the responder's policy does not allow any part of the traffic covered by TSi/TSr, it responds with TS_UNACCEPTABLE. | | BASIC | EN(responder) SGW(responder) | EN.R.1.1.7.2 SGW.R.1.1.7.2 |
| 20 | 1105 | o If the responder's policy allows the entire set of traffic covered by TSi/TSr, no narrowing is necessary, and the responder can return the same TSi/TSr values. | | BASIC | EN(responder) SGW(responder) | EN.R.1.1.1.2 SGW.R.1.1.1.2 |
| 20 | 1105 | o Otherwise, narrowing is needed. If the responder's policy allows all traffic covered by TSi[1]/TSr[1] (the first traffic selectors in TSi/TSr) but not entire TSi/TSr, the responder narrows to an acceptable subset of TSi/TSr that includes TSi[1]/TSr[1]. | | BASIC | EN(responder) SGW(responder) | EN.R.1.1.1.2 EN.R.1.1.1.3 SGW.R.1.1.7.2 SGW.R.1.1.7.3 |
| 20 | 1105 | o If the responder's policy does not allow all traffic covered by TSi[1]/TSr[1], but does allow some parts of TSi/TSr, it narrows to an acceptable subset of TSi/TSr. | | BASIC | EN(responder) SGW(responder) | EN.R.1.1.7.1 SGW.R.1.1.7.1 |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|-----------------------|
| 21 | 1130 | In the last two cases, there may be several subsets that are acceptable (but their union is not); in this case, the responder arbitrarily chooses one of them and includes ADDITIONAL_TS_POSSIBLE notification in the response. | | Not support | | Explanation |
| 21 | 1135 | 4.11. SINGLE PAIR REQUIRED | | | | |
| 21 | 1137 | The description of the SINGLE_PAIR_REQUIRED notify payload in Sections 2.9 and 3.10.1 is not fully consistent. | | Not support | | Explanation |
| 21 | 1140 | We do not attempt to describe this payload in this document either, since it is expected that most implementations will not have policies that require separate SAs for each address pair. | | Not support | | Explanation |
| 21 | 1144 | Thus, if only some part (or parts) of the TSi/TSr proposed by the initiator is (are) acceptable to the responder, most responders should simply narrow TSi/TSr to an acceptable subset (as described in the last two paragraphs of Section 2.9), rather than use SINGLE_PAIR_REQUIRED. | | Not support | | Explanation |
| 21 | 1150 | 4.12. Traffic Selectors Violating Own Policy | | | | |
| 21 | 1152 | Section 2.9 describes traffic selector negotiation in great detail. One aspect of this negotiation that may need some clarification is that when creating a new SA, the initiator should not propose traffic selectors that violate its own policy. If this rule is not followed, valid traffic may be dropped. | | Not support | | Explanation |
| 21 | 1158 | This is best illustrated by an example. Suppose that host A has a policy whose effect is that traffic to 192.0.1.66 is sent via host B encrypted using Advanced Encryption Standard (AES), and traffic to all other hosts in 192.0.1.0/24 is also sent via B, but encrypted using Triple Data Encryption Standard (3DES). Suppose also that host B accepts any combination of AES and 3DES. | | Not support | | Explanation |
| 21 | 1165 | If host A now proposes an SA that uses 3DES, and includes TSr containing (192.0.1.0-192.0.1.0.255), this will be accepted by host B. Now, host B can also use this SA to send traffic from 192.0.1.66, but those packets will be dropped by A since it requires the use of AES for those traffic. Even if host A creates a new SA only for 192.0.1.66 that uses AES, host B may freely continue to use the first SA for the traffic. In this situation, when proposing the SA, host A should have followed its own policy, and included a TSr containing ((192.0.1.0-192.0.1.65),(192.0.1.67-192.0.1.255)) instead. | | Not support | | Explanation |
| 22 | 1182 | In general, if (1) the initiator makes a proposal "for traffic X (TSi/TSr), do SA", and (2) for some subset X' of X, the initiator does not actually accept traffic X' with SA, and (3) the initiator would be willing to accept traffic X' with some SA' (!=SA), valid traffic can be unnecessarily dropped since the responder can apply either SA or SA' to traffic X'. | | Not support | | Explanation |
| 22 | 1189 | (References: "Question about "narrowing" ..." thread, Feb 2005. "IKEv2 needs a "policy usage mode"..." thread, Feb 2005. "IKEv2 Traffic Selectors?" thread, Feb 2005. "IKEv2 traffic selector negotiation examples", 2004-08-08.) | | Not support | | Explanation |
| 22 | 1194 | 4.13. Traffic Selector Authorization | | | | |
| 22 | 1196 | IKEv2 relies on information in the Peer Authorization Database (PAD) when determining what kind of IPsec SAs a peer is allowed to create. This process is described in [RFC4301] Section 4.4.3. When a peer requests the creation of an IPsec SA with some traffic selectors, the PAD must contain "Child SA Authorization Data" linking the identity authenticated by IKEv2 and the addresses permitted for traffic selectors. | | Not support | | Explanation |
| 22 | 1204 | For example, the PAD might be configured so that authenticated identity "sgw23.example.com" is allowed to create IPsec SAs for 192.0.2.0/24, meaning this security gateway is a valid "representative" for these addresses. Host-to-host IPsec requires similar entries, linking, for example, "fooserver4.example.com" with 192.0.1.66/32, meaning this identity a valid "owner" or "representative" of the address in question. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 24 | 1313 | The leading Notify payload of type REKEY_SA identifies the CHILD_SA being rekeyed, and it contains the SPI that the initiator expects in the headers of inbound packets. In addition, the initiator sends SA offer(s) in the SA payload, a nonce in the Ni payload, optionally a Diffie-Hellman value in the KEi payload, and the proposed traffic selectors in the TSi and TSr payloads. The request can also contain Notify payloads that specify additional details for the CHILD_SA. | | Not support | | Explanation |
| 24 | 1322 | The CREATE_CHILD_SA response for rekeying a CHILD_SA is: | | Not support | | Explanation |
| 24 | 1324 | <- HDR, SK {[N+], SA, Nr, [KEr], TSi, TSr} | | BASIC | EN(responder) SGW(responder) | EN.R.1.2.1.1 SGW.R.1.2.1.1 |
| 24 | 1327 | The responder replies with the accepted offer in an SA payload, and a Diffie-Hellman value in the KER payload if KEi was included in the request and the selected cryptographic suite includes that group. | | Not support | | Explanation |
| 24 | 1332 | The traffic selectors for traffic to be sent on that SA are specified in the TS payloads in the response, which may be a subset of what the initiator of the CHILD_SA proposed. | | Not support | | Explanation |
| 24 | 1336 | 5.2. Rekeying the IKE_SA vs. Reauthentication | | | | |
| 24 | 1338 | Rekeying the IKE_SA and reauthentication are different concepts in IKEv2. Rekeying the IKE_SA establishes new keys for the IKE_SA and resets the Message ID counters, but it does not authenticate the parties again (no AUTH or EAP payloads are involved). | | Not support | | Explanation |
| 25 | 1350 | While rekeying the IKE_SA may be important in some environments, reauthentication (the verification that the parties still have access to the long-term credentials) is often more important. | | Not support | | Explanation |
| 25 | 1354 | IKEv2 does not have any special support for reauthentication. Reauthentication is done by creating a new IKE_SA from scratch (using IKE_SA_INIT/IKE_AUTH exchanges, without any REKEY_SA notify payloads), creating new CHILD_SAs within the new IKE_SA (without REKEY_SA notify payloads), and finally deleting the old IKE_SA (which deletes the old CHILD_SAs as well). | | Not support | | Explanation |
| 25 | 1361 | This means that reauthentication also establishes new keys for the IKE_SA and CHILD_SAs. Therefore, while rekeying can be performed more often than reauthentication, the situation where "authentication lifetime" is shorter than "key lifetime" does not make sense. | | Not support | | Explanation |
| 25 | 1366 | While creation of a new IKE_SA can be initiated by either party (initiator or responder in the original IKE_SA), the use of EAP authentication and/or configuration payloads means in practice that reauthentication has to be initiated by the same party as the original IKE_SA. IKEv2 base specification does not allow the responder to request reauthentication in this case; however, this functionality is added in [ReAuth]. | | Not support | | Explanation |
| 25 | 1374 | (References: "Reauthentication in IKEv2" thread, Oct/Nov 2004.) | | Not support | | Explanation |
| 25 | 1376 | 5.3. SPIs When Rekeying the IKE_SA | | | | |
| 25 | 1378 | Section 2.18 says that "New initiator and responder SPIs are supplied in the SPI fields". This refers to the SPI fields in the Proposal structures inside the Security Association (SA) payloads, not the SPI fields in the IKE header. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.4.1 EN.R.1.2.6.1 SGW.I.1.2.4.1 SGW.R.1.2.6.1 |
| 25 | 1383 | (References: Tom Stiemerling's mail "Rekey IKE SA", 2005-01-24. Geoffrey Huang's reply, 2005-01-24.) | | Not support | | Explanation |
| 25 | 1386 | 5.4. SPI When Rekeying a CHILD_SA | | | | |
| 25 | 1388 | Section 3.10.1 says that in REKEY_SA notifications, "The SPI field identifies the SA being rekeyed." | | Not support | | Explanation |
| 25 | 1391 | Since CHILD_SAs always exist in pairs, there are two different SPIs. The SPI placed in the REKEY_SA notification is the SPI the exchange initiator would expect in inbound ESP or AH packets (just as in Delete payloads). | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 26 | 1406 | 5.5. Changing PRFs When Rekeying the IKE_SA | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|--|--|--|
| 26 | 1408 | When rekeying the IKE_SA, Section 2.18 says that "SKEYSEED for the new IKE_SA is computed using SK_d from the existing IKE_SA as follows: SKEYSEED = prf(SK_d (old), [g^ir (new)] Ni Nr)" | | Not support | | Explanation |
| 26 | 1414 | If the old and new IKE_SA selected a different PRF, it is not totally clear which PRF should be used. | | Not support | | Explanation |
| 26 | 1417 | Since the rekeying exchange belongs to the old IKE_SA, it is the old IKE_SA's PRF that is used. This also follows the principle that the same key (the old SK_d) should not be used with multiple cryptographic algorithms. | | ADVANCED *Because PRF_AES128_XCBC is ADVANCED algorithm | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.4.7 EN.R.1.2.6.7 SGW.I.1.2.4.7 SGW.R.1.2.6.7 |
| 26 | 1422 | Note that this may work poorly if the new IKE_SA's PRF has a fixed key size, since the output of the PRF may not be of the correct size. This supports our opinion earlier in the document that the use of PRFs with a fixed key size is a bad idea. | | Not support | | Explanation |
| 26 | 1427 | (References: "Changing PRFs when rekeying the IKE_SA" thread, June 2005.) | | Not support | | Explanation |
| 26 | 1430 | 5.6. Deleting vs. Closing SAs | | | | |
| 26 | 1432 | The IKEv2 specification talks about "closing" and "deleting" SAs, but it is not always clear what exactly is meant. However, other parts of the specification make it clear that when local state related to a CHILD_SA is removed, the SA must also be actively deleted with a Delete payload. | | Not support | | Explanation |
| 26 | 1438 | In particular, Section 2.4 says that "If an IKE endpoint chooses to delete CHILD_SAs, it MUST send Delete payloads to the other end notifying it of the deletion". Section 1.4 also explains that "ESP and AH SAs always exist in pairs, with one SA in each direction. When an SA is closed, both members of the pair MUST be closed." | | Not support | | Explanation |
| 26 | 1444 | 5.7. Deleting a CHILD_SA Pair | | | | |
| 26 | 1446 | Section 1.4 describes how to delete SA pairs using the Informational exchange: "To delete an SA, an INFORMATIONAL exchange with one or more delete payloads is sent listing the SPIs (as they would be expected in the headers of inbound packets) of the SAs to be deleted. The recipient MUST close the designated SAs." | | Not support | | Explanation |
| 27 | 1462 | The "one or more delete payloads" phrase has caused some confusion. You never send delete payloads for the two sides of an SA in a single message. If you have many SAs to delete at the same time (such as the nested example given in that paragraph), you include delete payloads for the inbound half of each SA in your Informational exchange. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.3.6 EN.R.1.1.3.5 SGW.I.1.1.3.6 SGW.R.1.1.3.5 |
| 27 | 1469 | 5.8. Deleting an IKE_SA | | | | |
| 27 | 1471 | Since IKE_SAs do not exist in pairs, it is not totally clear what the response message should contain when the request deleted the IKE_SA. | | Not support | | Explanation |
| 27 | 1474 | Since there is no information that needs to be sent to the other side (except that the request was received), an empty Informational response seems like the most logical choice. | | BASIC | EN(responder) SGW(responder) | EN.R.1.1.3.6 SGW.R.1.1.3.6 |
| 27 | 1478 | (References: "Question about delete IKE SA" thread, May 2005.) | | Not support | | Explanation |
| 27 | 1480 | 5.9. Who is the original initiator of IKE_SA | | | | |
| 27 | 1482 | In the IKEv2 document, "initiator" refers to the party who initiated the exchange being described, and "original initiator" refers to the party who initiated the whole IKE_SA. However, there is some potential for confusion because the IKE_SA can be rekeyed by either party. | | Not support | | Explanation |
| 27 | 1488 | To clear up this confusion, we propose that "original initiator" always refers to the party who initiated the exchange that resulted in the current IKE_SA. In other words, if the "original responder" starts rekeying the IKE_SA, that party becomes the "original initiator" of the new IKE_SA. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|-----------------------|
| 27 | 1494 | (References: Paul Hoffman's mail "Original initiator in IKEv2", 2005-04-21.) | | Not support | | Explanation |
| 27 | 1497 | 5.10. Comparing Nonces | | | | |
| 27 | 1499 | Section 2.8 about rekeying says that "If redundant SAs are created though such a collision, the SA created with the lowest of the four nonces used in the two exchanges SHOULD be closed by the endpoint that created it." | | Not support | | Explanation |
| 28 | 1518 | Here "lowest" uses an octet-by-octet (lexicographical) comparison (instead of, for instance, comparing the nonces as large integers). In other words, start by comparing the first octet; if they're equal, move to the next octet, and so on. If you reach the end of one nonce, that nonce is the lower one. | | Not support | | Explanation |
| 28 | 1524 | (References: "IKEv2 rekeying question" thread, July 2005.) | | Not support | | Explanation |
| 28 | 1526 | 5.11. Exchange Collisions | | | | |
| 28 | 1528 | Since IKEv2 exchanges can be initiated by both peers, it is possible that two exchanges affecting the same SA partly overlap. This can lead to a situation where the SA state information is temporarily not synchronized, and a peer can receive a request it cannot process in a normal fashion. Some of these corner cases are discussed in the specification, some are not. | | Not support | | Explanation |
| 28 | 1535 | Obviously, using a window size greater than one leads to infinitely more complex situations, especially if requests are processed out of order. In this section, we concentrate on problems that can arise even with window size 1. | | Not support | | Explanation |
| 28 | 1540 | (References: "IKEv2: invalid SPI in DELETE payload" thread, Dec 2005/ Jan 2006. "Problem with exchanges collisions" thread, Dec 2005.) | | Not support | | Explanation |
| 28 | 1543 | 5.11.1. Simultaneous CHILD_SA Close | | | | |
| 28 | 1545 | Probably the simplest case happens if both peers decide to close the same CHILD_SA pair at the same time: | | Not support | | Explanation |
| 28 | 1548 | <pre> Host A Host B ----- ----- send req1: D(SPIa) --> <-- send req2: D(SPIb) --> recv req1 <-- send resp1: 0 recv resp1 recv req2 send resp2: 0 --> --> recv resp2 </pre> | | Not support | | |
| 28 | 1559 | This case is described in Section 1.4 and is handled by omitting the Delete payloads from the response messages. | | Not support | | Explanation |
| 29 | 1574 | 5.11.2. Simultaneous IKE_SA Close | | | | |
| 29 | 1576 | Both peers can also decide to close the IKE_SA at the same time. The desired end result is obvious; however, in certain cases the final exchanges may not be fully completed. | | Not support | | Explanation |
| 29 | 1580 | <pre> Host A Host B ----- ----- send req1: D0 --> <-- send req2: D0 --> recv req1 </pre> | | Not support | | untestable |
| 29 | 1586 | At this point, host B should reply as usual (with empty Informal response), close the IKE_SA, and stop retransmitting req2. This is because once host A receives resp1, it may not be able to reply any longer. The situation is symmetric, so host A should behave the same way. | | Not support | | Explanation |
| 29 | 1592 | <pre> Host A Host B ----- ----- <-- send resp1: 0 send resp2: 0 </pre> | | Not support | | untestable |
| 29 | 1597 | Even if neither resp1 nor resp2 ever arrives, the end result is still correct: the IKE_SA is gone. The same happens if host A never receives req2. | | Not support | | Explanation |
| 29 | 1601 | 5.11.3. Simultaneous CHILD_SA Rekeying | | | | |
| 29 | 1603 | Another case that is described in the specification is simultaneous rekeying. Section 2.8 says | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|---------------------------------|---------------------------------|
| 35 | 1913 | It seems this situation is tricky to handle correctly. Our proposal is as follows: if a host receives a request to rekey the IKE_SA when it has CHILD_SAs in "half-open" state (currently being created or rekeyed), it should reply with NO_PROPOSAL_CHOSEN. If a host receives a request to create or rekey a CHILD_SA after it has started rekeying the IKE_SA, it should reply with NO_ADDITIONAL_SAS. | | Not support | | untestable |
| 35 | 1920 | The case where CHILD_SAs are being closed is even worse. Our recommendation is that if a host receives a request to rekey the IKE_SA when it has CHILD_SAs in "half-closed" state (currently being closed), it should reply with NO_PROPOSAL_CHOSEN. And if a host receives a request to close a CHILD_SA after it has started rekeying the IKE_SA, it should reply with an empty Informational response. This ensures that at least the other peer will eventually notice that the CHILD_SA is still in "half-closed" state and will start a new IKE_SA from scratch. | | Not support | | untestable |
| 35 | 1930 | 5.11.9. Closing and Rekeying the IKE_SA | | | | |
| 35 | 1932 | The final case considered in this section occurs if one peer decides to close the IKE_SA while the other peer tries to rekey it. | | Not support | | Explanation |
| 35 | 1935 | <pre> Host A Host B ----- ----- send req1: SA(...,SPIa1,...),Ni1 --> <-- send req2: D0 --> recv req1 recv req2 <-- </pre> | | Not support | | untestable |
| 35 | 1942 | At this point, host B should probably reply with NO_PROPOSAL_CHOSEN, and host A should reply as usual, close the IKE_SA, and stop retransmitting req1. | | Not support | | Explanation |
| 35 | 1946 | <pre> <-- send resp1: N(NO_PROPOSAL_CHOSEN) send resp2: 0 </pre> | | Not support | | untestable |
| 35 | 1949 | If host A wants to continue communication with B, it can now start a new IKE_SA. | | Not support | | Explanation |
| 35 | 1952 | 5.11.10. Summary | | | | |
| 35 | 1954 | If a host receives a request to rekey: | | Not support | | Explanation |
| 35 | 1956 | o a CHILD_SA pair that the host is currently trying to close: reply with NO_PROPOSAL_CHOSEN. | | Not support | | untestable |
| 36 | 1966 | o a CHILD_SA pair that the host is currently rekeying: reply as usual, but prepare to close redundant SAs later based on the nonces. | | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.6.3 SGW.I.1.2.6.3 |
| 36 | 1970 | o a CHILD_SA pair that does not exist: reply with NO_PROPOSAL_CHOSEN. | | Not support | | untestable |
| 36 | 1973 | o the IKE_SA, and the host is currently rekeying the IKE_SA: reply as usual, but prepare to close redundant SAs and move inherited CHILD_SAs later based on the nonces. | | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.6.5 SGW.I.1.2.6.5 |
| 36 | 1977 | o the IKE_SA, and the host is currently creating, rekeying, or closing a CHILD_SA: reply with NO_PROPOSAL_CHOSEN. | | Not support | | untestable |
| 36 | 1980 | o the IKE_SA, and the host is currently trying to close the IKE_SA: reply with NO_PROPOSAL_CHOSEN. | | Not support | | untestable |
| 36 | 1983 | If a host receives a request to close: | | Not support | Explanation | |
| 36 | 1985 | o a CHILD_SA pair that the host is currently trying to close: reply without Delete payloads. | | Not support | | untestable |
| 36 | 1988 | o a CHILD_SA pair that the host is currently rekeying: reply as usual, with Delete payload. | | BASIC | EN(initiator) SGW(initiator) | EN.I.1.2.6.16 SGW.I.1.2.6.16 |
| 36 | 1991 | o a CHILD_SA pair that does not exist: reply without Delete payloads. | | Not support | | untestable |
| 36 | 1994 | o the IKE_SA, and the host is currently rekeying the IKE_SA: reply as usual, and forget about our own rekeying request. | | Not support | | untestable |
| 36 | 1997 | o the IKE_SA, and the host is currently trying to close the IKE_SA: reply as usual, and forget about our own close request. | | Not support | | untestable |
| 36 | 2000 | If a host receives a request to create or rekey a CHILD_SA when it is currently rekeying the IKE_SA: reply with NO_ADDITIONAL_SAS. | | Not support | | untestable |
| 36 | 2003 | If a host receives a request to delete a CHILD_SA when it is currently rekeying the IKE_SA: reply without Delete payloads. | | Not support | | untestable |
| 36 | 2006 | 5.12. Diffie-Hellman and Rekeying the IKE_SA | | | | |
| 36 | 2008 | There has been some confusion whether doing a new Diffie-Hellman exchange is mandatory when the IKE_SA is rekeyed. | | Not support | | Explanation |
| 36 | 2011 | It seems that this case is allowed by the IKEv2 specification. Section 2.18 shows the Diffie-Hellman term (g ^{ir}) in brackets. Section 3.3.3 does not contradict this when it says that including the D-H transform is mandatory: although including the transform is mandatory, it can contain the value "NONE". | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|---------------------------------|-------------------------------|
| 37 | 2025 | However, having the option to skip the Diffie-Hellman exchange when rekeying the IKE_SA does not add useful functionality to the protocol. The main purpose of rekeying the IKE_SA is to ensure that the compromise of old keying material does not provide information about the current keys, or vice versa. This requires performing the Diffie-Hellman exchange when rekeying. Furthermore, it is likely that this option would have been removed from the protocol as unnecessary complexity had it been discussed earlier. | | Not support | | Explanation |
| 37 | 2034 | Given this, we recommend that implementations should have a hard-coded policy that requires performing a new Diffie-Hellman exchange when rekeying the IKE_SA. In other words, the initiator should not propose the value "NONE" for the D-H transform. | | ADVANCED | EN(initiator) SGW(initiator) | EN.I.1.2.4.2 SGW.I.1.2.4.2 |
| 37 | 2034 | and the responder should not accept such a proposal. This policy also implies that a successful exchange rekeying the IKE_SA always includes the KEi/KEr payloads. | | Not support | | untestable |
| 37 | 2042 | (References: "Rekeying IKE_SAs with the CREATE_CHILD_SA exchange" thread, Oct 2005. "Comments of draft:eronen-ipse-ikev2-clarifications-02.txt" thread, Apr 2005.) | | Not support | | Explanation |
| 37 | 2046 | 6. Configuration Payloads | | | | |
| 37 | 2048 | 6.1. Assigning IP Addresses | | | | |
| 37 | 2050 | Section 2.9 talks about traffic selector negotiation and mentions that "In support of the scenario described in section 1.1.3, an initiator may request that the responder assign an IP address and tell the initiator what it is." | | Not support | | Explanation |
| 37 | 2055 | This sentence is correct, but its placement is slightly confusing. IKEv2 does allow the initiator to request assignment of an IP address from the responder, but this is done using configuration payloads, not traffic selector payloads. An address in a TSi payload in a response does not mean that the responder has assigned that address to the initiator; it only means that if packets matching these traffic selectors are sent by the initiator, IPsec processing can be performed as agreed for this SA. The TSi payload itself does not give the initiator permission to configure the initiator's TCP/IP stack with the address and use it as its source address. | | Not support | | Explanation |
| 37 | 2066 | In other words, IKEv2 does not have two different mechanisms for assigning addresses, but only one: configuration payloads. In the scenario described in Section 1.1.3, both configuration and traffic selector payloads are usually included in the same message, and they often contain the same information in the response message (see Section 6.3 of this document for some examples). However, their semantics are still different. | | Not support | | Explanation |
| 38 | 2082 | 6.2. Requesting any INTERNAL_IP4/IP6_ADDRESS | | | | |
| 38 | 2084 | When describing the INTERNAL_IP4/IP6_ADDRESS attributes, Section 3.15.1 says that "In a request message, the address specified is a requested address (or zero if no specific address is requested)". The question here is whether "zero" means an address "0.0.0.0" or a zero-length string. | | Not support | | Explanation |
| 38 | 2090 | Earlier, the same section also says that "If an attribute in the CFG_REQUEST Configuration Payload is not zero-length, it is taken as a suggestion for that attribute". Also, the table of configuration attributes shows that the length of INTERNAL_IP4_ADDRESS is either "0 or 4 octets", and likewise, INTERNAL_IP6_ADDRESS is either "0 or 17 octets". | | Not support | | Explanation |
| 38 | 2097 | Thus, if the client does not request a specific address, it includes a zero-length INTERNAL_IP4/IP6_ADDRESS attribute, not an attribute containing an all-zeroes address. The example in 2.19 is thus incorrect, since it shows the attribute as "INTERNAL_ADDRESS(0.0.0.0)". | | ADVANCED | SGW(responder) | SGW.R.2.1.2.1 |
| 38 | 2103 | However, since the value is only a suggestion, implementations are recommended to ignore suggestions they do not accept; or in other words, to treat the same way a zero-length INTERNAL_IP4_ADDRESS, "0.0.0.0", and any other addresses the implementation does not recognize as a reasonable suggestion. | | Not support | | Explanation |
| 38 | 2109 | 6.3. INTERNAL_IP4_SUBNET/INTERNAL_IP6_SUBNET | | | | |
| 38 | 2111 | Section 3.15.1 describes the INTERNAL_IP4_SUBNET as "The protected sub-networks that this edge-device protects. This attribute is made up of two fields: the first is an IP address and the second is a netmask. Multiple sub-networks MAY be requested. The responder MAY respond with zero or more sub-network attributes." INTERNAL_IP6_SUBNET is defined in a similar manner. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--------|-----------------------|
| 38 | 2118 | This raises two questions: first, since this information is usually included in the TSr payload, what functionality does this attribute add? And second, what does this attribute mean in CFG_REQUESTs? | | Not support | | Explanation |
| 38 | 2122 | For the first question, there seem to be two sensible interpretations. Clearly TSr (in IKE_AUTH or CREATE_CHILD_SA response) indicates which subnets are accessible through the SA that was just created. | | Not support | | Explanation |
| 39 | 2134 | The first interpretation of the INTERNAL_IP4/6_SUBNET attributes is that they indicate additional subnets that can be reached through this gateway, but need a separate SA. According to this interpretation, the INTERNAL_IP4/6_SUBNET attributes are useful mainly when they contain addresses not included in TSr. | | Not support | | Explanation |
| 39 | 2140 | The second interpretation is that the INTERNAL_IP4/6_SUBNET attributes express the gateway's policy about what traffic should be sent through the gateway. The client can choose whether other traffic (covered by TSr, but not in INTERNAL_IP4/6_SUBNET) is sent through the gateway or directly to the destination. According to this interpretation, the attributes are useful mainly when TSr contains addresses not included in the INTERNAL_IP4/6_SUBNET attributes. | | Not support | | Explanation |
| 39 | 2149 | It turns out that these two interpretations are not incompatible, but rather two sides of the same principle: traffic to the addresses listed in the INTERNAL_IP4/6_SUBNET attributes should be sent via this gateway. If there are no existing IPsec SAs whose traffic selectors cover the address in question, new SAs have to be created. | | Not support | | Explanation |
| 39 | 2155 | A couple of examples are given below. For instance, if there are two subnets, 192.0.1.0/26 and 192.0.2.0/24, and the client's request contains the following: | | Not support | | Explanation |
| 39 | 2159 | CP(CFG_REQUEST) = INTERNAL_IP4_ADDRESS0 TSi = (0, 0-65535, 0.0.0.0-255.255.255.255) TSr = (0, 0-65535, 0.0.0.0-255.255.255.255) | | Not support | | Explanation |
| 39 | 2164 | Then a valid response could be the following (in which TSr and INTERNAL_IP4_SUBNET contain the same information): | | Not support | | Explanation |
| 39 | 2167 | CP(CFG_REPLY) = INTERNAL_IP4_ADDRESS(192.0.1.234) INTERNAL_IP4_SUBNET(192.0.1.0/255.255.255.192) INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0) TSi = (0, 0-65535, 192.0.1.234-192.0.1.234) TSr = ((0, 0-65535, 192.0.1.0-192.0.1.63), (0, 0-65535, 192.0.2.0-192.0.2.255)) | | Not support | | Explanation |
| 39 | 2175 | In these cases, the INTERNAL_IP4_SUBNET does not really carry any useful information. Another possible reply would have been this: | | Not support | | Explanation |
| 39 | 2178 | CP(CFG_REPLY) = INTERNAL_IP4_ADDRESS(192.0.1.234) INTERNAL_IP4_SUBNET(192.0.1.0/255.255.255.192) INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0) | | Not support | | Explanation |
| 40 | 2190 | TSi = (0, 0-65535, 192.0.1.234-192.0.1.234) TSr = (0, 0-65535, 0.0.0.0-255.255.255.255) | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|-----------------------|
| 40 | 2193 | This would mean that the client can send all its traffic through the gateway, but the gateway does not mind if the client sends traffic not included by INTERNAL_IP4_SUBNET directly to the destination (without going through the gateway). | | Not support | | Explanation |
| 40 | 2198 | A different situation arises if the gateway has a policy that requires the traffic for the two subnets to be carried in separate SAs. Then a response like this would indicate to the client that if it wants access to the second subnet, it needs to create a separate SA: | | Not support | | Explanation |
| 40 | 2204 | CP(CFG_REPLY) = INTERNAL_IP4_ADDRESS(192.0.1.234) INTERNAL_IP4_SUBNET(192.0.1.0/255.255.255.192) INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0) TSi = (0, 0-65535, 192.0.1.234-192.0.1.234) TSr = (0, 0-65535, 192.0.1.0-192.0.1.63) | | Not support | | Explanation |
| 40 | 2211 | INTERNAL_IP4_SUBNET can also be useful if the client's TSr included only part of the address space. For instance, if the client requests the following: | | Not support | | Explanation |
| 40 | 2215 | CP(CFG_REQUEST) = INTERNAL_IP4_ADDRESS0 TSi = (0, 0-65535, 0.0.0.0-255.255.255.255) TSr = (0, 0-65535, 192.0.2.155-192.0.2.155) | | Not support | | Explanation |
| 40 | 2220 | Then the gateway's reply could be this: | | Not support | | Explanation |
| 40 | 2222 | CP(CFG_REPLY) = INTERNAL_IP4_ADDRESS(192.0.1.234) INTERNAL_IP4_SUBNET(192.0.1.0/255.255.255.192) INTERNAL_IP4_SUBNET(192.0.2.0/255.255.255.0) TSi = (0, 0-65535, 192.0.1.234-192.0.1.234) TSr = (0, 0-65535, 192.0.2.155-192.0.2.155) | | Not support | | Explanation |
| 40 | 2229 | It is less clear what the attributes mean in CFG_REQUESTs, and whether other lengths than zero make sense in this situation (but for INTERNAL_IP6_SUBNET, zero length is not allowed at all!). This document recommends that implementations should not include INTERNAL_IP4_SUBNET or INTERNAL_IP6_SUBNET attributes in CFG_REQUESTs. | | Not support | | Explanation |
| 40 | 2236 | For the IPv4 case, this document recommends using only netmasks consisting of some amount of "1" bits followed by "0" bits; for instance, "255.0.255.0" would not be a valid netmask for INTERNAL_IP4_SUBNET. | | Not support | | Explanation |
| 41 | 2249 | It is also worthwhile to note that the contents of the INTERNAL_IP4/6_SUBNET attributes do not imply link boundaries. For instance, a gateway providing access to a large company intranet using addresses from the 10.0.0.0/8 block can send a single INTERNAL_IP4_SUBNET attribute (10.0.0.0/255.0.0.0) even if the intranet has hundreds of routers and separate links. | | Not support | | Explanation |
| 41 | 2256 | (References: Tero Kivinen's mail "Intent of couple of attributes in Configuration Payload in IKEv2?", 2004-11-19. Srinivasa Rao Addepalli's mail "INTERNAL_IP4_SUBNET and INTERNAL_IP6_SUBNET in IKEv2", 2004-09-10. Yoav Nir's mail "Re: New I-D: IKEv2 Clarifications and Implementation Guidelines", 2005-02-07. "Clarifications open issue: INTERNAL_IP4_SUBNET/NETMASK" thread, April 2005.) | | Not support | | Explanation |
| 41 | 2264 | 6.4. INTERNAL_IP4_NETMASK | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--------|-----------------------|
| 41 | 2266 | Section 3.15.1 defines the INTERNAL_IP4_NETMASK attribute and says that "The internal network's netmask. Only one netmask is allowed in the request and reply messages (e.g., 255.255.255.0) and it MUST be used only with an INTERNAL_IP4_ADDRESS attribute". | | Not support | | Explanation |
| 41 | 2271 | However, it is not clear what exactly this attribute means, as the concept of "netmask" is not very well defined for point-to-point links (unlike multi-access links, where it means "you can reach hosts inside this netmask directly using layer 2, instead of sending packets via a router"). Even if the operating system's TCP/IP stack requires a netmask to be configured, for point-to-point links it could be just set to 255.255.255.255. So, why is this information sent in IKEv2? | | Not support | | Explanation |
| 41 | 2280 | One possible interpretation would be that the host is given a whole block of IP addresses instead of a single address. This is also what Framed-IP-Netmask does in [RADIUS], the IPCP "subnet mask" extension does in PPP [IPCPSubnet], and the prefix length in the IPv6 Framed-IPv6-Prefix attribute does in [RADIUS6]. However, nothing in the specification supports this interpretation, and discussions on the IPsec WG mailing list have confirmed it was not intended. Section 3.15.1 also says that multiple addresses are assigned using multiple INTERNAL_IP4/6_ADDRESS attributes. | | Not support | | Explanation |
| 41 | 2290 | Currently, this document's interpretation is the following: INTERNAL_IP4_NETMASK in a CFG_REPLY means roughly the same thing as INTERNAL_IP4_SUBNET containing the same information ("send traffic to these addresses through me"), but also implies a link boundary. For instance, the client could use its own address and the netmask to calculate the broadcast address of the link. (Whether the gateway will actually deliver broadcast packets to other VPN clients and/or other nodes connected to this link is another matter.) | | Not support | | Explanation |
| 42 | 2307 | An empty INTERNAL_IP4_NETMASK attribute can be included in a CFG_REQUEST to request this information (although the gateway can send the information even when not requested). However, it seems that non-empty values for this attribute do not make sense in CFG_REQUESTS. | | Not support | | Explanation |
| 42 | 2313 | Fortunately, Section 4 clearly says that a minimal implementation does not need to include or understand the INTERNAL_IP4_NETMASK attribute, and thus this document recommends that implementations should not use the INTERNAL_IP4_NETMASK attribute or assume that the other peer supports it. | | Not support | | Explanation |
| 42 | 2319 | (References: Charlie Kaufman's mail "RE: Proposed Last Call based revisions to IKEv2", 2004-05-27. Email discussion with Tero Kivinen, Jan 2005. Yoav Nir's mail "Re: New I-D: IKEv2 Clarifications and Implementation Guidelines", 2005-02-07. "Clarifications open issue: INTERNAL_IP4_SUBNET/NETMASK" thread, April 2005.) | | Not support | | Explanation |
| 42 | 2325 | 6.5. Configuration Payloads for IPv6 | | | | |
| 42 | 2327 | IKEv2 also defines configuration payloads for IPv6. However, they are based on the corresponding IPv4 payloads and do not fully follow the "normal IPv6 way of doing things". | | Not support | | Explanation |
| 42 | 2331 | A client can be assigned an IPv6 address using the INTERNAL_IP6_ADDRESS configuration payload. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--------|-----------------------|
| 42 | 2335 | A minimal exchange could look like this: CP(CFG_REQUEST) = INTERNAL_IP6_ADDRESS0 INTERNAL_IP6_DNS0 TSi = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF) TSr = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF) CP(CFG_REPLY) = INTERNAL_IP6_ADDRESS(2001:DB8:0:1:2:3:4:5/64) INTERNAL_IP6_DNS(2001:DB8:99:88:77:66:55:44) TSi = (0, 0-65535, 2001:DB8:0:1:2:3:4:5 - 2001:DB8:0:1:2:3:4:5) TSr = (0, 0-65535, :: - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF) | | Not support | | Explanation |
| 42 | 2347 | In particular, IPv6 stateless autoconfiguration or router advertisement messages are not used; neither is neighbor discovery. | | Not support | | Explanation |
| 43 | 2358 | The client can also send a non-empty INTERNAL_IP6_ADDRESS attribute in the CFG_REQUEST to request a specific address or interface identifier. The gateway first checks if the specified address is acceptable, and if it is, returns that one. If the address was not acceptable, the gateway will attempt to use the interface identifier with some other prefix; if even that fails, the gateway will select another interface identifier. | | Not support | | Explanation |
| 43 | 2366 | The INTERNAL_IP6_ADDRESS attribute also contains a prefix length field. When used in a CFG_REPLY, this corresponds to the INTERNAL_IP4_NETMASK attribute in the IPv4 case (and indeed, was called INTERNAL_IP6_NETMASK in earlier versions of the IKEv2 draft). See the previous section for more details. | | Not support | | Explanation |
| 43 | 2372 | While this approach to configuring IPv6 addresses is reasonably simple, it has some limitations: IPsec tunnels configured using IKEv2 are not fully-featured "interfaces" in the IPv6 addressing architecture [IPv6Addr] sense. In particular, they do not necessarily have link-local addresses, and this may complicate the use of protocols that assume them, such as [MLDv2]. (Whether they are called "interfaces" in some particular operating system is a different issue.) | | Not support | | Explanation |
| 43 | 2381 | (References: "VPN remote host configuration IPv6 ?" thread, May 2004. "Clarifications open issue: INTERNAL_IP4_SUBNET/NETMASK" thread, April 2005.) | | Not support | | Explanation |
| 43 | 2385 | 6.6. INTERNAL_IP6_NBNS | | | | |
| 43 | 2387 | Section 3.15.1 defines the INTERNAL_IP6_NBNS attribute for sending the IPv6 address of NetBIOS name servers. | | Not support | | Explanation |
| 43 | 2390 | However, NetBIOS is not defined for IPv6 and probably never will be. Thus, this attribute most likely does not make much sense. | | Not support | | Explanation |
| 43 | 2393 | (Pointed out by Bernard Aboba in the IP Configuration Security (ICOS) BoF at IETF62.) | | Not support | | Explanation |
| 43 | 2396 | 6.7. INTERNAL_ADDRESS_EXPIRY | | | | |
| 43 | 2398 | Section 3.15.1 defines the INTERNAL_ADDRESS_EXPIRY attribute as "Specifies the number of seconds that the host can use the internal IP address. The host MUST renew the IP address before this expiry time. Only one of these attributes MAY be present in the reply." | | Not support | | Explanation |
| 43 | 2403 | Expiry times and explicit renewals are primarily useful in environments like DHCP, where the server cannot reliably know when the client has gone away. However, in IKEv2 this is known, and the gateway can simply free the address when the IKE_SA is deleted. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|---------------|-----------------------|
| 44 | 2417 | Also, Section 4 says that supporting renewals is not mandatory. Given that this functionality is usually not needed, we recommend that gateways should not send the INTERNAL_ADDRESS_EXPIRY attribute. (And since this attribute does not seem to make much sense for CFG_REQUESTs, clients should not send it either.) | | Not support | | Explanation |
| 44 | 2423 | Note that according to Section 4, clients are required to understand INTERNAL_ADDRESS_EXPIRY if they receive it. A minimum implementation would use the value to limit the lifetime of the IKE_SA. | | Not support | | Explanation |
| 44 | 2427 | (References: Tero Kivinen's mail "Comments of draft-eronen-ipsec-ikev2-clarifications-02.txt", 2005-04-05. "Questions about internal address" thread, April 2005.) | | Not support | | Explanation |
| 44 | 2431 | 6.8. Address Assignment Failures | | | | |
| 44 | 2433 | If the responder encounters an error while attempting to assign an IP address to the initiator, it responds with an INTERNAL_ADDRESS_FAILURE notification as described in Section 3.10.1. However, there are some more complex error cases. | | Not support | | Explanation |
| 44 | 2438 | First, if the responder does not support configuration payloads at all, it can simply ignore all configuration payloads. This type of implementation never sends INTERNAL_ADDRESS_FAILURE notifications. | | Not support | | Explanation |
| 44 | 2441 | If the initiator requires the assignment of an IP address, it will treat a response without CFG_REPLY as an error. | | ADVANCED | EN(initiator) | EN.I.2.1.2.4 |
| 44 | 2444 | A second case is where the responder does support configuration payloads, but only for particular type of addresses (IPv4 or IPv6). Section 4 says that "A minimal IPv4 responder implementation will ignore the contents of the CP payload except to determine that it includes an INTERNAL_IP4_ADDRESS attribute". If, for instance, the initiator includes both INTERNAL_IP4_ADDRESS and INTERNAL_IP6_ADDRESS in the CFG_REQUEST, an IPv4-only responder can thus simply ignore the IPv6 part and process the IPv4 request as usual. | | Not support | | Explanation |
| 44 | 2453 | A third case is where the initiator requests multiple addresses of a type that the responder supports: what should happen if some (but not all) of the requests fail? It seems that an optimistic approach would be the best one here: if the responder is able to assign at least one address, it replies with those; it sends INTERNAL_ADDRESS_FAILURE only if no addresses can be assigned. | | Not support | | Explanation |
| 44 | 2460 | (References: "ikev2 and internal_ipv4_addr" thread, June 2005.) | | Not support | | Explanation |
| 45 | 2470 | 7. Miscellaneous Issues | | | | |
| 45 | 2472 | 7.1. Matching ID_IPV4_ADDR and ID_IPV6_ADDR | | | | |
| 45 | 2474 | When using the ID_IPV4_ADDR/ID_IPV6_ADDR identity types in IDi/IDr payloads, IKEv2 does not require this address to match anything in the TSi/TSr payloads. For example, in a site-to-site VPN between two security gateways, the gateways could authenticate each other as ID_IPV4_ADDR(192.0.1.1) and ID_IPV4_ADDR(192.0.2.1), and then create a CHILD_SA for protecting traffic between 192.0.1.55/32 (a host behind the first security gateway) and 192.0.2.240/28 (a network behind the second security gateway). The authenticated identities (IDi/IDr) are linked to the authorized traffic selectors (TSi/TSr) using "Child SA Authorization Data" in the Peer Authorization Database (PAD). | | Not support | | Explanation |
| 45 | 2486 | Furthermore, IKEv2 does not require that the addresses in ID_IPV4_ADDR/ID_IPV6_ADDR match the address in the IP header of the IKE packets. However, other specifications may place additional requirements regarding this. For example, [PKI4IPsec] requires that implementation must be capable of comparing the addresses in the ID_IPV4_ADDR/ID_IPV6_ADDR with the addresses in the IP header of the IKE packets, and this comparison must be enabled by default. | | Not support | | Explanation |
| 45 | 2494 | (References: "Identities types IP address, FQDN/user FQDN and DN and its usage in preshared key authentication" thread, Jan 2005. "Matching ID_IPV4_ADDR and ID_IPV6_ADDR" thread, May 2006.) | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--------|-----------------------|
| 45 | 2498 | 7.2. Relationship of IKEv2 to RFC 4301 | | | | |
| 45 | 2500 | The IKEv2 specification refers to [RFC4301], but it never clearly defines the exact relationship. | | Not support | | Explanation |
| 45 | 2503 | However, there are some requirements in the specification that make it clear that IKEv2 requires [RFC4301]. In other words, an implementation that does IPsec processing strictly according to [RFC2401] cannot be compliant with the IKEv2 specification. | | Not support | | Explanation |
| 45 | 2508 | One such example can be found in Section 2.24: "Specifically, tunnel encapsulators and decapsulators for all tunnel-mode SAs created by IKEv2 [...] MUST implement the tunnel encapsulation and decapsulation processing specified in [RFC4301] to prevent discarding of ECN congestion indications." | | Not support | | Explanation |
| 45 | 2514 | Nevertheless, the changes required to existing [RFC2401] implementations are not very large, especially since supporting many of the new features (such as Extended Sequence Numbers) is optional. | | Not support | | Explanation |
| 46 | 2526 | 7.3. Reducing the Window Size | | | | |
| 46 | 2528 | In IKEv2, the window size is assumed to be a (possibly configurable) property of a particular implementation and is not related to congestion control (unlike the window size in TCP, for instance). | | Not support | | Explanation |
| 46 | 2532 | In particular, it is not defined what the responder should do when it receives a SET_WINDOW_SIZE notification containing a smaller value than is currently in effect. Thus, there is currently no way to reduce the window size of an existing IKE_SA. However, when rekeying an IKE_SA, the new IKE_SA starts with window size 1 until it is explicitly increased by sending a new SET_WINDOW_SIZE notification. | | Not support | | Explanation |
| 46 | 2539 | (References: Tero Kivinen's mail "Comments of draft-eronen-ipsec-ikev2-clarifications-02.txt", 2005-04-05.) | | Not support | | Explanation |
| 46 | 2542 | 7.4. Minimum Size of Nonces | | | | |
| 46 | 2544 | Section 2.10 says that "Nonces used in IKEv2 MUST be randomly chosen, MUST be at least 128 bits in size, and MUST be at least half the key size of the negotiated prf." | | Not support | | Explanation |
| 46 | 2548 | However, the initiator chooses the nonce before the outcome of the negotiation is known. In this case, the nonce has to be long enough for all the PRFs being proposed. | | Not support | | Explanation |
| 46 | 2552 | 7.5. Initial Zero Octets on Port 4500 | | | | |
| 46 | 2554 | It is not clear whether a peer sending an IKE_SA_INIT request on port 4500 should include the initial four zero octets. Section 2.23 talks about how to upgrade to tunneling over port 4500 after message 2, but it does not say what to do if message 1 is sent on port 4500. | | Not support | | Explanation |
| 46 | 2559 | IKE MUST listen on port 4500 as well as port 500. [...] The IKE initiator MUST check these payloads if present and if they do not match the addresses in the outer packet MUST tunnel all future IKE and ESP packets associated with this IKE_SA over UDP port 4500. To tunnel IKE packets over UDP port 4500, the IKE header has four octets of zero prepended and the result immediately follows the UDP header. [...] | | Not support | | Explanation |
| 46 | 2559 | The very beginning of Section 2 says "... though IKE messages may also be received on UDP port 4500 with a slightly different format (see section 2.23)." | | Not support | | Explanation |
| 47 | 2586 | That "slightly different format" is only described in discussing what to do after changing to port 4500. However, [RFC3948] shows clearly the format has the initial zeros even for initiators on port 4500. Furthermore, without the initial zeros, the processing engine cannot determine whether the packet is an IKE packet or an ESP packet. | | Not support | | Explanation |
| 47 | 2592 | Thus, all packets sent on port 4500 need the four-zero prefix; otherwise, the receiver won't know how to handle them. | | Not support | | Explanation |
| 47 | 2595 | 7.6. Destination Port for NAT Traversal | | | | |
| 47 | 2597 | Section 2.23 says that "an IPsec endpoint that discovers a NAT between it and its correspondent MUST send all subsequent traffic to and from port 4500". | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 47 | 2601 | This sentence is misleading. The peer "outside" the NAT uses source port 4500 for the traffic it sends, but the destination port is, of course, taken from packets sent by the peer behind the NAT. This port number is usually dynamically allocated by the NAT. | | Not support | | Explanation |
| 47 | 2606 | 7.7. SPI Values for Messages outside an IKE_SA | | | | |
| 47 | 2608 | The IKEv2 specification is not quite clear what SPI values should be used in the IKE header for the small number of notifications that are allowed to be sent outside an IKE_SA. Note that such notifications are explicitly not Informational exchanges; Section 1.5 makes it clear that these are one-way messages that must not be responded to. | | Not support | | Explanation |
| 47 | 2614 | There are two cases when such a one-way notification can be sent: INVALID_IKE_SPI and INVALID_SPI. | | Not support | | Explanation |
| 47 | 2617 | In case of INVALID_IKE_SPI, the message sent is a response message, and Section 2.21 says that "If a response is sent, the response MUST be sent to the IP address and port from whence it came with the same IKE SPIs and the Message ID copied." | | Not support | | Explanation |
| 47 | 2622 | In case of INVALID_SPI, however, there are no IKE SPI values that would be meaningful to the recipient of such a notification. Also, the message sent is now an INFORMATIONAL request. A strict interpretation of the specification would require the sender to invent garbage values for the SPI fields. However, we think this was not the intention, and using zero values is acceptable. | | Not support | | Explanation |
| 47 | 2629 | (References: "INVALID_IKE_SPI" thread, June 2005.) | | Not support | | Explanation |
| 48 | 2638 | 7.8. Protocol ID/SPI Fields in Notify Payloads | | | | |
| 48 | 2640 | Section 3.10 says that the Protocol ID field in Notify payloads "For notifications that do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt". However, the specification does not clearly say which notifications are related to existing SAs and which are not. | | Not support | | Explanation |
| 48 | 2646 | Since the main purpose of the Protocol ID field is to specify the type of the SPI, our interpretation is that the Protocol ID field should be non-zero only when the SPI field is non-empty. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 48 | 2650 | There are currently only two notifications where this is the case: INVALID_SELECTORS and REKEY_SA. | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.1.1 EN.R.1.2.1.1 SGW.I.1.2.1.1 SGW.R.1.2.1.1 |
| 48 | 2653 | 7.9. Which message should contain INITIAL_CONTACT | | | | |
| 48 | 2655 | The description of the INITIAL_CONTACT notification in Section 3.10.1 says that "This notification asserts that this IKE_SA is the only IKE_SA currently active between the authenticated identities". However, neither Section 2.4 nor 3.10.1 says in which message this payload should be placed. | | Not support | | Explanation |
| 48 | 2661 | The general agreement is that INITIAL_CONTACT is best communicated in the first IKE_AUTH request, not as a separate exchange afterwards. | | Not support | | untestable |
| 48 | 2664 | (References: "Clarifying the use of INITIAL_CONTACT in IKEv2" thread, April 2005. "Initial Contact messages" thread, December 2004. "IKEv2 and Initial Contact" thread, September 2004 and April 2005.) | | Not support | | Explanation |
| 48 | 2668 | 7.10. Alignment of Payloads | | | | |
| 48 | 2670 | Many IKEv2 payloads contain fields marked as "RESERVED", mostly because IKEv1 had them, and partly because they make the pictures easier to draw. In particular, payloads in IKEv2 are not, in general, aligned to 4-octet boundaries. (Note that payloads were not aligned to 4-octet boundaries in IKEv1 either.) | | Not support | | Explanation |
| 48 | 2676 | (References: "IKEv2: potential 4-byte alignment problem" thread, June 2004.) | | Not support | | Explanation |
| 48 | 2679 | 7.11. Key Length Transform Attribute | | | | |
| 48 | 2681 | Section 3.3.5 says that "The only algorithms defined in this document that accept attributes are the AES based encryption, integrity, and pseudo-random functions, which require a single attribute specifying key width." | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--------|-----------------------|
| 49 | 2694 | This is incorrect. The AES-based integrity and pseudo-random functions defined in [IKEv2] always use a 128-bit key. In fact, there are currently no integrity or PRF algorithms that use the key length attribute (and we recommend that they should not be defined in the future either). | | Not support | | Explanation |
| 49 | 2700 | For encryption algorithms, the situation is slightly more complex since there are three different types of algorithms: | | Not support | | Explanation |
| 49 | 2703 | o The key length attribute is never used with algorithms that use a fixed length key, such as DES and IDEA. | | Not support | | Explanation |
| 49 | 2706 | o The key length attribute is always included for the currently defined AES-based algorithms (Cipher Block Chaining (CBC), Counter (CTR) Mode, Counter with CBC-MAC (CCM), and Galois/Counter Mode (GCM)). Omitting the key length attribute is not allowed; if the proposal does not contain it, the proposal has to be rejected. | | Not support | | Explanation |
| 49 | 2712 | o For other algorithms, the key length attribute can be included but is not mandatory. These algorithms include, e.g., RC5, CAST, and BLOWFISH. If the key length attribute is not included, the default value specified in [RFC2451] is used. | | Not support | | Explanation |
| 49 | 2717 | 7.12. IPsec IANA Considerations | | | | |
| 49 | 2719 | There are currently three different IANA registry files that contain important numbers for IPsec: ikev2-registry, isakmp-registry, and ipsec-registry. Implementers should note that IKEv2 may use numbers different from those of IKEv1 for a particular algorithm. | | Not support | | Explanation |
| 49 | 2724 | For instance, an encryption algorithm can have up to three different numbers: the IKEv2 "Transform Type 1" identifier in ikev2-registry, the IKEv1 phase 1 "Encryption Algorithm" identifier in ipsec-registry, and the IKEv1 phase 2 "IPSEC ESP Transform Identifier" in isakmp-registry. Although some algorithms have the same number in all three registries, the registries are not identical. | | Not support | | Explanation |
| 49 | 2731 | Similarly, an integrity algorithm can have at least the IKEv2 "Transform Type 3" identifier in ikev2-registry, the IKEv1 phase 2 "IPSEC AH Transform Identifier" in isakmp-registry, and the IKEv1 phase 2 ESP "Authentication Algorithm Security Association Attribute" identifier in isakmp-registry. And there is also the IKEv1 phase 1 "Hash Algorithm" list in ipsec-registry. | | Not support | | Explanation |
| 49 | 2738 | This issue needs special care also when writing a specification for how a new algorithm is used with IPsec. | | Not support | | Explanation |
| 50 | 2750 | 7.13. Combining ESP and AH | | | | |
| 50 | 2752 | The IKEv2 specification contains some misleading text about how ESP and AH can be combined. | | Not support | | Explanation |
| 50 | 2755 | IKEv2 is based on [RFC4301], which does not include "SA bundles" that were part of [RFC2401]. While a single packet can go through IPsec processing multiple times, each of these passes uses a separate SA, and the passes are coordinated by the forwarding tables. In IKEv2, each of these SAs has to be created using a separate CREATE_CHILD_SA exchange. Thus, the text in Section 2.7 about a single proposal containing both ESP and AH is incorrect. | | Not support | | Explanation |
| 50 | 2763 | Moreover, the combination of ESP and AH (between the same endpoints) had already become largely obsolete in 1998 when RFC 2406 was published. Our recommendation is that IKEv2 implementations should not support this combination, and implementers should not assume the combination can be made to work in an interoperable manner. | | Not support | | Explanation |
| 50 | 2769 | (References: "Rekeying SA bundles" thread, Oct 2005.) | | Not support | | Explanation |
| 50 | 2771 | 8. Implementation Mistakes | | | | |
| 50 | 2773 | Some implementers at the early IKEv2 bakeoffs didn't do everything correctly. This may seem like an obvious statement, but it is probably useful to list a few things that were clear in the document, but that some implementers didn't do. All of these things caused interoperability problems. | | Not support | | Explanation |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|---|-----------------|------------------|--|--|
| 50 | 2779 | o Some implementations continued to send traffic on a CHILD_SA after it was rekeyed, even after receiving an DELETE payload. | | Not support | | Explanation |
| 50 | 2782 | o After rekeying an IKE_SA, some implementations did not reset their message counters to zero. One set the counter to 2, another did not reset the counter at all. | | Not support | | Explanation |
| 50 | 2786 | o Some implementations could only handle a single pair of traffic selectors or would only process the first pair in the proposal. | | Not support | | Explanation |
| 50 | 2789 | o Some implementations responded to a delete request by sending an empty INFORMATIONAL response and then initiated their own INFORMATIONAL exchange with the pair of SAs to delete. | | Not support | | Explanation |
| 50 | 2793 | o Although this did not happen at the bakeoff, from the discussion there, it is clear that some people had not implemented message window sizes correctly. Some implementations might have sent messages that did not fit into the responder's message windows, and some implementations may not have torn down an SA if they did not ever receive a message that they know they should have. | | Not support | | Explanation |
| 54 | 2974 | Appendix A. Exchanges and Payloads | | | | |
| 54 | 2976 | This appendix contains a short summary of the IKEv2 exchanges, and what payloads can appear in which message. This appendix is purely informative; if it disagrees with the body of this document or the IKEv2 specification, the other text is considered correct. | | Not support | | Explanation |
| 54 | 2981 | Vendor-ID (V) payloads may be included in any place in any message. This sequence shows what are, in our opinion, the most logical places for them. | | Not support | | Explanation |
| 54 | 2985 | The specification does not say which messages can contain N(SET_WINDOW_SIZE). It can possibly be included in any message, but it is not yet shown below. | | Not support | | Explanation |
| 54 | 2989 | A.1. IKE_SA_INIT Exchange | | | | |
| 54 | 2991 | request --> [N(COOKIE)], SA, KE, Ni, [N(NAT_DETECTION_SOURCE_IP)+, N(NAT_DETECTION_DESTINATION_IP)], [V+] | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.1 EN.R.1.1.1.1 SGW.I.1.1.1.1 SGW.R.1.1.1.1 |
| 54 | 2997 | normal response <-- SA, KE, Nr, (no cookie) [N(NAT_DETECTION_SOURCE_IP), N(NAT_DETECTION_DESTINATION_IP)], [[N(HTTP_CERT_LOOKUP_SUPPORTED)], CERTREQ+], [V+] | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.R.1.1.1.1 SGW.I.1.1.1.2 SGW.R.1.1.1.1 |
| 54 | 3003 | A.2. IKE_AUTH Exchange without EAP | | | | |
| 54 | 3005 | request --> IDi, [CERT+], [N(INITIAL_CONTACT)], [[N(HTTP_CERT_LOOKUP_SUPPORTED)], CERTREQ+], [IDr], AUTH, [CP(CFG_REQUEST)], [N(IPCOMP_SUPPORTED)+], [N(USE_TRANSPORT_MODE)], [N(ESP_TFC_PADDING_NOT_SUPPORTED)], [N(NON_FIRST_FRAGMENTS_ALSO)], SA, TSi, TSr, [V+] | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.2 EN.R.1.1.1.2 SGW.I.1.1.1.2 SGW.R.1.1.1.2 |
| 55 | 3030 | response <-- IDr, [CERT+], AUTH, [CP(CFG_REPLY)], [N(IPCOMP_SUPPORTED)], [N(USE_TRANSPORT_MODE)], [N(ESP_TFC_PADDING_NOT_SUPPORTED)], [N(NON_FIRST_FRAGMENTS_ALSO)], SA, TSi, TSr, [N(ADDITIONAL_TS_POSSIBLE)], [V+] | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.1.1.3 EN.R.1.1.1.2 SGW.I.1.1.1.3 SGW.R.1.1.1.2 |
| 55 | 3041 | A.3. IKE_AUTH Exchange with EAP | | | | |

| page | line | sentence | RFC requirement | test requirement | target | test number or reason |
|------|------|--|-----------------|------------------|--|--|
| 55 | 3043 | first request --> IDi, [N(HTTP_CERT_LOOKUP_SUPPORTED)], CERTREQ+], [IDr], [CP(CFG_REQUEST)], [N(IPCOMP_SUPPORTED)+], [N(USE_TRANSPORT_MODE)], [N(ESP_TFC_PADDING_NOT_SUPPORTED)], [N(NON_FIRST_FRAGMENTS_ALSO)], SA, TSi, TSr, [V+] | | Not support | | Explanation |
| 55 | 3055 | first response <- IDr, [CERT+], AUTH, EAP, [V+] | | Not support | | Explanation |
| 55 | 3059 | repeat 1..N times / --> EAP Y <- EAP | | Not support | | Explanation |
| 55 | 3063 | last request --> AUTH | | Not support | | Explanation |
| 55 | 3065 | last response <- AUTH, [CP(CFG_REPLY)], [N(IPCOMP_SUPPORTED)], [N(USE_TRANSPORT_MODE)], [N(ESP_TFC_PADDING_NOT_SUPPORTED)], [N(NON_FIRST_FRAGMENTS_ALSO)], SA, TSi, TSr, [N(ADDITIONAL_TS_POSSIBLE)], [V+] | | Not support | | Explanation |
| 56 | 3086 | A.4. CREATE_CHILD_SA Exchange for Creating/Rekeying CHILD_SAs | | | | |
| 56 | 3088 | request --> [N(REKEY_SA)], [N(IPCOMP_SUPPORTED)+], [N(USE_TRANSPORT_MODE)], [N(ESP_TFC_PADDING_NOT_SUPPORTED)], [N(NON_FIRST_FRAGMENTS_ALSO)], SA, Ni, [KEi], TSi, TSr | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.3.2 EN.I.1.2.5.2 EN.R.1.2.5.2 EN.R.1.2.7.1 SGW.I.1.2.3.2 SGW.I.1.2.5.2 SGW.R.1.2.5.2 SGW.R.1.2.7.1 |
| 56 | 3095 | response <- [N(IPCOMP_SUPPORTED)], [N(USE_TRANSPORT_MODE)], [N(ESP_TFC_PADDING_NOT_SUPPORTED)], [N(NON_FIRST_FRAGMENTS_ALSO)], SA, Nr, [KEr], TSi, TSr, [N(ADDITIONAL_TS_POSSIBLE)] | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.3.2 EN.I.1.2.5.2 EN.R.1.2.5.2 EN.R.1.2.7.1 SGW.I.1.2.3.2 SGW.I.1.2.5.2 SGW.R.1.2.5.2 SGW.R.1.2.7.1 |
| 56 | 3102 | A.5. CREATE_CHILD_SA Exchange for Rekeying the IKE_SA | | | | |
| 56 | 3104 | request --> SA, Ni, [KEi] | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.4.2 EN.R.1.2.6.1 SGW.I.1.2.4.2 SGW.R.1.2.6.1 |
| 56 | 3106 | response <- SA, Nr, [KEr] | | BASIC | EN(initiator) EN(responder) SGW(initiator) SGW(responder) | EN.I.1.2.4.2 EN.R.1.2.6.1 SGW.I.1.2.4.2 SGW.R.1.2.6.1 |
| 56 | 3108 | A.6. INFORMATIONAL Exchange | | | | |
| 56 | 3110 | request --> [N+], [D+], [CP(CFG_REQUEST)] | | BASIC | EN(responder) SGW(responder) | EN.R.1.3.1.1 SGW.R.1.3.1.1 |
| 56 | 3114 | response <- [N+], [D+], [CP(CFG_REPLY)] | | BASIC | EN(responder) SGW(responder) | EN.R.1.3.1.1 SGW.R.1.3.1.1 |