ETSI TS 102 558 V1.1.1 (2006-12)

Technical Specification

Methods for Testing and Specification (MTS);
Internet Protocol Testing (IPT);
IPv6 Security;
Requirements Catalogue

Reference
DTS/MTS-IPT-008-IPV6-SecReq

Keywords
IP, IPv6, security, testing

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

1 Scope

The present document is a catalogue of all of the security-related IPv6 requirements extracted from the following IETF specifications:

RFC 4301 [1]: "Security Architecture for the Internet Protocol".

RFC 4302 [2]: "IP Authentication Header".

RFC 4303 [3]: "IP Encapsulating Security Payload (ESP)".

RFC 4305 [4]: "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)".

RFC 4306 [5] "Internet Key Exchange (IKEv2) Protocol".

RFC 2405 [6]: "The ESP DES-CBC Cipher Algorithm With Explicit IV".

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

[1] IETF RFC 4301: "Security Architecture for the Internet Protocol".

[2] IETF RFC 4302: "IP Authentication Header".

[3] IETF RFC 4303: "IP Encapsulating Security Payload (ESP)".

[4] IETF RFC 4305: "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)".

[5] IETF RFC 4306 "Internet Key Exchange (IKEv2) Protocol".

[6] IETF RFC 2405: "The ESP DES-CBC Cipher Algorithm With Explicit IV".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AH	Authentication Header
CBC	Cipher Block Chaining
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
EAP	Extensible Authentication Procedure
ESN	Extended Sequence Number
ESP	Encapsulated Security Payload
IANA	Internet Assigned Number Association
ICMP	Internet Control Message Protocol
ICV	Integrity Check Value
IETF	Internet Engineering Task Force
IKEv2	Internet Key Exchange protocol version 2
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IV	Initialization Vector
MAC	Message Authentication Code
PMTU	Path Maximum Transmission Unit
RFC	Request For Comments

NOTE: IETF terminology for a draft standard.

SA	Security Association
SAD	Security Association Database
SPD	Security Policies Database
SPI	Security Parameters Index
TCP	Transport Control Protocol
UDP	User Datagram Protocol

4 Requirements Catalogue

The security requirements related to Internet Protocol version 6 (IPv6) are specified in a number of IETF documents. These documents include requirements for the overall IPv6 security architecture [1], the use of the IP Authentication Header (AH) [2], IP Encapsulating Security Payload (ESP) [3], the use of cryptographic algorithms [4], [6] and the Internet Key Exchange (IKEv2) [5]. The present document is a catalogue of all of the normative requirements from these security specifications. Each requirement is given a unique identifier (for example, RQ_002_1234) and the following information is included with each:

- the clause number in the RFC from which the requirement has been extracted;
- the type of requirement (Mandatory, Optional or Recommended);
- the type of device to which the requirement applies (for example, Host or Router);
- the actual text from which the requirement was extracted.

4.1 Requirements extracted from RFC 4301

Identifier: RQ_002_1004
RFC Clause: 3.2
Type: Mandatory
Applies to: IPsec host

Requirement:
IPsec implementations MUST support ESP

RFC Text:
IPsec implementations MUST support ESP and MAY support AH.

Identifier: RQ_002_1005
RFC Clause: 3.2
Type: Optional
Applies to: IPsec host

Requirement:
IPsec implementations MAY support AH.

RFC Text:
IPsec implementations MUST support ESP and MAY support AH.

Identifier: RQ_002_1010
RFC Clause: 3.2
Type: Mandatory
Applies to: IPsec host

Requirement:
Manual distribution of keys MUST be supported

RFC Text:
Because most of the security services provided by IPsec require the use of cryptographic keys, IPsec relies on a separate set of mechanisms for putting these keys in place. **This document requires support for both manual and automated distribution of keys**. It specifies a specific public-key based approach (IKEv2 [Kau05]) for automated key management, but other automated key distribution techniques MAY be used.

Identifier: RQ_002_1011
RFC Clause: 3.2
Type: Mandatory
Applies to: IPsec host

Requirement:
Automatic distribution of keys MUST be supported

RFC Text:
Because most of the security services provided by IPsec require the use of cryptographic keys, IPsec relies on a separate set of mechanisms for putting these keys in place. **This document requires support for both manual and automated distribution of keys**. It specifies a specific public-key based approach (IKEv2 [Kau05]) for automated key management, but other automated key distribution techniques MAY be used.

Identifier: RQ_002_1014
RFC Clause: 4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
A Security Association MUST apply to exactly one of ESP or AH

RFC Text:
An SA is a simplex "connection" that affords security services to the traffic carried by it. Security services are afforded to an SA by the use of AH, or ESP, but not both. If both AH and ESP protection are applied to a traffic stream, then two SAs must be created and coordinated to effect protection through iterated application of the security protocols. To secure typical, bi-directional communication between two IPsec-enabled systems, a pair of SAs (one in each direction) is required. IKE explicitly creates SA pairs in recognition of this common usage requirement.

Identifier: RQ_002_1020
RFC Clause: 4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
A host implementation of IPsec MUST support transport mode

RFC Text:
In summary,

a) **A host implementation of IPsec MUST support both transport and tunnel mode**. This is true for native, BITS, and BITW implementations for hosts.

b) A security gateway MUST support tunnel mode and MAY support transport mode. If it supports transport mode, that should be used only when the security gateway is acting as a host, e.g., for network management, or to provide security between two intermediate systems along a path.

Identifier: RQ_002_1021
RFC Clause: 4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
A host implementation of IPsec MUST support tunnel mode

RFC Text:
In summary,

a) **A host implementation of IPsec MUST support both transport and tunnel mode**. This is true for native, BITS, and BITW implementations for hosts.

b) A security gateway MUST support tunnel mode and MAY support transport mode. If it supports transport mode, that should be used only when the security gateway is acting as a host, e.g., for network management, or to provide security between two intermediate systems along a path.

Identifier: RQ_002_1022
RFC Clause: 4.1
Type: Mandatory
Applies to: IPsec gateway

Requirement:
A gateway implementation of IPsec MUST support tunnel mode

RFC Text:
In summary,

a) A host implementation of IPsec MUST support both transport and tunnel mode. This is true for native, BITS, and BITW implementations for hosts.

b) **A security gateway MUST support tunnel mode** and MAY support transport mode. If it supports transport mode, that should be used only when the security gateway is acting as a host, e.g., for network management, or to provide security between two intermediate systems along a path.

Identifier: RQ_002_1023
RFC Clause: 4.1
Type: Optional
Applies to: IPsec gateway

Requirement:
A gateway implementation of IPsec MAY support transport mode

RFC Text:
In summary,

a) A host implementation of IPsec MUST support both transport and tunnel mode. This is true for native, BITS, and BITW implementations for hosts.

b) **A security gateway MUST support tunnel mode and MAY support transport mode**. If it supports transport mode, that should be used only when the security gateway is acting as a host, e.g., for network management, or to provide security between two intermediate systems along a path.

4.2 Requirements extracted from RFC 4302

Identifier: RQ_002_2000
RFC Clause: 2
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends an IP packet containing an Authentication Header (AH), it MUST set the appropriate Next Header field (either in the IPv6 Header or in the previous Extension Header) to the value fifty-one (51)

RFC Text:
The protocol header (IPv4, IPv6, or IPv6 Extension) immediately preceding the AH header SHALL contain the value 51 in its Protocol (IPv4) or Next Header (IPv6, Extension) fields [DH98]. Figure 1 illustrates the format for AH.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Payload Len  |          RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Security Parameters Index (SPI)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Sequence Number Field                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                Integrity Check Value-ICV (variable)           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
                       Figure 1.  AH Format
```

Identifier: RQ_002_2001
RFC Clause: 2
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends an IP packet containing an Authentication Header (AH), it MUST construct the Authentication Header in the following format:

```
  Octet           Field
  ---------------------
  1               Next Header
  2               Payload Length
  3 & 4           Reserved
  5 to 8          Security Parameters Index (SPI)
  9 to 12         Sequence Number
  13 to end       Integrity Check Value (ICV)
```

RFC Text:
The protocol header (IPv4, IPv6, or IPv6 Extension) immediately preceding the AH header SHALL contain the value 51 in its Protocol (IPv4) or Next Header (IPv6, Extension) fields [DH98]. **Figure 1 illustrates the format for AH.**

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Payload Len  |          RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Security Parameters Index (SPI)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Sequence Number Field                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                Integrity Check Value-ICV (variable)           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                       Figure 1.  AH Format
```

Identifier: RQ_002_2002
RFC Clause: 2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends an IP packet containing an Authentication Header (AH), it MUST set the AH Next Header field to the appropriate value as defined in IETF RFC 1700

RFC Text:
The Next Header is an 8-bit field that identifies the type of the next payload after the Authentication Header. **The value of this field is chosen from the set of IP Protocol Numbers defined on the web page of Internet Assigned Numbers Authority (IANA)**. For example, a value of 4 indicates IPv4, a value of 41 indicates IPv6, and a value of 6 indicates TCP.

Identifier: RQ_002_2003
RFC Clause: 2.2
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends an IP packet containing an Authentication Header (AH), it MUST set the AH Payload Length field to a value equal to two less than the length in 32-bit words of the Authentication Header

RFC Text:
This 8-bit field specifies the length of AH in 32-bit words (4-byte units), minus "2". Thus, for example, if an integrity algorithm yields a 96-bit authentication value, this length field will be "4" (3 32-bit word fixed fields plus 3 32-bit words for the ICV, minus 2). For IPv6, the total length of the header must be a multiple of 8-octet units. (Note that although IPv6 [DH98] characterizes AH as an extension header, its length is measured in 32-bit words, not the 64-bit words used by other IPv6 extension headers.) See Section 2.6, "Integrity Check Value (ICV)", for comments on padding of this field, and Section 3.3.3.2.1, "ICV Padding".

Identifier: RQ_002_2004
RFC Clause: 2.3
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends an IP packet containing an Authentication Header (AH), it MUST set to zero the octets identified as "Reserved" in the Authentication Header.

RFC Text:
This 16-bit field is reserved for future use. **It MUST be set to "zero" by the sender**, and it SHOULD be ignored by the recipient. (Note that the value is included in the ICV calculation, but is otherwise ignored by the recipient.)

Identifier: RQ_002_2005
RFC Clause: 2.3
Type: Recommended
Applies to: IPsec host

Requirement:
When an IPsec Host receives an IP packet containing an Authentication Header (AH), it SHOULD ignore the octets identified as "Reserved" in the Authentication Header.

RFC Text:
This 16-bit field is reserved for future use. It MUST be set to "zero" by the sender, **and it SHOULD be ignored by the recipient**. (Note that the value is included in the ICV calculation, but is otherwise ignored by the recipient.)

Identifier: RQ_002_2006
RFC Clause: 2.4
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends a unicast IP packet containing an Authentication Header (AH), it MUST set the AH Security Parameters Index (SPI) to the SPI value provided by the other IPsec Security Association (SA) endpoint when the SA was established.

RFC Text:
The SPI is an arbitrary 32-bit value that is used by a receiver to identify the SA to which an incoming packet is bound. For a unicast SA, the SPI can be used by itself to specify an SA, or it may be used in conjunction with the IPsec protocol type (in this case AH). Because for unicast SAs the SPI value is generated by the receiver, whether the value is sufficient to identify an SA by itself or whether it must be used in conjunction with the IPsec protocol value is a local matter. The SPI field is mandatory, and this mechanism for mapping inbound traffic to unicast SAs described above MUST be supported by all AH implementations.

Identifier: RQ_002_2007
RFC Clause: 2.4
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends a multicast IP packet containing an Authentication Header (AH), it MUST set the AH Security Parameters Index (SPI) to the value assigned to it.

RFC Text:
In many secure multicast architectures, e.g., [RFC3740], a central Group Controller/Key Server unilaterally assigns the group security association's SPI. This SPI assignment is not negotiated or coordinated with the key management (e.g., IKE) subsystems that reside in the individual end systems that comprise the group. Consequently, it is possible that a group security association and a unicast security association can simultaneously use the same SPI. A multicast-capable IPsec implementation MUST correctly de-multiplex inbound traffic even in the context of SPI collisions.

Identifier: RQ_002_2008
RFC Clause: 2.4
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host receives a multicast IP packet containing an Authentication Header (AH), it MUST use the AH Security Parameters Index field to identify correctly Security Association related to the incoming packet.

RFC Text:
In many secure multicast architectures, e.g., [RFC3740], a central Group Controller/Key Server unilaterally assigns the group security association's SPI. This SPI assignment is not negotiated or coordinated with the key management (e.g., IKE) subsystems that reside in the individual end systems that comprise the group. Consequently, it is possible that a group security association and a unicast security association can simultaneously use the same SPI. **A multicast-capable IPsec implementation MUST correctly de-multiplex inbound traffic even in the context of SPI collisions**.

Identifier: RQ_002_2009
RFC Clause: 2.4
Type: Mandatory
Applies to: IPsec host

Requirement:

```
If an IPsec Host receives a multicast  IP packet containing an Authentication Header (AH) but is
unable to relate the header to an established Security |Association, it MUST discard the incoming
packet.
```

RFC Text:

```
Each entry in the Security Association Database (SAD) [Ken-Arch] must
indicate whether the SA lookup makes use of the destination, or
destination and source, IP addresses, in addition to the SPI.  For
multicast SAs, the protocol field is not employed for SA lookups.
For each inbound, IPsec-protected packet, an implementation must
conduct its search of the SAD such that it finds the entry that
matches the "longest" SA identifier.  In this context, if two or more
SAD entries match based on the SPI value, then the entry that also
matches based on destination, or destination and source, address
comparison (as indicated in the SAD entry) is the "longest" match.
This implies a logical ordering of the SAD search as follows:

        1. Search the SAD for a match on {SPI, destination
           address, source address}.  If an SAD entry
           matches, then process the inbound AH packet with that
           matching SAD entry.  Otherwise, proceed to step 2.

        2. Search the SAD for a match on {SPI, destination
           address}.  If an SAD entry matches, then process
           the inbound AH packet with that matching SAD
           entry.  Otherwise, proceed to step 3.

        3. Search the SAD for a match on only {SPI} if the receiver
           has chosen to maintain a single SPI space for AH and ESP,
           or on {SPI, protocol} otherwise.  If an SAD
           entry matches, then process the inbound AH packet with
           that matching SAD entry.  Otherwise, discard the packet
           and log an auditable event.

--------------------
```

Identifier: RQ_002_2010
RFC Clause: 2.4
Type: Recommended
Applies to: IPsec host

Requirement:

If an IPsec Host receives a multicast IP packet containing an Authentication Header (AH) but is unable to relate the header to an established Security |Association, it SHOULD record in the audit log the SPI value, date/time, Source Address, Destination Address, and (in IPv6) the Flow ID

RFC Text:

Each entry in the Security Association Database (SAD) [Ken-Arch] must indicate whether the SA lookup makes use of the destination, or destination and source, IP addresses, in addition to the SPI. For multicast SAs, the protocol field is not employed for SA lookups. For each inbound, IPsec-protected packet, an implementation must conduct its search of the SAD such that it finds the entry that matches the "longest" SA identifier. In this context, if two or more SAD entries match based on the SPI value, then the entry that also matches based on destination, or destination and source, address comparison (as indicated in the SAD entry) is the "longest" match. This implies a logical ordering of the SAD search as follows:

1. Search the SAD for a match on {SPI, destination address, source address}. If an SAD entry matches, then process the inbound AH packet with that matching SAD entry. Otherwise, proceed to step 2.

2. Search the SAD for a match on {SPI, destination address}. If an SAD entry matches, then process the inbound AH packet with that matching SAD entry. Otherwise, proceed to step 3.

3. Search the SAD for a match on only {SPI} if the receiver has chosen to maintain a single SPI space for AH and ESP, or on {SPI, protocol} otherwise. If an SAD entry matches, then process the inbound AH packet with that matching SAD entry. Otherwise, discard the packet and **log an auditable event**.

Identifier: RQ_002_2011
RFC Clause: 2.4
Type: Mandatory
Applies to: IPsec host

Requirement:

When an IPsec Host sends an IP packet containing an Authentication Header (AH), it MUST NOT set a value in the range 0 to 255 into the Security Parameters Index field of the Authentication Header

RFC Text:

The set of SPI values in the range 1 through 255 is reserved by the Internet Assigned Numbers Authority (IANA) for future use; a reserved SPI value will not normally be assigned by IANA unless the use of the assigned SPI value is specified in an RFC. **The SPI value of zero (0) is reserved for local, implementation-specific use and MUST NOT be sent on the wire**. (For example, a key management implementation might use the zero SPI value to mean "No Security Association Exists" during the period when the IPsec implementation has requested that its key management entity establish a new SA, but the SA has not yet been established.)

Identifier: RQ_002_2012
RFC Clause: 2.5
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends an IP packet containing an Authentication Header (AH) on a unicast or single-sender multicast Security Association (SA), it MUST set the value in the Sequence Number field to one more than the value set in the same field of the previous packet sent to the same SA

RFC Text:
This unsigned 32-bit field contains a counter value that increases by one for each packet sent, i.e., a per-SA packet sequence number. **For a unicast SA or a single-sender multicast SA, the sender MUST increment this field for every transmitted packet**. Sharing an SA among multiple senders is permitted, though generally not recommended. AH provides no means of synchronizing packet counters among multiple senders or meaningfully managing a receiver packet counter and window in the context of multiple senders. Thus, for a multi-sender SA, the anti-reply features of AH are not available (see Sections 3.3.2 and 3.4.3).

The field is mandatory and MUST always be present even if the receiver does not elect to enable the anti-replay service for a specific SA. Processing of the Sequence Number field is at the discretion of the receiver, but all AH implementations MUST be capable of performing the processing described in Section 3.3.2, "Sequence Number Generation", and Section 3.4.3, "Sequence Number Verification". Thus, **the sender MUST always transmit this field**, but the receiver need not act upon it.

The sender's counter and the receiver's counter are initialized to 0 when an SA is established. (The first packet sent using a given SA will have a sequence number of 1; see Section 3.3.2 for more details on how the sequence number is generated.) If anti-replay is enabled (the default), the transmitted sequence number must never be allowed to cycle. Thus, the sender's counter and the receiver's counter MUST be reset (by establishing a new SA and thus a new key) prior to the transmission of the 2^32nd packet on an SA

Identifier: RQ_002_2013
RFC Clause: 2.5
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends the first IP packet containing an Authentication Header (AH) on a particular unicast or single-sender multicast Security Association (SA), it MUST set the value in the Sequence Number field to one (1)

RFC Text:
This unsigned 32-bit field contains a counter value that increases by one for each packet sent, i.e., a per-SA packet sequence number. For a unicast SA or a single-sender multicast SA, the sender MUST increment this field for every transmitted packet. Sharing an SA among multiple senders is permitted, though generally not recommended. AH provides no means of synchronizing packet counters among multiple senders or meaningfully managing a receiver packet counter and window in the context of multiple senders. Thus, for a multi-sender SA, the anti-reply features of AH are not available (see Sections 3.3.2 and 3.4.3).

The field is mandatory and MUST always be present even if the receiver does not elect to enable the anti-replay service for a specific SA. Processing of the Sequence Number field is at the discretion of the receiver, but all AH implementations MUST be capable of performing the processing described in Section 3.3.2, "Sequence Number Generation", and Section 3.4.3, "Sequence Number Verification". Thus, the sender MUST always transmit this field, but the receiver need not act upon it.

The sender's counter and the receiver's counter are initialized to 0 when an SA is established. (**The first packet sent using a given SA will have a sequence number of 1;** see Section 3.3.2 for more details on how the sequence number is generated.) If anti-replay is enabled (the default), the transmitted sequence number must never be allowed to cycle. Thus, the sender's counter and the receiver's counter MUST be reset (by establishing a new SA and thus a new key) prior to the transmission of the 2^32nd packet on an SA

Identifier: RQ_002_2014
RFC Clause: 2.5
Type: Mandatory
Applies to: IPsec host

Requirement:
If incrementing the value in the Sequence Number field of an Authentication Header would cause it to overflow as a 32-bit value (i.e., return to zero) prior to sending the associated IP packet and if the anti-replay service is also enabled, an IPsec Host MUST delete the corresponding Security Association and establish a new one to replace it.

RFC Text:
This unsigned 32-bit field contains a counter value that increases by one for each packet sent, i.e., a per-SA packet sequence number. For a unicast SA or a single-sender multicast SA, the sender MUST increment this field for every transmitted packet. Sharing an SA among multiple senders is permitted, though generally not recommended. AH provides no means of synchronizing packet counters among multiple senders or meaningfully managing a receiver packet counter and window in the context of multiple senders. Thus, for a multi-sender SA, the anti-reply features of AH are not available (see Sections 3.3.2 and 3.4.3).

The field is mandatory and MUST always be present even if the receiver does not elect to enable the anti-replay service for a specific SA. Processing of the Sequence Number field is at the discretion of the receiver, but all AH implementations MUST be capable of performing the processing described in Section 3.3.2, "Sequence Number Generation", and Section 3.4.3, "Sequence Number Verification". Thus, the sender MUST always transmit this field, but the receiver need not act upon it.

The sender's counter and the receiver's counter are initialized to 0 when an SA is established. (The first packet sent using a given SA will have a sequence number of 1; see Section 3.3.2 for more details on how the sequence number is generated.) **If anti-replay is enabled (the default), the transmitted sequence number must never be allowed to cycle. Thus, the sender's counter and the receiver's counter MUST be reset (by establishing a new SA and thus a new key) prior to the transmission of the 2^32nd packet on an SA**

Identifier: RQ_002_2015
RFC Clause: 2.6
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends an IP packet containing an Authentication Header (AH), it MUST set the value in the Integrity Check Value (ICV) field to a check value which is computed according to the integrity algorithm negotiated during the establishment of the corresponding Security Association

RFC Text:
This is a variable-length field that contains the Integrity Check Value (ICV) for this packet. The field must be an integral multiple of 32 bits (IPv4 or IPv6) in length. The details of ICV processing are described in Section 3.3.3, "Integrity Check Value Calculation", and Section 3.4.4, "Integrity Check Value Verification". This field may include explicit padding, if required to ensure that the length of the AH header is an integral multiple of 32 bits (IPv4) or 64 bits (IPv6). All implementations MUST support such padding and MUST insert only enough padding to satisfy the IPv4/IPv6 alignment requirements. Details of how to compute the required padding length are provided below in Section 3.3.3.2, "Padding". The integrity algorithm specification MUST specify the length of the ICV and the comparison rules and processing steps for validation.

Identifier: RQ_002_2016
RFC Clause: 2.6
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends an IPv4 packet containing an Authentication Header (AH), it MUST include up to 31 padding bits within the Integrity Check Value (ICV) field if these are necessary to ensure that the field is an integral multiple of 32 bits in length

RFC Text:
This is a variable-length field that contains the Integrity Check Value (ICV) for this packet. **The field must be an integral multiple of 32 bits (IPv4 or IPv6) in length**. The details of ICV processing are described in Section 3.3.3, "Integrity Check Value Calculation", and Section 3.4.4, "Integrity Check Value Verification". **This field may include explicit padding, if required to ensure that the length of the AH header is an integral multiple of 32 bits (IPv4) or 64 bits (IPv6). All implementations MUST support such padding and MUST insert only enough padding to satisfy the IPv4/IPv6 alignment requirements.** Details of how to compute the required padding length are provided below in Section 3.3.3.2, "Padding". The integrity algorithm specification MUST specify the length of the ICV and the comparison rules and processing steps for validation.

Identifier: RQ_002_2017
RFC Clause: 2.6
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host sends an IPv6 packet containing an Authentication Header (AH), it MUST include up to 63 padding bits within the Integrity Check Value (ICV) field if these are necessary to ensure that the field is an integral multiple of 64 bits in length

RFC Text:
This is a variable-length field that contains the Integrity Check Value (ICV) for this packet. **The field must be an integral multiple of 32 bits (IPv4 or IPv6) in length**. The details of ICV processing are described in Section 3.3.3, "Integrity Check Value Calculation", and Section 3.4.4, "Integrity Check Value Verification". **This field may include explicit padding, if required to ensure that the length of the AH header is an integral multiple of 32 bits (IPv4) or 64 bits (IPv6). All implementations MUST support such padding and MUST insert only enough padding to satisfy the IPv4/IPv6 alignment requirements.** Details of how to compute the required padding length are provided below in Section 3.3.3.2, "Padding". The integrity algorithm specification MUST specify the length of the ICV and the comparison rules and processing steps for validation.

Identifier: RQ_002_2018
RFC Clause: 3.1.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host uses Transport Mode to send an IPv4 packet containing an Authentication Header (AH), it MUST insert the Authentication Header after the IPv4 Header (and any options that it contains) but before the next layer protocol.

RFC Text:
In transport mode, AH is inserted after the IP header and before a next layer protocol (e.g., TCP, UDP, ICMP, etc.) or before any other IPsec headers that have already been inserted. In the context of IPv4, this calls for placing AH after the IP header (and any options that it contains), but before the next layer protocol. (Note that the term "transport" mode should not be misconstrued as restricting its use to TCP and UDP.) The following diagram illustrates AH transport mode positioning for a typical IPv4 packet, on a "before and after" basis.

```
                BEFORE APPLYING AH
          ----------------------------
    IPv4  |orig IP hdr  |     |      |
          |(any options)| TCP | Data |
          ----------------------------

                AFTER APPLYING AH
          -------------------------------------------------------
    IPv4  |original IP hdr (any options) | AH | TCP |    Data   |
          -------------------------------------------------------
          |<- mutable field processing ->|<- immutable fields ->|
          |<----- authenticated except for mutable fields ----->|
```

Identifier: RQ_002_2019
RFC Clause: 3.1.1
Type: Recommended
Applies to: IPsec host

Requirement:
When an IPsec Host uses Transport Mode to send an IPv6 packet containing an Authentication Header (AH), it SHOULD insert the Authentication Header after the IPv6 Hop-By-Hop, Routing and Fragmentation Extension Headers

RFC Text:
In the IPv6 context, AH is viewed as an end-to-end payload, and thus should appear after hop-by-hop, routing, and fragmentation extension headers. The destination options extension header(s) could appear before or after or both before and after the AH header depending on the semantics desired. The following diagram illustrates AH transport mode positioning for a typical IPv6 packet.

```
                     BEFORE APPLYING AH
          ---------------------------------------
    IPv6  |             | ext hdrs |     |      |
          | orig IP hdr |if present| TCP | Data |
          ---------------------------------------

                    AFTER APPLYING AH
         ------------------------------------------------------------
   IPv6  |             |hop-by-hop, dest*, |    | dest |     |      |
         |orig IP hdr  |routing, fragment. | AH | opt* | TCP | Data |
         ------------------------------------------------------------
         |<--- mutable field processing -->|<-- immutable fields -->|
         |<---- authenticated except for mutable fields ----------->|

               * = if present, could be before AH, after AH, or both
```

Identifier: RQ_002_2020
RFC Clause: 3.1.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host uses Tunnel Mode to send an IPv4 packet containing an Authentication Header (AH) within the payload of an "outer" IP packet, it MUST insert the Authentication Header after the "inner" IPv4 Header (and any options that it contains) but before the next layer protocol.

RFC Text:
In tunnel mode, the "inner" IP header carries the ultimate (IP) source and destination addresses, while an "outer" IP header contains the addresses of the IPsec "peers," e.g., addresses of security gateways. Mixed inner and outer IP versions are allowed, i.e., IPv6 over IPv4 and IPv4 over IPv6. In tunnel mode, AH protects the entire inner IP packet, including the entire inner IP header. The position of AH in tunnel mode, relative to the outer IP header, is the same as for AH in transport mode. The following diagram illustrates AH tunnel mode positioning for typical IPv4 and IPv6 packets.

```
     ----------------------------------------------------------------
IPv4 |                              |    | orig IP hdr*  |   |      |
     |new IP header * (any options) | AH | (any options) |TCP| Data |
     ----------------------------------------------------------------
     |<- mutable field processing ->|<------ immutable fields ----->|
     |<- authenticated except for mutable fields in the new IP hdr->|

     --------------------------------------------------------------
IPv6 |           | ext hdrs*|    |            | ext hdrs*|   |    |
     |new IP hdr*|if present| AH |orig IP hdr*|if present|TCP|Data|
     --------------------------------------------------------------
     |<--- mutable field -->|<--------- immutable fields -------->|
     |       processing     |
     |<-- authenticated except for mutable fields in new IP hdr ->|

       * = if present, construction of outer IP hdr/extensions and
           modification of inner IP hdr/extensions is discussed in
           the Security Architecture document.
```

Identifier: RQ_002_2021
RFC Clause: 3.1.1
Type: Recommended
Applies to: IPsec host

Requirement:
When an IPsec Host uses Tunnel Mode to send an IPv6 packet containing an Authentication Header (AH) within an "outer" IP packet, it SHOULD insert the Authentication Header after the "inner" IPv6 Hop-By-Hop, Routing and Fragmentation Extension Headers

RFC Text:
In tunnel mode, the "inner" IP header carries the ultimate (IP) source and destination addresses, while an "outer" IP header contains the addresses of the IPsec "peers," e.g., addresses of security gateways. Mixed inner and outer IP versions are allowed, i.e., IPv6 over IPv4 and IPv4 over IPv6. In tunnel mode, AH protects the entire inner IP packet, including the entire inner IP header. The position of AH in tunnel mode, relative to the outer IP header, is the same as for AH in transport mode. The following diagram illustrates AH tunnel mode positioning for typical IPv4 and IPv6 packets.

```
     ----------------------------------------------------------------
IPv4 |                              |    | orig IP hdr*  |   |      |
     |new IP header * (any options) | AH | (any options) |TCP| Data |
     ----------------------------------------------------------------
     |<- mutable field processing ->|<------ immutable fields ----->|
     |<- authenticated except for mutable fields in the new IP hdr->|

     --------------------------------------------------------------
IPv6 |           | ext hdrs*|    |            | ext hdrs*|   |    |
     |new IP hdr*|if present| AH |orig IP hdr*|if present|TCP|Data|
     --------------------------------------------------------------
     |<--- mutable field -->|<--------- immutable fields -------->|
     |       processing     |
     |<-- authenticated except for mutable fields in new IP hdr ->|

       * = if present, construction of outer IP hdr/extensions and
           modification of inner IP hdr/extensions is discussed in
           the Security Architecture document.
```

Identifier: RQ_002_2022
RFC Clause: 3.3.1
Type: Mandatory
Applies to: IPsec host

Requirement:

When an IPsec Host sends the first IP packet containing an Authentication Header (AH) on a particular unicast or single-sender multicast Security Association (SA), it MUST set the value in the Sequence Number field to one (1)

RFC Text:

The sender's counter is initialized to 0 when an SA is established. The sender increments the sequence number (or ESN) counter for this SA and inserts the low-order 32 bits of the value into the Sequence Number field. **Thus, the first packet sent using a given SA will contain a sequence number of 1.**

If anti-replay is enabled (the default), the sender checks to ensure that the counter has not cycled before inserting the new value in the Sequence Number field. In other words, the sender MUST NOT send a packet on an SA if doing so would cause the sequence number to cycle. An attempt to transmit a packet that would result in sequence number overflow is an auditable event. The audit log entry for this event SHOULD include the SPI value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.

The sender assumes anti-replay is enabled as a default, unless otherwise notified by the receiver (see Section 3.4.3) or if the SA was configured using manual key management. Thus, typical behavior of an AH implementation calls for the sender to establish a new SA when the Sequence Number (or ESN) cycles, or in anticipation of this value cycling.

If anti-replay is disabled (as noted above), the sender does not need to monitor or reset the counter, e.g., in the case of manual key management (see Section 5). However, the sender still increments the counter and when it reaches the maximum value, the counter rolls over back to zero. (This behavior is recommended for multi-sender, multicast SAs, unless anti-replay mechanisms outside the scope of this standard are negotiated between the sender and receiver.)

If ESN (see Appendix B) is selected, only the low-order 32 bits of the sequence number are transmitted in the Sequence Number field, although both sender and receiver maintain full 64-bit ESN counters. However, the high-order 32 bits are included in the ICV calculation. Note: If a receiver chooses not to enable anti-replay for an SA, then the receiver SHOULD NOT negotiate ESN in an SA management protocol. Use of ESN creates a need for the receiver to manage the anti-replay window (in order to determine the correct value for the high-order bits of the ESN, which are employed in the ICV computation), which is generally contrary to the notion of disabling anti-replay for an SA

Identifier: RQ_002_2023
RFC Clause: 3.3.2
Type: Mandatory
Applies to: IPsec host

Requirement:

If incrementing the value in the Sequence Number field of an Authentication Header would cause it to overflow as a 32-bit value (i.e., return to zero) prior to sending the associated IP packet and if the anti-replay service is also enabled, an IPsec Host MUST delete the corresponding Security Association and establish a new one to replace it.

RFC Text:

The sender's counter is initialized to 0 when an SA is established. The sender increments the sequence number (or ESN) counter for this SA and inserts the low-order 32 bits of the value into the Sequence Number field. Thus, the first packet sent using a given SA will contain a sequence number of 1.

If anti-replay is enabled (the default), the sender checks to ensure that the counter has not cycled before inserting the new value in the Sequence Number field. In other words, the sender MUST NOT send a packet on an SA if doing so would cause the sequence number to cycle. An attempt to transmit a packet that would result in sequence number overflow is an auditable event. The audit log entry for this event SHOULD include the SPI value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.

The sender assumes anti-replay is enabled as a default, unless otherwise notified by the receiver (see Section 3.4.3) or if the SA was configured using manual key management. Thus, typical behavior of an AH implementation calls for the sender to establish a new SA when the Sequence Number (or ESN) cycles, or in anticipation of this value cycling.

If anti-replay is disabled (as noted above), the sender does not need to monitor or reset the counter, e.g., in the case of manual key management (see Section 5). However, the sender still increments the counter and when it reaches the maximum value, the counter rolls over back to zero. (This behavior is recommended for multi-sender, multicast SAs, unless anti-replay mechanisms outside the scope of this standard are negotiated between the sender and receiver.)

If ESN (see Appendix B) is selected, only the low-order 32 bits of the sequence number are transmitted in the Sequence Number field, although both sender and receiver maintain full 64-bit ESN counters. However, the high-order 32 bits are included in the ICV calculation. Note: If a receiver chooses not to enable anti-replay for an SA, then the receiver SHOULD NOT negotiate ESN in an SA management protocol. Use of ESN creates a need for the receiver to manage the anti-replay window (in order to determine the correct value for the high-order bits of the ESN, which are employed in the ICV computation), which is generally contrary to the notion of disabling anti-replay for an SA

Identifier: RQ_002_2024
RFC Clause: 3.3.2
Type: Recommended
Applies to: IPsec host

Requirement:

```
If incrementing the value in the Sequence Number field of an Authentication Header would cause it to
overflow as a 32-bit value (i.e., return to zero) prior to sending the associated IP packet and if
the anti-replay service is also enabled, an IPsec Host SHOULD record in the audit log for this
event, the SPI value, current date/time, Source Address, Destination Address, and (in IPv6) the
cleartext Flow ID
```

RFC Text:

```
The sender's counter is initialized to 0 when an SA is established.
The sender increments the sequence number (or ESN) counter for this
SA and inserts the low-order 32 bits of the value into the Sequence
Number field.  Thus, the first packet sent using a given SA will
contain a sequence number of 1.

If anti-replay is enabled (the default), the sender checks to ensure
that the counter has not cycled before inserting the new value in the
Sequence Number field.  In other words, the sender MUST NOT send a
packet on an SA if doing so would cause the sequence number to cycle.
```

An attempt to transmit a packet that would result in sequence number overflow is an auditable event. The audit log entry for this event SHOULD include the SPI value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.

```
The sender assumes anti-replay is enabled as a default, unless
otherwise notified by the receiver (see Section 3.4.3) or if the SA
was configured using manual key management.  Thus, typical behavior
of an AH implementation calls for the sender to establish a new SA
when the Sequence Number (or ESN) cycles, or in anticipation of this
value cycling.

If anti-replay is disabled (as noted above), the sender does not need
to monitor or reset the counter, e.g., in the case of manual key
management (see Section 5).  However, the sender still increments the
counter and when it reaches the maximum value, the counter rolls over
back to zero.  (This behavior is recommended for multi-sender,
multicast SAs, unless anti-replay mechanisms outside the scope of
this standard are negotiated between the sender and receiver.)

If ESN (see Appendix B) is selected, only the low-order 32 bits of
the sequence number are transmitted in the Sequence Number field,
although both sender and receiver maintain full 64-bit ESN counters.
However, the high-order 32 bits are included in the ICV calculation.
Note: If a receiver chooses not to enable anti-replay for an SA, then
the receiver SHOULD NOT negotiate ESN in an SA management protocol.
Use of ESN creates a need for the receiver to manage the anti-replay
window (in order to determine the correct value for the high-order
bits of the ESN, which are employed in the ICV computation), which is
generally contrary to the notion of disabling anti-replay for an SA

--------------------
```

Identifier: RQ_002_2025
RFC Clause: 3.3.2
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec Host which has the anti-replay service enabled sends an IP packet containing an Authentication Header (AH) on a particular unicast or single-sender multicast Security Association (SA) on which the use of Extended Sequence Numbers (ESN) has been established, it MUST set the value in the Sequence Number field to the low-order 32 bits of the ESN

RFC Text:
The sender's counter is initialized to 0 when an SA is established. The sender increments the sequence number (or ESN) counter for this SA and inserts the low-order 32 bits of the value into the Sequence Number field. Thus, the first packet sent using a given SA will contain a sequence number of 1.

If anti-replay is enabled (the default), the sender checks to ensure that the counter has not cycled before inserting the new value in the Sequence Number field. In other words, the sender MUST NOT send a packet on an SA if doing so would cause the sequence number to cycle. An attempt to transmit a packet that would result in sequence number overflow is an auditable event. The audit log entry for this event SHOULD include the SPI value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.

The sender assumes anti-replay is enabled as a default, unless otherwise notified by the receiver (see Section 3.4.3) or if the SA was configured using manual key management. Thus, typical behavior of an AH implementation calls for the sender to establish a new SA when the Sequence Number (or ESN) cycles, or in anticipation of this value cycling.

If anti-replay is disabled (as noted above), the sender does not need to monitor or reset the counter, e.g., in the case of manual key management (see Section 5). However, the sender still increments the counter and when it reaches the maximum value, the counter rolls over back to zero. (This behavior is recommended for multi-sender, multicast SAs, unless anti-replay mechanisms outside the scope of this standard are negotiated between the sender and receiver.)

If ESN (see Appendix B) is selected, only the low-order 32 bits of the sequence number are transmitted in the Sequence Number field, although both sender and receiver maintain full 64-bit ESN counters. However, the high-order 32 bits are included in the ICV calculation. Note: If a receiver chooses not to enable anti-replay for an SA, then the receiver SHOULD NOT negotiate ESN in an SA management protocol. Use of ESN creates a need for the receiver to manage the anti-replay window (in order to determine the correct value for the high-order bits of the ESN, which are employed in the ICV computation), which is generally contrary to the notion of disabling anti-replay for an SA

Identifier: RQ_002_2026
RFC Clause: 3.3.2
Type: Recommended
Applies to: IPsec host

Requirement:
An IPsec host SHOULD NOT negotiate the use of Extended Sequence Numbers (ESN) on a Security Association for which the anti-replay service is not enabled.

RFC Text:
```
The sender's counter is initialized to 0 when an SA is established.
The sender increments the sequence number (or ESN) counter for this
SA and inserts the low-order 32 bits of the value into the Sequence
Number field.  Thus, the first packet sent using a given SA will
contain a sequence number of 1.

If anti-replay is enabled (the default), the sender checks to ensure
that the counter has not cycled before inserting the new value in the
Sequence Number field.  In other words, the sender MUST NOT send a
packet on an SA if doing so would cause the sequence number to cycle.
An attempt to transmit a packet that would result in sequence number
overflow is an auditable event.  The audit log entry for this event
SHOULD include the SPI value, current date/time, Source Address,
Destination Address, and (in IPv6) the cleartext Flow ID.

The sender assumes anti-replay is enabled as a default, unless
otherwise notified by the receiver (see Section 3.4.3) or if the SA
was configured using manual key management.  Thus, typical behavior
of an AH implementation calls for the sender to establish a new SA
when the Sequence Number (or ESN) cycles, or in anticipation of this
value cycling.

If anti-replay is disabled (as noted above), the sender does not need
to monitor or reset the counter, e.g., in the case of manual key
management (see Section 5).  However, the sender still increments the
counter and when it reaches the maximum value, the counter rolls over
back to zero.  (This behavior is recommended for multi-sender,
multicast SAs, unless anti-replay mechanisms outside the scope of
this standard are negotiated between the sender and receiver.)

If ESN (see Appendix B) is selected, only the low-order 32 bits of
the sequence number are transmitted in the Sequence Number field,
although both sender and receiver maintain full 64-bit ESN counters.
However, the high-order 32 bits are included in the ICV calculation.
```

Note: If a receiver chooses not to enable anti-replay for an SA, then the receiver SHOULD NOT negotiate ESN in an SA management protocol.
Use of ESN creates a need for the receiver to manage the anti-replay window (in order to determine the correct value for the high-order bits of the ESN, which are employed in the ICV computation), which is generally contrary to the notion of disabling anti-replay for an SA

Identifier: RQ_002_2027
RFC Clause: 3.3.3
Type: Mandatory
Applies to: IPsec host

Requirement:

```
When an IPsec host sends an IP packet containing an Authentication Header (AH), it MUST set a value
in the Integrity Check Value field which has been calculated over the following other fields using
the algorithm negotiated during SA establishment:

   * all immutable or predictable (at the receiving endpoint) IP and extension headers prior
     to the Authentication Header
   * the Authentication Header (Next Header field, Payload Length field, Reserved field, SPI
     field, Sequence Number field, ICV field - set to zero for the purposes of the calculation
     - and any explicit padding bytes)
   * all mutable and unpredictable fields with their values assumed to be zero
   * all information following the Authentication Header
   * the high-order 32 bits of the Extended Sequence Number (if enabled)
   * all implicit padding bytes required by the integrity algorithm
```

RFC Text:

```
The AH ICV is computed over:

     o IP or extension header fields before the AH header that are
       either immutable in transit or that are predictable in value
       upon arrival at the endpoint for the AH SA
     o the AH header (Next Header, Payload Len, Reserved, SPI,
       Sequence Number (low-order 32 bits), and the ICV (which is set
       to zero for this computation), and explicit padding bytes (if
       any))
     o everything after AH is assumed to be immutable in transit
     o the high-order bits of the ESN (if employed), and any implicit
       padding required by the integrity algorithm
```

Identifier: RQ_002_2028
RFC Clause: 3.3.3
Type: Mandatory
Applies to: IPsec host

Requirement:

```
When an IPsec host receives an IP packet containing an Authentication Header (AH), it MUST calculate
an Integrity Check Value over the following fields in the incoming packet using the algorithm
negotiated during SA establishment:

   * all immutable or predictable (at the receiving endpoint) IP and extension headers
     prior to the Authentication Header
   * the Authentication Header (Next Header field, Payload Length field, Reserved field,
     SPI field, Sequence Number field, ICV field - set to zero for the purposes of the
     calculation - and any explicit padding bytes)
   * all mutable and unpredictable fields with their values assumed to be zero
   * all information following the Authentication Header
   * the high-order 32 bits of the Extended Sequence Number (if enabled)
   * all implicit padding bytes required by the integrity algorithm
```

RFC Text:

```
The AH ICV is computed over:

     o IP or extension header fields before the AH header that are
       either immutable in transit or that are predictable in value
       upon arrival at the endpoint for the AH SA
     o the AH header (Next Header, Payload Len, Reserved, SPI,
       Sequence Number (low-order 32 bits), and the ICV (which is set
       to zero for this computation), and explicit padding bytes (if
       any))
     o everything after AH is assumed to be immutable in transit
     o the high-order bits of the ESN (if employed), and any implicit
       padding required by the integrity algorithm
```

Identifier: RQ_002_2029
RFC Clause: 3.3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When sending an IPv4 packet which contains an Authentication Header (AH), an IPsec host MUST include explicit padding bytes in the Integrity Check Value field if these are required to ensure that the header is a multiple of 32 bits.

RFC Text:
As mentioned in Section 2.6, the ICV field may include explicit padding if required to ensure that the AH header is a multiple of 32 bits (IPv4) or 64 bits (IPv6). If padding is required, its length is determined by two factors:

- the length of the ICV
- the IP protocol version (v4 or v6)

For example, if the output of the selected algorithm is 96 bits, no padding is required for IPv4 or IPv6. However, if a different length ICV is generated, due to use of a different algorithm, then padding may be required depending on the length and IP protocol version. The content of the padding field is arbitrarily selected by the sender. (The padding is arbitrary, but need not be random to achieve security.) These padding bytes are included in the ICV calculation, counted as part of the Payload Length, and transmitted at the end of the ICV field to enable the receiver to perform the ICV calculation. Inclusion of padding in excess of the minimum amount required to satisfy IPv4/IPv6 alignment requirements is prohibited

----

Identifier: RQ_002_2030
RFC Clause: 3.3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When sending an IPv6 packet which contains an Authentication Header (AH), an IPsec host MUST include explicit padding bytes in the Integrity Check Value field if these are required to ensure that the header is a multiple of 64 bits.

RFC Text:
As mentioned in Section 2.6, the ICV field may include explicit padding if required to ensure that the AH header is a multiple of 32 bits (IPv4) or 64 bits (IPv6). If padding is required, its length is determined by two factors:

- the length of the ICV
- the IP protocol version (v4 or v6)

For example, if the output of the selected algorithm is 96 bits, no padding is required for IPv4 or IPv6. However, if a different length ICV is generated, due to use of a different algorithm, then padding may be required depending on the length and IP protocol version. The content of the padding field is arbitrarily selected by the sender. (The padding is arbitrary, but need not be random to achieve security.) These padding bytes are included in the ICV calculation, counted as part of the Payload Length, and transmitted at the end of the ICV field to enable the receiver to perform the ICV calculation. Inclusion of padding in excess of the minimum amount required to satisfy IPv4/IPv6 alignment requirements is prohibited

----

Identifier: RQ_002_2031
RFC Clause: 3.3.3.2.1
Type: Optional
Applies to: IPsec host

Requirement:
When sending an IP packet which contains an Authentication Header (AH) with explicit padding bytes in the Integrity Check Value field, an IPsec Host MAY set each padding byte to any 8-bit value

RFC Text:
```
As mentioned in Section 2.6, the ICV field may include explicit
padding if required to ensure that the AH header is a multiple of 32
bits (IPv4) or 64 bits (IPv6).  If padding is required, its length is
determined by two factors:

        - the length of the ICV
        - the IP protocol version (v4 or v6)

For example, if the output of the selected algorithm is 96 bits, no
padding is required for IPv4 or IPv6.  However, if a different length
ICV is generated, due to use of a different algorithm, then padding
may be required depending on the length and IP protocol version.  The
content of the padding field is arbitrarily selected by the sender.
(The padding is arbitrary, but need not be random to achieve
security.)  These padding bytes are included in the ICV calculation,
counted as part of the Payload Length, and transmitted at the end of
the ICV field to enable the receiver to perform the ICV calculation.
Inclusion of padding in excess of the minimum amount required to
satisfy IPv4/IPv6 alignment requirements is prohibited
```

Identifier: RQ_002_2032
RFC Clause: 3.3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When sending an IP packet which contains an Authentication Header (AH) with explicit padding bytes in the Integrity Check Value field, an IPsec Host MUST include the padding bytes in the calculation of the Integrity Check Value

RFC Text:
```
As mentioned in Section 2.6, the ICV field may include explicit
padding if required to ensure that the AH header is a multiple of 32
bits (IPv4) or 64 bits (IPv6).  If padding is required, its length is
determined by two factors:

        - the length of the ICV
        - the IP protocol version (v4 or v6)

For example, if the output of the selected algorithm is 96 bits, no
padding is required for IPv4 or IPv6.  However, if a different length
ICV is generated, due to use of a different algorithm, then padding
may be required depending on the length and IP protocol version.  The
content of the padding field is arbitrarily selected by the sender.
(The padding is arbitrary, but need not be random to achieve
security.)  These padding bytes are included in the ICV calculation,
counted as part of the Payload Length, and transmitted at the end of
the ICV field to enable the receiver to perform the ICV calculation.
Inclusion of padding in excess of the minimum amount required to
satisfy IPv4/IPv6 alignment requirements is prohibited
```

Identifier: RQ_002_2033
RFC Clause: 3.3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When sending an IP packet which contains an Authentication Header (AH) with explicit padding bytes in the Integrity Check Value field, an IPsec Host MUST include the padding bytes in the calculation of the value to be set in the Payload Length field

RFC Text:
```
As mentioned in Section 2.6, the ICV field may include explicit
padding if required to ensure that the AH header is a multiple of 32
bits (IPv4) or 64 bits (IPv6).  If padding is required, its length is
determined by two factors:

        - the length of the ICV
        - the IP protocol version (v4 or v6)

For example, if the output of the selected algorithm is 96 bits, no
padding is required for IPv4 or IPv6.  However, if a different length
ICV is generated, due to use of a different algorithm, then padding
may be required depending on the length and IP protocol version.  The
content of the padding field is arbitrarily selected by the sender.
(The padding is arbitrary, but need not be random to achieve
security.)  These padding bytes are included in the ICV calculation,
counted as part of the Payload Length, and transmitted at the end of
the ICV field to enable the receiver to perform the ICV calculation.
Inclusion of padding in excess of the minimum amount required to
satisfy IPv4/IPv6 alignment requirements is prohibited
```

Identifier: RQ_002_2034
RFC Clause: 3.3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When sending an IP packet which contains an Authentication Header (AH) which requires the inclusion of explicit padding bytes in the Integrity Check Value field, an IPsec Host MUST insert the padding bytes at the end of the Integrity Check Value field

RFC Text:
```
As mentioned in Section 2.6, the ICV field may include explicit
padding if required to ensure that the AH header is a multiple of 32
bits (IPv4) or 64 bits (IPv6).  If padding is required, its length is
determined by two factors:

        - the length of the ICV
        - the IP protocol version (v4 or v6)

For example, if the output of the selected algorithm is 96 bits, no
padding is required for IPv4 or IPv6.  However, if a different length
ICV is generated, due to use of a different algorithm, then padding
may be required depending on the length and IP protocol version.  The
content of the padding field is arbitrarily selected by the sender.
(The padding is arbitrary, but need not be random to achieve
security.)  These padding bytes are included in the ICV calculation,
counted as part of the Payload Length, and transmitted at the end of
the ICV field to enable the receiver to perform the ICV calculation.
Inclusion of padding in excess of the minimum amount required to
satisfy IPv4/IPv6 alignment requirements is prohibited
```

Identifier: RQ_002_2035
RFC Clause: 3.3.3.2.2
Type: Mandatory
Applies to: IPsec host

Requirement:
When sending an IP packet containing an Authentication Header (AH) and the use of Extended Sequence Number (ESN) option is selected, an IPsec Host MUST include the high-order 32-bits of the ESN in the computation of the value to be inserted in the Integrity Check Value field of the packet

RFC Text:
If the ESN option is elected for an SA, then the high-order 32 bits of the ESN must be included in the ICV computation. For purposes of ICV computation, these bits are appended (implicitly) immediately after the end of the payload, and before any implicit packet padding.

For some integrity algorithms, the byte string over which the ICV computation is performed must be a multiple of a blocksize specified by the algorithm. If the IP packet length (including AH and the 32 high-order bits of the ESN, if enabled) does not match the blocksize requirements for the algorithm, implicit padding MUST be appended to the end of the packet, prior to ICV computation. The padding octets MUST have a value of zero. The blocksize (and hence the length of the padding) is specified by the algorithm specification. This padding is not transmitted with the packet. The document that defines an integrity algorithm MUST be consulted to determine if implicit padding is required as described above. If the document does not specify an answer to this, then the default is to assume that implicit padding is required (as needed to match the packet length to the algorithm's blocksize.) If padding bytes are needed but the algorithm does not specify the padding contents, then the padding octets MUST have a value of zero.

Identifier: RQ_002_2036
RFC Clause: 3.3.3.2.2
Type: Mandatory
Applies to: IPsec host

Requirement:
When sending an IP packet containing an Authentication Header (AH) and the packet length does not match the requirements of the selected integrity algorithm, an IPsec Host MUST include the number of bytes necessary to satisfy the algorithm requirements, each with the value zero (0), in the calculation of the value to be set in the Integrity Check Value field

RFC Text:
If the ESN option is elected for an SA, then the high-order 32 bits of the ESN must be included in the ICV computation. For purposes of ICV computation, these bits are appended (implicitly) immediately after the end of the payload, and before any implicit packet padding.

For some integrity algorithms, the byte string over which the ICV computation is performed must be a multiple of a blocksize specified by the algorithm. **If the IP packet length (including AH and the 32 high-order bits of the ESN, if enabled) does not match the blocksize requirements for the algorithm, implicit padding MUST be appended to the end of the packet, prior to ICV computation. The padding octets MUST have a value of zero**. The blocksize (and hence the length of the padding) is specified by the algorithm specification. This padding is not transmitted with the packet. The document that defines an integrity algorithm MUST be consulted to determine if implicit padding is required as described above. If the document does not specify an answer to this, then the default is to assume that implicit padding is required (as needed to match the packet length to the algorithm's blocksize.) If padding bytes are needed but the algorithm does not specify the padding contents, then the padding octets MUST have a value of zero.

Identifier: RQ_002_2037
RFC Clause: 3.3.4
Type: Mandatory
Applies to: IPsec host

Requirement:
When sending an IP packet which needs to be fragmented, an IPsec Host MUST apply Authentication Header processing to the packet before fragmenting it

RFC Text:

If required, IP fragmentation occurs after AH processing within an IPsec implementation. Thus, transport mode AH is applied only to whole IP datagrams (not to IP fragments). An IPv4 packet to which AH has been applied may itself be fragmented by routers en route, and such fragments must be reassembled prior to AH processing at a receiver. (This does not apply to IPv6, where there is no router- initiated fragmentation.) In tunnel mode, AH is applied to an IP packet, the payload of which may be a fragmented IP packet. For example, a security gateway or a "bump-in-the-stack" or "bump-in- the-wire" IPsec implementation (see the Security Architecture document for details) may apply tunnel mode AH to such fragments.

Identifier: RQ_002_2038
RFC Clause: 3.3.4
Type: Optional
Applies to: IPsec host

Requirement:
An IPsec host MAY support the fragmentation of packets containing an Authentication Header

RFC Text:
Fragmentation, whether performed by an IPsec implementation or by routers along the path between IPsec peers, significantly reduces performance. Moreover, the requirement for an AH receiver to accept fragments for reassembly creates denial of service vulnerabilities. **Thus, an AH implementation MAY choose to not support fragmentation** and may mark transmitted packets with the DF bit, to facilitate Path MTU (PMTU) discovery. In any case, an AH implementation MUST support generation of ICMP PMTU messages (or equivalent internal signaling for native host implementations) to minimize the likelihood of fragmentation. Details of the support required for MTU management are contained in the Security Architecture document.

Identifier: RQ_002_2039
RFC Clause: 3.3.4
Type: Optional
Applies to: IPsec host

Requirement:
An IPsec host that does not support the fragmentation of IPv4 packets containing an Authentication Header MAY set the "Do not Fragment (DF)" flag in the packet header.

RFC Text:
Fragmentation, whether performed by an IPsec implementation or by routers along the path between IPsec peers, significantly reduces performance. Moreover, the requirement for an AH receiver to accept fragments for reassembly creates denial of service vulnerabilities. Thus, an AH implementation MAY choose to not support fragmentation **and may mark transmitted packets with the DF bit,** to facilitate Path MTU (PMTU) discovery. In any case, an AH implementation MUST support generation of ICMP PMTU messages (or equivalent internal signaling for native host implementations) to minimize the likelihood of fragmentation. Details of the support required for MTU management are contained in the Security Architecture document.

Identifier: RQ_002_2040
RFC Clause: 3.3.4
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host that supports the use of Authentication Headers MUST also support the generation of ICMP Path MTU messages

RFC Text:
Fragmentation, whether performed by an IPsec implementation or by routers along the path between IPsec peers, significantly reduces performance. Moreover, the requirement for an AH receiver to accept fragments for reassembly creates denial of service vulnerabilities.
Thus, an AH implementation MAY choose to not support fragmentation and may mark transmitted packets with the DF bit, to facilitate Path MTU (PMTU) discovery. **In any case, an AH implementation MUST support generation of ICMP PMTU messages (or equivalent internal signaling for native host implementations) to minimize the likelihood of fragmentation.** Details of the support required for MTU management are contained in the Security Architecture document.

Identifier: RQ_002_2041
RFC Clause: 3.4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host that supports Authentication Headers receives packets which are fragments of a larger packet, it MUST reassemble the fragments into a single packet before processing the Authentication Header if present

RFC Text:
If required, reassembly is performed prior to AH processing. If a packet offered to AH for processing appears to be an IP fragment, i.e., the OFFSET field is nonzero or the MORE FRAGMENTS flag is set, the receiver MUST discard the packet; this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address, and (in IPv6) the Flow ID.

Identifier: RQ_002_2042
RFC Clause: 3.4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host processes the Authentication Header of a received IPv6 packet, it MUST discard the packet if the Offset field in the IPv6 Fragmentation Extension Header contains a non-zero value.

RFC Text:
If required, reassembly is performed prior to AH processing. **If a packet offered to AH for processing appears to be an IP fragment, i.e., the OFFSET field is nonzero** or the MORE FRAGMENTS flag is set, **the receiver MUST discard the packet;** this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address, and (in IPv6) the Flow ID.

Identifier: RQ_002_2043
RFC Clause: 3.4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host processes the Authentication Header of a received IPv4 packet, it MUST discard the packet if the More Fragments flag is set in the IPv4 packet header.

RFC Text:
If required, reassembly is performed prior to AH processing. **If a packet offered to AH for processing appears to be an IP fragment, i.e.,** the OFFSET field is nonzero **or the MORE FRAGMENTS flag is set, the receiver MUST discard the packet;** this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address, and (in IPv6) the Flow ID.

Identifier: RQ_002_2044
RFC Clause: 3.4.1
Type: Recommended
Applies to: IPsec host

Requirement:
If an IPsec host discards an IPv6 packet because it contains an Authentication Header and the Fragmentation Offset field in the Fragmentation Extension Header contains a non-zero value, it SHOULD record the event in a log along with the following parameters:

- SPI value
- date and time of the event
- Source Address
- Destination Address
- the Flow label

RFC Text:
If required, reassembly is performed prior to AH processing. If a packet offered to AH for processing appears to be an IP fragment, i.e., the OFFSET field is nonzero or the MORE FRAGMENTS flag is set, the receiver MUST discard the packet; **this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address, and (in IPv6) the Flow ID**.

Identifier: RQ_002_2045
RFC Clause: 3.4.1
Type: Recommended
Applies to: IPsec host

Requirement:
If an IPsec host discards an IPv4 packet because it contains an Authentication Header and the More Fragments flag is set in the packet header, it SHOULD record the event in a log along with the following parameters:

- SPI value
- date and time of the event
- Source Address
- Destination Address

RFC Text:
If required, reassembly is performed prior to AH processing. If a packet offered to AH for processing appears to be an IP fragment, i.e., the OFFSET field is nonzero or the MORE FRAGMENTS flag is set, the receiver MUST discard the packet; **this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address**, and (in IPv6) the Flow ID.

Identifier: RQ_002_2046
RFC Clause: 3.4.2
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host receives an IPv6 packet containing an Authentication Header but no valid Security Association exists, it MUST discard the packet

RFC Text:
Upon receipt of a packet containing an IP Authentication Header, the receiver determines the appropriate (unidirectional) SA via lookup in the SAD. For a unicast SA, this determination is based on the SPI or the SPI plus protocol field, as described in Section 2.4. If an implementation supports multicast traffic, the destination address is also employed in the lookup (in addition to the SPI), and the sender address also may be employed, as described in Section 2.4. (This process is described in more detail in the Security Architecture document.) The SAD entry for the SA also indicates whether the Sequence Number field will be checked and whether 32- or 64-bit sequence numbers are employed for the SA. The SAD entry for the SA also specifies the algorithm(s) employed for ICV computation, and indicates the key required to validate the ICV.

If no valid Security Association exists for this packet the receiver MUST discard the packet; this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address, and (in IPv6) the Flow ID.

(Note that SA management traffic, such as IKE packets, does not need to be processed based on SPI, i.e., one can de-multiplex this traffic separately based on Next Protocol and Port fields, for example.)

Identifier: RQ_002_2047
RFC Clause: 3.4.2
Type: Recommended
Applies to: IPsec host

Requirement:
If an IPsec host receives an IPv6 packet containing an Authentication Header but no valid Security Association exists, it SHOULD record the event in a log along with the following parameters:

- SPI value
- date and time of the event
- Source Address
- Destination Address
- the Flow label

RFC Text:
Upon receipt of a packet containing an IP Authentication Header, the receiver determines the appropriate (unidirectional) SA via lookup in the SAD. For a unicast SA, this determination is based on the SPI or the SPI plus protocol field, as described in Section 2.4. If an implementation supports multicast traffic, the destination address is also employed in the lookup (in addition to the SPI), and the sender address also may be employed, as described in Section 2.4. (This process is described in more detail in the Security Architecture document.) The SAD entry for the SA also indicates whether the Sequence Number field will be checked and whether 32- or 64-bit sequence numbers are employed for the SA. The SAD entry for the SA also specifies the algorithm(s) employed for ICV computation, and indicates the key required to validate the ICV.

If no valid Security Association exists for this packet the receiver MUST discard the packet; **this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address, and (in IPv6) the Flow ID**.

(Note that SA management traffic, such as IKE packets, does not need to be processed based on SPI, i.e., one can de-multiplex this traffic separately based on Next Protocol and Port fields, for example.)

Identifier: RQ_002_2048
RFC Clause: 3.4.2
Type: Recommended
Applies to: IPsec host

Requirement:
If an IPsec host receives an IPv4 packet containing an Authentication Header but no valid Security Association exists, it SHOULD record the event in a log along with the following parameters:

- SPI value
- date and time of the event
- Source Address
- Destination Address

RFC Text:
Upon receipt of a packet containing an IP Authentication Header, the receiver determines the appropriate (unidirectional) SA via lookup in the SAD. For a unicast SA, this determination is based on the SPI or the SPI plus protocol field, as described in Section 2.4. If an implementation supports multicast traffic, the destination address is also employed in the lookup (in addition to the SPI), and the sender address also may be employed, as described in Section 2.4. (This process is described in more detail in the Security Architecture document.) The SAD entry for the SA also indicates whether the Sequence Number field will be checked and whether 32- or 64-bit sequence numbers are employed for the SA. The SAD entry for the SA also specifies the algorithm(s) employed for ICV computation, and indicates the key required to validate the ICV.

If no valid Security Association exists for this packet the receiver MUST discard the packet; **this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address,** and (in IPv6) the Flow ID.

(Note that SA management traffic, such as IKE packets, does not need to be processed based on SPI, i.e., one can de-multiplex this traffic separately based on Next Protocol and Port fields, for example.)

Identifier: RQ_002_2049
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host that supports the use of Authentication Headers MUST also support the anti-replay service

RFC Text:
All AH implementations MUST support the anti-replay service, though its use may be enabled or disabled by the receiver on a per-SA basis. Anti-replay is applicable to unicast as well as multicast SAs. However, this standard specifies no mechanisms for providing anti- replay for a multi-sender SA (unicast or multicast). In the absence of negotiation (or manual configuration) of an anti-replay mechanism for such an SA, it is recommended that sender and receiver checking of the Sequence Number for the SA be disabled (via negotiation or manual configuration), as noted below.

If the receiver does not enable anti-replay for an SA, no inbound checks are performed on the Sequence Number. However, from the perspective of the sender, the default is to assume that anti-replay is enabled at the receiver. To avoid having the sender do unnecessary sequence number monitoring and SA setup (see Section 3.3.2, "Sequence Number Generation"), if an SA establishment protocol such as IKE is employed, the receiver SHOULD notify the sender, during SA establishment, if the receiver will not provide anti-replay protection.

If the receiver has enabled the anti-replay service for this SA, the receive packet counter for the SA MUST be initialized to zero when the SA is established. For each received packet, the receiver MUST verify that the packet contains a Sequence Number that does not duplicate the Sequence Number of any other packets received during the life of this SA. This SHOULD be the first AH check applied to a packet after it has been matched to an SA, to speed rejection of duplicate packets.

Identifier: RQ_002_2050
RFC Clause: 3.4.3
Type: Optional
Applies to: IPsec host

Requirement:
An IPsec host that supports the use of Authentication Headers MAY enable or disable the anti-replay service on a per-Security Association basis.

RFC Text:
All AH implementations MUST support the anti-replay service, **though its use may be enabled or disabled by the receiver on a per-SA basis.** Anti-replay is applicable to unicast as well as multicast SAs. However, this standard specifies no mechanisms for providing anti- replay for a multi-sender SA (unicast or multicast). In the absence of negotiation (or manual configuration) of an anti-replay mechanism for such an SA, it is recommended that sender and receiver checking of the Sequence Number for the SA be disabled (via negotiation or manual configuration), as noted below.

If the receiver does not enable anti-replay for an SA, no inbound checks are performed on the Sequence Number. However, from the perspective of the sender, the default is to assume that anti-replay is enabled at the receiver. To avoid having the sender do unnecessary sequence number monitoring and SA setup (see Section 3.3.2, "Sequence Number Generation"), if an SA establishment protocol such as IKE is employed, the receiver SHOULD notify the sender, during SA establishment, if the receiver will not provide anti-replay protection.

If the receiver has enabled the anti-replay service for this SA, the receive packet counter for the SA MUST be initialized to zero when the SA is established. For each received packet, the receiver MUST verify that the packet contains a Sequence Number that does not duplicate the Sequence Number of any other packets received during the life of this SA. This SHOULD be the first AH check applied to a packet after it has been matched to an SA, to speed rejection of duplicate packets.

Identifier: RQ_002_2051
RFC Clause: 3.4.3
Type: Recommended
Applies to: IPsec host

Requirement:
If an IPsec host that supports Authentication Headers receives a request to establish a Security Association with another IPsec host (using the IKEv2 protocol for instance) but is unable to provide anti-replay protection, it SHOULD include a notification of this fact in its response to the SA initiator

RFC Text:
All AH implementations MUST support the anti-replay service, though its use may be enabled or disabled by the receiver on a per-SA basis. Anti-replay is applicable to unicast as well as multicast SAs. However, this standard specifies no mechanisms for providing anti- replay for a multi-sender SA (unicast or multicast). In the absence of negotiation (or manual configuration) of an anti-replay mechanism for such an SA, it is recommended that sender and receiver checking of the Sequence Number for the SA be disabled (via negotiation or manual configuration), as noted below.

If the receiver does not enable anti-replay for an SA, no inbound checks are performed on the Sequence Number. However, from the perspective of the sender, the default is to assume that anti-replay is enabled at the receiver. To avoid having the sender do unnecessary sequence number monitoring and SA setup (see Section 3.3.2, "Sequence Number Generation"), **if an SA establishment protocol such as IKE is employed, the receiver SHOULD notify the sender, during SA establishment, if the receiver will not provide anti-replay protection**.

If the receiver has enabled the anti-replay service for this SA, the receive packet counter for the SA MUST be initialized to zero when the SA is established. For each received packet, the receiver MUST verify that the packet contains a Sequence Number that does not duplicate the Sequence Number of any other packets received during the life of this SA. This SHOULD be the first AH check applied to a packet after it has been matched to an SA, to speed rejection of duplicate packets.

Identifier: RQ_002_2052
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host that supports Authentication Headers accepts a request to establish a Security Association with another IPsec host (using the IKEv2 protocol for instance) and it enables the anti-replay service for this SA, it MUST set the received packet counter to zero (0) when the SA is established.

RFC Text:
All AH implementations MUST support the anti-replay service, though its use may be enabled or disabled by the receiver on a per-SA basis. Anti-replay is applicable to unicast as well as multicast SAs. However, this standard specifies no mechanisms for providing anti- replay for a multi-sender SA (unicast or multicast). In the absence of negotiation (or manual configuration) of an anti-replay mechanism for such an SA, it is recommended that sender and receiver checking of the Sequence Number for the SA be disabled (via negotiation or manual configuration), as noted below.

If the receiver does not enable anti-replay for an SA, no inbound checks are performed on the Sequence Number. However, from the perspective of the sender, the default is to assume that anti-replay is enabled at the receiver. To avoid having the sender do unnecessary sequence number monitoring and SA setup (see Section 3.3.2, "Sequence Number Generation"), if an SA establishment protocol such as IKE is employed, the receiver SHOULD notify the sender, during SA establishment, if the receiver will not provide anti-replay protection.

If the receiver has enabled the anti-replay service for this SA, the receive packet counter for the SA MUST be initialized to zero when the SA is established. For each received packet, the receiver MUST verify that the packet contains a Sequence Number that does not duplicate the Sequence Number of any other packets received during the life of this SA. This SHOULD be the first AH check applied to a packet after it has been matched to an SA, to speed rejection of duplicate packets.

Identifier: RQ_002_2053
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host that supports Authentication Headers receives a packet which includes an Authentication Header, it MUSTY reject the packet if the value in the AH Sequence Number field of the received packet is the same as the value in a previous packet received on the same Security Association.

RFC Text:
All AH implementations MUST support the anti-replay service, though its use may be enabled or disabled by the receiver on a per-SA basis. Anti-replay is applicable to unicast as well as multicast SAs. However, this standard specifies no mechanisms for providing anti- replay for a multi-sender SA (unicast or multicast). In the absence of negotiation (or manual configuration) of an anti-replay mechanism for such an SA, it is recommended that sender and receiver checking of the Sequence Number for the SA be disabled (via negotiation or manual configuration), as noted below.

If the receiver does not enable anti-replay for an SA, no inbound checks are performed on the Sequence Number. However, from the perspective of the sender, the default is to assume that anti-replay is enabled at the receiver. To avoid having the sender do unnecessary sequence number monitoring and SA setup (see Section 3.3.2, "Sequence Number Generation"), if an SA establishment protocol such as IKE is employed, the receiver SHOULD notify the sender, during SA establishment, if the receiver will not provide anti-replay protection.

If the receiver has enabled the anti-replay service for this SA, the receive packet counter for the SA MUST be initialized to zero when the SA is established. **For each received packet, the receiver MUST verify that the packet contains a Sequence Number that does not duplicate the Sequence Number of any other packets received during the life of this SA. This SHOULD be the first AH check applied to a packet after it has been matched to an SA, to speed rejection of duplicate packets**.

Identifier: RQ_002_2054
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host that supports Authentication Headers receives a packet which includes an Authentication Header, it MUST be able to check the Sequence Number in that header against the Sequence Numbers of the previous 32 received packets

RFC Text:
Duplicates are rejected through the use of a sliding receive window. How the window is implemented is a local matter, but the following text describes the functionality that the implementation must exhibit.

....

A MINIMUM window size of 32 packets MUST be supported, but a window size of 64 is preferred and SHOULD be employed as the default. Another window size (larger than the MINIMUM) MAY be chosen by the receiver. (The receiver does NOT notify the sender of the window size.) The receive window size should be increased for higher-speed environments, irrespective of assurance issues. Values for minimum and recommended receive window sizes for very high-speed (e.g., multi-gigabit/second) devices are not specified by this standard.

Identifier: RQ_002_2055
RFC Clause: 3.4.3
Type: Recommended
Applies to: IPsec host

Requirement:
When an IPsec host that supports Authentication Headers receives a packet which includes an Authentication Header, it SHOULD be able to check the Sequence Number in that header against the Sequence Numbers of the previous 64 received packets

RFC Text:
Duplicates are rejected through the use of a sliding receive window. How the window is implemented is a local matter, but the following text describes the functionality that the implementation must exhibit.

....

A MINIMUM window size of 32 packets MUST be supported, **but a window size of 64 is preferred and SHOULD be employed as the default**. Another window size (larger than the MINIMUM) MAY be chosen by the receiver. (The receiver does NOT notify the sender of the window size.) The receive window size should be increased for higher-speed environments, irrespective of assurance issues. Values for minimum and recommended receive window sizes for very high-speed (e.g., multi-gigabit/second) devices are not specified by this standard.

Identifier: RQ_002_2056
RFC Clause: 3.4.3
Type: Optional
Applies to: IPsec host

Requirement:
When an IPsec host that supports Authentication Headers receives a packet which includes an Authentication Header, it MAY check the Sequence Number in that header against the Sequence Numbers of more than the previous 64 received packets

RFC Text:
Duplicates are rejected through the use of a sliding receive window. How the window is implemented is a local matter, but the following text describes the functionality that the implementation must exhibit.

....

A MINIMUM window size of 32 packets MUST be supported, but a window size of 64 is preferred and SHOULD be employed as the default. **Another window size (larger than the MINIMUM) MAY be chosen by the receiver**. (The receiver does NOT notify the sender of the window size.) The receive window size should be increased for higher-speed environments, irrespective of assurance issues. Values for minimum and recommended receive window sizes for very high-speed (e.g., multi-gigabit/second) devices are not specified by this standard.

Identifier: RQ_002_2057
RFC Clause: 3.4.4
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host that supports Authentication headers receives an IP packet which includes an Authentication Header, it MUST calculate the Integrity Check Value for the packet using the integrity algorithm specified during the establishment of the relevant Security Association and accept the received packet if this calculated value is the same as the value held in the Integrity Check Value field of the packet

RFC Text:
The receiver computes the ICV over the appropriate fields of the packet, using the specified integrity algorithm, and verifies that it is the same as the ICV included in the ICV field of the packet. Details of the computation are provided below.

If the computed and received ICVs match, then the datagram is valid, and it is accepted. If the test fails, then the receiver MUST discard the received IP datagram as invalid. This is an auditable event. The audit log entry SHOULD include the SPI value, date/time received, Source Address, Destination Address, and (in IPv6) the Flow ID.

Identifier: RQ_002_2058
RFC Clause: 3.4.4
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host that supports Authentication headers receives an IP packet which includes an Authentication Header, it MUST calculate the Integrity Check Value for the packet using the integrity algorithm specified during the establishment of the relevant Security Association and reject the received packet as invalid if this calculated value is not the same as the value held in the Integrity Check Value field of the packet

RFC Text:
The receiver computes the ICV over the appropriate fields of the packet, using the specified integrity algorithm, and verifies that it is the same as the ICV included in the ICV field of the packet. Details of the computation are provided below.

If the computed and received ICVs match, then the datagram is valid, and it is accepted. **If the test fails, then the receiver MUST discard the received IP datagram as invalid.** This is an auditable event. The audit log entry SHOULD include the SPI value, date/time received, Source Address, Destination Address, and (in IPv6) the Flow ID.

Identifier: RQ_002_2059
RFC Clause: 4
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host that supports both Authentication Headers and auditing MUST support auditing of Authentication Headers

RFC Text:

Not all systems that implement AH will implement auditing. **However, if AH is incorporated into a system that supports auditing, then the AH implementation MUST also support auditing** and MUST allow a system administrator to enable or disable auditing for AH. For the most part, the granularity of auditing is a local matter. However, several auditable events are identified in this specification, and for each of these events a minimum set of information that SHOULD be included in an audit log is defined. Additional information also MAY be included in the audit log for each of these events, and additional events, not explicitly called out in this specification, also MAY result in audit log entries. There is no requirement for the receiver to transmit any message to the purported sender in response to the detection of an auditable event, because of the potential to induce denial of service via such action.

Identifier: RQ_002_2060
RFC Clause: 4
Type: Optional
Applies to: IPsec host

Requirement:

When recording an auditable event in its log, an IPsec host MAY include additional parameters to those recommended for the particular event

RFC Text:

Not all systems that implement AH will implement auditing. However, if AH is incorporated into a system that supports auditing, then the AH implementation MUST also support auditing and MUST allow a system administrator to enable or disable auditing for AH. For the most part, the granularity of auditing is a local matter. However, several auditable events are identified in this specification, and for each of these events a minimum set of information that SHOULD be included in an audit log is defined. **Additional information also MAY be included in the audit log for each of these events,** and additional events, not explicitly called out in this specification, also MAY result in audit log entries. There is no requirement for the receiver to transmit any message to the purported sender in response to the detection of an auditable event, because of the potential to induce denial of service via such action.

Identifier: RQ_002_2061
RFC Clause: 4
Type: Optional
Applies to: IPsec host

Requirement:

An IPsec host MAY include AH-related events in its audit log in addition to those specifically required in the support of Authentication Headers

RFC Text:

Not all systems that implement AH will implement auditing. However, if AH is incorporated into a system that supports auditing, then the AH implementation MUST also support auditing and MUST allow a system administrator to enable or disable auditing for AH. For the most part, the granularity of auditing is a local matter. However, several auditable events are identified in this specification, and for each of these events a minimum set of information that SHOULD be included in an audit log is defined. Additional information also MAY be included in the audit log for each of these events, and **additional events, not explicitly called out in this specification, also MAY result in audit log entries**. There is no requirement for the receiver to transmit any message to the purported sender in response to the detection of an auditable event, because of the potential to induce denial of service via such action.

4.3 Requirements extracted from RFC 4303

Identifier: RQ_002_3000
RFC Clause: 1
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host MUST support the ESP security service of integrity only

RFC Text:
Although confidentiality and integrity can be offered independently, ESP typically will employ both services, i.e., packets will be protected with regard to confidentiality and integrity. Thus, there are three possible ESP security service combinations involving these services:

- confidentiality-only (MAY be supported)
- **integrity only (MUST be supported)**
- confidentiality and integrity (MUST be supported)

Identifier: RQ_002_3001
RFC Clause: 1
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host MUST support the ESP security service of confidentiality and integrity

RFC Text:
Although confidentiality and integrity can be offered independently, ESP typically will employ both services, i.e., packets will be protected with regard to confidentiality and integrity. Thus, there are three possible ESP security service combinations involving these services:

- confidentiality-only (MAY be supported)
- integrity only (MUST be supported)
- **confidentiality and integrity (MUST be supported)**

Identifier: RQ_002_3002
RFC Clause: 1
Type: Optional
Applies to: IPsec host

Requirement:
An IPsec host MAY support the ESP security service of confidentiality only

RFC Text:
Although confidentiality and integrity can be offered independently, ESP typically will employ both services, i.e., packets will be protected with regard to confidentiality and integrity. Thus, there are three possible ESP security service combinations involving these services:

- **confidentiality-only (MAY be supported)**
- integrity only (MUST be supported)
- confidentiality and integrity (MUST be supported)

Identifier: RQ_002_3003
RFC Clause: 2.1
Type: Optional
Applies to: IPsec host

Requirement:
An IPsec ESP implementation MAY use the destination address of the received packet in addition to the SPI for SA identification

RFC Text:
Each entry in the SA Database (SAD) (Section 4.4.2) must indicate whether the SA lookup makes use of the destination IP address, or the destination and source IP addresses, in addition to the SPI. For multicast SAs, the protocol field is not employed for SA lookups. For each inbound, IPsec-protected packet, an implementation must conduct its search of the SAD such that it finds the entry that matches the "longest" SA identifier. In **this context, if two or more SAD entries match based on the SPI value, then the entry that also matches based on destination address**, or destination and source address (as indicated in the SAD entry) is the "longest" match.

Identifier: RQ_002_3004
RFC Clause: 2
Type: Mandatory
Applies to: IPsec host

Requirement:
The (outer) protocol header (IPv4, IPv6, or Extension) that immediately precedes the ESP header SHALL contain the value 50 in its Protocol (IPv4) or Next Header (IPv6, Extension) field

RFC Text:
The (outer) protocol header (IPv4, IPv6, or Extension) that immediately precedes the ESP header SHALL contain the value 50 in its Protocol (IPv4) or Next Header (IPv6, Extension) field (see IANA web page at http://www.iana.org/assignments/protocol-numbers). Figure 1 illustrates the top-level format of an ESP packet. The packet begins with two 4-byte fields (Security Parameters Index (SPI) and Sequence Number). Following these fields is the Payload Data, which has substructure that depends on the choice of encryption algorithm and mode, and on the use of TFC padding, which is examined in more detail later. Following the Payload Data are Padding and Pad Length fields, and the Next Header field. The optional Integrity check Value (ICV) field completes the packet.

Identifier: RQ_002_3005
RFC Clause: 2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host sends an ESP packet to an established Security Association, it MUST set a non-zero value in the SPI field of the packet

RFC Text:
The set of SPI values in the range 1 through 255 are reserved by the Internet Assigned Numbers Authority (IANA) for future use; a reserved SPI value will not normally be assigned by IANA unless the use of the assigned SPI value is specified in an RFC. **The SPI value of zero (0) is reserved for local, implementation-specific use and MUST NOT be sent on the wire.** (For example, a key management implementation might use the zero SPI value to mean "No Security Association Exists" during the period when the IPsec implementation has requested that its key management entity establish a new SA, but the SA has not yet been established.)

Identifier: RQ_002_3006
RFC Clause: 2.2
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host sends ESP packets across a unicast Security Association, it MUST increment the value in the sequence number field for every transmitted packet

RFC Text:
This unsigned 32-bit field contains a counter value that increases by one for each packet sent, i.e., a per-SA packet sequence number. **For a unicast SA or a single-sender multicast SA, the sender MUST increment this field for every transmitted packet.** Sharing an SA among multiple senders is permitted, though generally not recommended. ESP provides no means of synchronizing packet counters among multiple senders or meaningfully managing a receiver packet counter and window in the context of multiple senders. Thus, for a multi-sender SA, the anti-replay features of ESP are not available (see Sections 3.3.3 and 3.4.3.)

Identifier: RQ_002_3007
RFC Clause: 2.2
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host sends ESP packets across a single sender multicast Security Association, it MUST increment the value in the sequence number field for every transmitted packet

RFC Text:
This unsigned 32-bit field contains a counter value that increases by one for each packet sent, i.e., a per-SA packet sequence number. **For a unicast SA or a single-sender multicast SA, the sender MUST increment this field for every transmitted packet.** Sharing an SA among multiple senders is permitted, though generally not recommended. ESP provides no means of synchronizing packet counters among multiple senders or meaningfully managing a receiver packet counter and window in the context of multiple senders. Thus, for a multi-sender SA, the anti-replay features of ESP are not available (see Sections 3.3.3 and 3.4.3.)

Identifier: RQ_002_3008
RFC Clause: 2.2
Type: Recommended
Applies to: IPsec host

Requirement:
The sharing of an SA among multiple IPsec ESP senders is NOT RECOMMENDED

RFC Text:
This unsigned 32-bit field contains a counter value that increases by one for each packet sent, i.e., a per-SA packet sequence number. For a unicast SA or a single-sender multicast SA, the sender MUST increment this field for every transmitted packet. **Sharing an SA among multiple senders is permitted, though generally not recommended**. ESP provides no means of synchronizing packet counters among multiple senders or meaningfully managing a receiver packet counter and window in the context of multiple senders. Thus, for a multi-sender SA, the anti-replay features of ESP are not available (see Sections 3.3.3 and 3.4.3.)

Identifier: RQ_002_3009
RFC Clause: 2.2
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host sends an ESP packet across an established Security Association, it MUST include a non-zero value in the Sequence Number field of the packet

RFC Text:

The field is mandatory and MUST always be present even if the receiver does not elect to enable the anti-replay service for a specific SA. Processing of the Sequence Number field is at the discretion of the receiver, but all ESP implementations MUST be capable of performing the processing described in Sections 3.3.3 and 3.4.3. Thus, the sender MUST always transmit this field, but the receiver need not act upon it (see the discussion of Sequence Number Verification in the "Inbound Packet Processing" section (3.4.3) below).

Identifier: RQ_002_3012
RFC Clause: 2.2
Type: Mandatory
Applies to: IPsec host

Requirement:

The value of the sequence number field in the header of the first ESP packet sent across a newly established IPsec Security Association MUST be set to one (1)

RFC Text:

The sender's counter and the receiver's counter are initialized to 0 when an SA is established. (The first packet sent using a given SA will have a sequence number of 1; see Section 3.3.3 for more details on how the sequence number is generated.) If anti-replay is enabled (the default), the transmitted sequence number must never be allowed to cycle. Thus, the sender's counter and the receiver's counter MUST be reset (by establishing a new SA and thus a new key) prior to the transmission of the 2^32nd packet on an SA.

Identifier: RQ_002_3013
RFC Clause: 2.2
Type: Mandatory
Applies to: IPsec host

Requirement:

If an IPsec host increments an ESP packet sequence number to a value greater than can be held in 32 bits, it MUST delete the corresponding Security Association and establish a new one to replace it

RFC Text:

The sender's counter and the receiver's counter are initialized to 0 when an SA is established. (The first packet sent using a given SA will have a sequence number of 1; see Section 3.3.3 for more details on how the sequence number is generated.) **If anti-replay is enabled (the default), the transmitted sequence number must never be allowed to cycle**. Thus, the sender's counter and the receiver's counter MUST be reset (by establishing a new SA and thus a new key) prior to the transmission of the 2^32nd packet on an SA.

Identifier: RQ_002_3014
RFC Clause: 2.2.1
Type: Recommended
Applies to: IPsec host

Requirement:

An IPsec ESP implementation SHOULD implement Extended Sequence Numbers (ESNs)

RFC Text:

To support high-speed IPsec implementations, **Extended Sequence Numbers (ESNs) SHOULD be implemented**, as an extension to the current, 32-bit sequence number field. Use of an ESN MUST be negotiated by an SA management protocol. Note that in IKEv2, this negotiation is implicit; the default is ESN unless 32-bit sequence numbers are explicitly negotiated. (The ESN feature is applicable to multicast as well as unicast SAs.)

Identifier: RQ_002_3015
RFC Clause: 2.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host MUST use a Security Association management protocol to negotiate the use of an Extended Sequence Number (ESN) on a particular ESP SA

RFC Text:
To support high-speed IPsec implementations, Extended Sequence Numbers (ESNs) SHOULD be implemented, as an extension to the current, 32-bit sequence number field. **Use of an ESN MUST be negotiated by an SA management protocol**. Note that in IKEv2, this negotiation is implicit; the default is ESN unless 32-bit sequence numbers are explicitly negotiated. (The ESN feature is applicable to multicast as well as unicast SAs.)

Identifier: RQ_002_3016
RFC Clause: 3.4.3
Type: Recommended
Applies to: IPsec host

Requirement:
An IPsec host SHOULD record in the audit log entry SHOULD the SPI value, date/time received, Source Address, Destination Address, the Sequence Number, and the Flow ID for each failed integrity check event during Sequence Number verification

RFC Text:
If the received packet falls within the window and is not a duplicate, or if the packet is to the right of the window, and if a separate integrity algorithm is employed, then the receiver proceeds to integrity verification. If a combined mode algorithm is employed, the integrity check is performed along with decryption. In either case, if the integrity check fails, the receiver MUST discard the received IP datagram as invalid; this is an auditable event. **The audit log entry for this event SHOULD include the SPI value, date/time received, Source Address, Destination Address, the Sequence Number, and (in IPv6) the Flow ID**. The receive window is updated only if the integrity verification succeeds. (If a combined mode algorithm is being used, then the integrity protected Sequence Number must also match the Sequence Number used for anti-replay protection.)

Identifier: RQ_002_3017
RFC Clause: 2.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When using an Extended Sequence Number (ESN) on a particular ESP Security Association, an IPsec host MUST set the Sequence Number field in the ESP packet of each ESP packet to the low-order 32 bits of the ESN

RFC Text:
The ESN facility allows use of a 64-bit sequence number for an SA. (See Appendix A, "Extended (64-bit) Sequence Numbers", for details.) **Only the low-order 32 bits of the sequence number are transmitted in the plaintext ESP header of each packet,** thus minimizing packet overhead. The high-order 32 bits are maintained as part of the sequence number counter by both transmitter and receiver and are included in the computation of the ICV (if the integrity service is selected). If a separate integrity algorithm is employed, the high order bits are included in the implicit ESP trailer, but are not transmitted, analogous to integrity algorithm padding bits. If a combined mode algorithm is employed, the algorithm choice determines whether the high-order ESN bits are transmitted or are included implicitly in the computation. See Section 3.3.2.2 for processing details.

Identifier: RQ_002_3018
RFC Clause: 2.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP if a separate integrity algorithm is employed the high order bits of the ESN MUST be included in the implicit ESP trailer

RFC Text:
The ESN facility allows use of a 64-bit sequence number for an SA. (See Appendix A, "Extended (64-bit) Sequence Numbers", for details.) Only the low-order 32 bits of the sequence number are transmitted in the plaintext ESP header of each packet, thus minimizing packet overhead. The high-order 32 bits are maintained as part of the sequence number counter by both transmitter and receiver and are included in the computation of the ICV (if the integrity service is selected). **If a separate integrity algorithm is employed, the high order bits are included in the implicit ESP trailer**, but are not transmitted, analogous to integrity algorithm padding bits. If a combined mode algorithm is employed, the algorithm choice determines whether the high-order ESN bits are transmitted or are included implicitly in the computation. See Section 3.3.2.2 for processing details.

Identifier: RQ_002_3019
RFC Clause: 2.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP the implicit ESP trailer MUST NOT be transmitted

RFC Text:
The ESN facility allows use of a 64-bit sequence number for an SA. (See Appendix A, "Extended (64-bit) Sequence Numbers", for details.) Only the low-order 32 bits of the sequence number are transmitted in the plaintext ESP header of each packet, thus minimizing packet overhead. The high-order 32 bits are maintained as part of the sequence number counter by both transmitter and receiver and are included in the computation of the ICV (if the integrity service is selected). If a separate integrity algorithm is employed, **the high order bits are included in the implicit ESP trailer, but are not transmitted**, analogous to integrity algorithm padding bits. If a combined mode algorithm is employed, the algorithm choice determines whether the high-order ESN bits are transmitted or are included implicitly in the computation. See Section 3.3.2.2 for processing details.

Identifier: RQ_002_3021
RFC Clause: 2.3
Type: Mandatory
Applies to: IPsec host

Requirement:
The Payload Data field in an ESP packet MUST be an integral number of bytes in length

RFC Text:
Payload Data is a variable-length field containing data (from the original IP packet) described by the Next Header field. **The Payload Data field is mandatory and is an integral number of bytes in length.** If the algorithm used to encrypt the payload requires cryptographic synchronization data, e.g., an Initialization Vector (IV), then this data is carried explicitly in the Payload field, but it is not called out as a separate field in ESP, i.e., the transmission of an explicit IV is invisible to ESP. (See Figure 2.) Any encryption algorithm that requires such explicit, per-packet synchronization data MUST indicate the length, any structure for such data, and the location of this data as part of an RFC specifying how the algorithm is used with ESP. (Typically, the IV immediately precedes the ciphertext. See Figure 2.) If such synchronization data is implicit, the algorithm for deriving the data MUST be part of the algorithm definition RFC. (If included in the Payload field, cryptographic synchronization data, e.g., an Initialization Vector (IV), usually is not encrypted per se (see Tables 1 and 2), although it sometimes is referred to as being part of the ciphertext.)

Identifier: RQ_002_3022
RFC Clause: 2.3
Type: Mandatory
Applies to: IPsec host

Requirement:
If the algorithm used to encrypt the payload data in an ESP packet requires cryptographic synchronization data this data MUST be carried explicitly in the Payload field

RFC Text:
Payload Data is a variable-length field containing data (from the original IP packet) described by the Next Header field. The Payload Data field is mandatory and is an integral number of bytes in length. **If the algorithm used to encrypt the payload requires cryptographic synchronization data, e.g., an Initialization Vector (IV), then this data is carried explicitly in the Payload field**, but it is not called out as a separate field in ESP, i.e., the transmission of an explicit IV is invisible to ESP. (See Figure 2.) Any encryption algorithm that requires such explicit, per-packet synchronization data MUST indicate the length, any structure for such data, and the location of this data as part of an RFC specifying how the algorithm is used with ESP. (Typically, the IV immediately precedes the ciphertext. See Figure 2.) If such synchronization data is implicit, the algorithm for deriving the data MUST be part of the algorithm definition RFC. (If included in the Payload field, cryptographic synchronization data, e.g., an Initialization Vector (IV), usually is not encrypted per se (see Tables 1 and 2), although it sometimes is referred to as being part of the ciphertext.)

Identifier: RQ_002_3023
RFC Clause: 2.3
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPv4 the beginning of the next layer protocol header MUST be aligned relative to the beginning of the ESP header a multiple of 4 bytes

RFC Text:
Note that the beginning of the next layer protocol header MUST be aligned relative to the beginning of the ESP header as follows. For IPv4, this alignment is a multiple of 4 bytes. For IPv6, the alignment is a multiple of 8 bytes.

Identifier: RQ_002_3024
RFC Clause: 2.3
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPv6 the beginning of the next layer protocol header MUST be aligned relative to the beginning of the ESP header a multiple of 8 bytes

RFC Text:
Note that the beginning of the next layer protocol header MUST be aligned relative to the beginning of the ESP header as follows. For IPv4, this alignment is a multiple of 4 bytes. **For IPv6, the alignment is a multiple of 8 bytes.**

Identifier: RQ_002_3025
RFC Clause: 2.4
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec ESP implementation MUST be able to insert up to 255 bytes of padding immediately after the Payload Data field in an ESP packet

RFC Text:
The sender MAY add 0 to 255 bytes of padding. Inclusion of the Padding field in an ESP packet is optional, subject to the requirements noted above, but **all implementations MUST support generation and consumption of padding**.

o For the purpose of ensuring that the bits to be encrypted are a multiple of the algorithm's block size (first bullet above), the padding computation applies to the Payload Data exclusive of any IV, but including the ESP trailer fields. If a combined algorithm mode requires transmission of the SPI and Sequence Number to effect integrity, e.g., replication of the SPI and Sequence Number in the Payload Data, then the replicated versions of these data items, and any associated, ICV-equivalent data, are included in the computation of the pad length. (If the ESN option is selected, the high-order 32 bits of the ESN also would enter into the computation, if the combined mode algorithm requires their transmission for integrity.)

o For the purposes of ensuring that the ICV is aligned on a 4-byte boundary (second bullet above), the padding computation applies to the Payload Data inclusive of the IV, the Pad Length, and Next Header fields. If a combined mode algorithm is used, any replicated data and ICV-equivalent data are included in the Payload Data covered by the padding computation.

Identifier: RQ_002_3026
RFC Clause: 2.4
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec ESP implementation MUST be able to receive and process ESP packets containing up to 255 bytes of padding immediately following the Payload Data field

RFC Text:
The sender MAY add 0 to 255 bytes of padding. Inclusion of the Padding field in an ESP packet is optional, subject to the requirements noted above, but all implementations MUST support generation and consumption of padding.

o For the purpose of ensuring that the bits to be encrypted are a multiple of the algorithm's block size (first bullet above), the padding computation applies to the Payload Data exclusive of any IV, but including the ESP trailer fields. If a combined algorithm mode requires transmission of the SPI and Sequence Number to effect integrity, e.g., replication of the SPI and Sequence Number in the Payload Data, then the replicated versions of these data items, and any associated, ICV-equivalent data, are included in the computation of the pad length. (If the ESN option is selected, the high-order 32 bits of the ESN also would enter into the computation, if the combined mode algorithm requires their transmission for integrity.)

o For the purposes of ensuring that the ICV is aligned on a 4-byte boundary (second bullet above), the padding computation applies to the Payload Data inclusive of the IV, the Pad Length, and Next Header fields. If a combined mode algorithm is used, any replicated data and ICV-equivalent data are included in the Payload Data covered by the padding computation.

Identifier: RQ_002_3027
RFC Clause: 2.5
Type: Mandatory
Applies to: IPsec host

Requirement:
When sending an ESP packet across an established Security Association, an IPsec host MUST include in the packet a value indicating the number of padding bytes inserted in the packet

RFC Text:
The Pad Length field indicates the number of pad bytes immediately preceding it in the Padding field. The range of valid values is 0 to 255, where a value of zero indicates that no Padding bytes are present. As noted above, this does not include any TFC padding bytes. **The Pad Length field is mandatory.**

Identifier: RQ_002_3029
RFC Clause: 2.6
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host sends a dummy ESP packet, it MUST set the Next Header field to the value fifty-nine (59)

RFC Text:
To facilitate the rapid generation and discarding of the padding traffic in support of traffic flow confidentiality (see Section 2.4), the protocol value 59 (which means "no next header") MUST be used to designate a "dummy" packet. **A transmitter MUST be capable of generating dummy packets marked with this value in the next protocol field**, and a receiver MUST be prepared to discard such packets, without indicating an error. All other ESP header and trailer fields (SPI, Sequence Number, Padding, Pad Length, Next Header, and ICV) MUST be present in dummy packets, but the plaintext portion of the payload, other than this Next Header field, need not be well-formed, e.g., the rest of the Payload Data may consist of only random bytes. Dummy packets are discarded without prejudice.

Identifier: RQ_002_3030
RFC Clause: 2.6
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host receives a packets marked with the protocol value 59 in the next protocol field it MUST discard the packet without indicating an error

RFC Text:
To facilitate the rapid generation and discarding of the padding traffic in support of traffic flow confidentiality (see Section 2.4), the protocol value 59 (which means "no next header") MUST be used to designate a "dummy" packet. A transmitter MUST be capable of generating dummy packets marked with this value in the next protocol field, and **a receiver MUST be prepared to discard such packets, without indicating an error**. All other ESP header and trailer fields (SPI, Sequence Number, Padding, Pad Length, Next Header, and ICV) MUST be present in dummy packets, but the plaintext portion of the payload, other than this Next Header field, need not be well-formed, e.g., the rest of the Payload Data may consist of only random bytes. Dummy packets are discarded without prejudice.

Identifier: RQ_002_3031
RFC Clause: 2.6
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP a dummy packet MUST contain all ESP header and trailer fields (SPI, Sequence Number, Padding, Pad Length, Next Header, and ICV) and the payload field

RFC Text:

To facilitate the rapid generation and discarding of the padding traffic in support of traffic flow confidentiality (see Section 2.4), the protocol value 59 (which means "no next header") MUST be used to designate a "dummy" packet. A transmitter MUST be capable of generating dummy packets marked with this value in the next protocol field, and a receiver MUST be prepared to discard such packets, without indicating an error. **All other ESP header and trailer fields (SPI, Sequence Number, Padding, Pad Length, Next Header, and ICV) MUST be present in dummy packets**, but the plaintext portion of the payload, other than this Next Header field, need not be well-formed, e.g., the rest of the Payload Data may consist of only random bytes. Dummy packets are discarded without prejudice.

Identifier: RQ_002_3032
RFC Clause: 2.7
Type: Recommended
Applies to: IPsec host

Requirement:

An IPsec implementation SHOULD be capable of padding traffic by adding bytes after the end of the ESP Payload Data prior to the beginning of the Padding field

RFC Text:

An IPsec implementation SHOULD be capable of padding traffic by adding bytes after the end of the Payload Data, prior to the beginning of the Padding field. However, this padding (hereafter referred to as TFC padding) can be added only if the Payload Data field contains a specification of the length of the IP datagram. This is always true in tunnel mode, and may be true in transport mode depending on whether the next layer protocol (e.g., IP, UDP, ICMP) contains explicit length information. This length information will enable the receiver to discard the TFC padding, because the true length of the Payload Data will be known. (ESP trailer fields are located by counting back from the end of the ESP packet.) Accordingly, if TFC padding is added, the field containing the specification of the length of the IP datagram MUST NOT be modified to reflect this padding. No requirements for the value of this padding are established by this standard.

Identifier: RQ_002_3033
RFC Clause: 2.7
Type: Mandatory
Applies to: IPsec host

Requirement:

When an IPsec host sends an ESP packet in which TFC padding is added, the field containing the specification of the length of the encapsulated IP datagram MUST NOT be modified to reflect this padding

RFC Text:

An IPsec implementation SHOULD be capable of padding traffic by adding bytes after the end of the Payload Data, prior to the beginning of the Padding field. However, this padding (hereafter referred to as TFC padding) can be added only if the Payload Data field contains a specification of the length of the IP datagram. This is always true in tunnel mode, and may be true in transport mode depending on whether the next layer protocol (e.g., IP, UDP, ICMP) contains explicit length information. This length information will enable the receiver to discard the TFC padding, because the true length of the Payload Data will be known. (ESP trailer fields are located by counting back from the end of the ESP packet.) Accordingly, **if TFC padding is added, the field containing the specification of the length of the IP datagram MUST NOT be modified to reflect this padding**. No requirements for the value of this padding are established by this standard.

Identifier: RQ_002_3035
RFC Clause: 2.7
Type: Recommended
Applies to: IPsec host

Requirement:

An IPsec ESP implementation SHOULD provide local management controls to enable the use of the traffic flow confidentiality capability on a per-SA basis

RFC Text:
Implementations SHOULD provide local management controls to enable the use of this capability on a per-SA basis. The controls should allow the user to specify if this feature is to be used and also provide parametric controls for the feature.

Identifier: RQ_002_3037
RFC Clause: 2.8
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host icludes an Integrity Check Value (ICV) in an ESP packet it MUST be compute the ICV over the ESP header, Payload, and ESP trailer (implicit and explicit) fields

RFC Text:
The Integrity Check Value is a variable-length field computed over the ESP header, Payload, and ESP trailer fields. Implicit ESP trailer fields (integrity padding and high-order ESN bits, if applicable) are included in the ICV computation. The ICV field is optional. It is present only if the integrity service is selected and is provided by either a separate integrity algorithm or a combined mode algorithm that uses an ICV. The length of the field is specified by the integrity algorithm selected and associated with the SA. The integrity algorithm specification MUST specify the length of the ICV and the comparison rules and processing steps for validation.

Identifier: RQ_002_3039
RFC Clause: 3.1
Type: Optional
Applies to: IPsec host

Requirement:
ESP MAY be employed in transport mode

RFC Text:
ESP may be employed in two ways: transport mode or tunnel mode.

Identifier: RQ_002_3040
RFC Clause: 3.1
Type: Optional
Applies to: IPsec host

Requirement:
ESP MAY be employed in tunnel mode.

RFC Text:
ESP may be employed in two ways: transport mode or tunnel mode.

Identifier: RQ_002_3041
RFC Clause: 3.1.1
Type: Recommended
Applies to: IPsec host

Requirement:
When an IPsec host sends an ESP packet it SHOULD place the ESP header in the IPv6 packet after the hop-by-hop, routing, and fragmentation extension headers.

RFC Text:
In the IPv6 context, **ESP is viewed as an end-to-end payload, and thus should appear after hop-by-hop, routing, and fragmentation extension headers.** Destination options extension header(s) could appear before, after, or both before and after the ESP header depending on the semantics desired. However, because ESP protects only fields after the ESP header, it generally will be desirable to place the destination options header(s) after the ESP header. The following diagram illustrates ESP transport mode positioning for a typical IPv6 packet.

Identifier: RQ_002_3042
RFC Clause: 3.1.1
Type: Recommended
Applies to: IPsec host

Requirement:
When an IPsec host sends an ESP packet it SHOULD place the ESP header in the IPv6 packet before any destination options header(s)

RFC Text:
In the IPv6 context, ESP is viewed as an end-to-end payload, and thus should appear after hop-by-hop, routing, and fragmentation extension headers. Destination options extension header(s) could appear before, after, or both before and after the ESP header depending on the semantics desired. However, because ESP protects only fields after the ESP header, **it generally will be desirable to place the destination options header(s) after the ESP header.** The following diagram illustrates ESP transport mode positioning for a typical IPv6 packet.

Identifier: RQ_002_3043
RFC Clause: 3.2
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host MUST select one, or both, of the confidentiality service and integrity service

RFC Text:
The mandatory-to-implement algorithms for use with ESP are described in a separate RFC, to facilitate updating the algorithm requirements independently from the protocol per se. Additional algorithms, beyond those mandated for ESP, MAY be supported. Note that **although both confidentiality and integrity are optional, at least one of these services MUST be selected,** hence both algorithms MUST NOT be simultaneously NULL.

Identifier: RQ_002_3044
RFC Clause: 3.2
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP the algorithms for confidentiality and integrity MUST NOT be simultaneously NULL.

RFC Text:
The mandatory-to-implement algorithms for use with ESP are described in a separate RFC, to facilitate updating the algorithm requirements independently from the protocol per se. Additional algorithms, beyond those mandated for ESP, MAY be supported. Note that although both confidentiality and integrity are optional, at least one of these services MUST be selected, hence **both algorithms MUST NOT be simultaneously NULL.**

Identifier: RQ_002_3045
RFC Clause: 3.2
Type: Optional
Applies to: IPsec host

Requirement:
In IPsec ESP additional algorithms beyond those mandated in RFC4305 for ESP MAY be supported

RFC Text:
The mandatory-to-implement algorithms for use with ESP are described in a separate RFC, to facilitate updating the algorithm requirements independently from the protocol per se. **Additional algorithms, beyond those mandated for ESP, MAY be supported.** Note that although both confidentiality and integrity are optional, at least one of these services MUST be selected, hence both algorithms MUST NOT be simultaneously NULL.

Identifier: RQ_002_3046
RFC Clause: 3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
Each IPv6 packet protected by the ESP confidentiality service MUST carry any data required to allow the receiver to establish cryptographic synchronization for decryption.

RFC Text:
The encryption algorithm employed to protect an ESP packet is specified by the SA via which the packet is transmitted/received. Because IP packets may arrive out of order, and not all packets may arrive (packet loss), **each packet must carry any data required to allow the receiver to establish cryptographic synchronization for decryption.** This data may be carried explicitly in the payload field, e.g., as an IV (as described above), or the data may be derived from the plaintext portions of the (outer IP or ESP) packet header. (Note that if plaintext header information is used to derive an IV, that information may become security critical and thus the protection boundary associated with the encryption process may grow. For example, if one were to use the ESP Sequence Number to derive an IV, the Sequence Number generation logic (hardware or software) would have to be evaluated as part of the encryption algorithm implementation. In the case of FIPS 140-2 [NIST01], this could significantly extend the scope of a cryptographic module evaluation.) Because ESP makes provision for padding of the plaintext, encryption algorithms employed with ESP may exhibit either block or stream mode characteristics. Note that because encryption (confidentiality) MAY be an optional service (e.g., integrity-only ESP), this algorithm MAY be "NULL" [Ken-Arch].

Identifier: RQ_002_3047
RFC Clause: 3.2.1
Type: Optional
Applies to: IPsec host

Requirement:
IPsec ESP synchronsation data MAY be carried explicitly in the payload field

RFC Text:
The encryption algorithm employed to protect an ESP packet is specified by the SA via which the packet is transmitted/received. Because IP packets may arrive out of order, and not all packets may arrive (packet loss), each packet must carry any data required to allow the receiver to establish cryptographic synchronization for decryption. **This data may be carried explicitly in the payload field, e.g., as an IV (as described above), o**r the data may be derived from the plaintext portions of the (outer IP or ESP) packet header. (Note that if plaintext header information is used to derive an IV, that information may become security critical and thus the protection boundary associated with the encryption process may grow. For example, if one were to use the ESP Sequence Number to derive an IV, the Sequence Number generation logic (hardware or software) would have to be evaluated as part of the encryption algorithm implementation. In the case of FIPS 140-2 [NIST01], this could significantly extend the scope of a cryptographic module evaluation.) Because ESP makes provision for padding of the plaintext, encryption algorithms employed with ESP may exhibit either block or stream mode characteristics. Note that because encryption (confidentiality) MAY be an optional service (e.g., integrity-only ESP), this algorithm MAY be "NULL" [Ken-Arch].

Identifier: RQ_002_3048
RFC Clause: 3.2.1
Type: Optional
Applies to: IPsec host

Requirement:
IPsec ESP synchronisaton data may be derived from the plaintext portions of the (outer IP or ESP) packet header.

RFC Text:
The encryption algorithm employed to protect an ESP packet is specified by the SA via which the packet is transmitted/received. Because IP packets may arrive out of order, and not all packets may arrive (packet loss), each packet must carry any data required to allow the receiver to establish cryptographic synchronization for decryption. **This data may be carried explicitly in the payload field, e.g., as an IV (as described above), or the data may be derived from the plaintext portions of the (outer IP or ESP) packet header. (**Note that if plaintext header information is used to derive an IV, that information may become security critical and thus the protection boundary associated with the encryption process may grow. For example, if one were to use the ESP Sequence Number to derive an IV, the Sequence Number generation logic (hardware or software) would have to be evaluated as part of the encryption algorithm implementation. In the case of FIPS 140-2 [NIST01], this could significantly extend the scope of a cryptographic module evaluation.) Because ESP makes provision for padding of the plaintext, encryption algorithms employed with ESP may exhibit either block or stream mode characteristics. Note that because encryption (confidentiality) MAY be an optional service (e.g., integrity-only ESP), this algorithm MAY be "NULL" [Ken-Arch].

Identifier: RQ_002_3049
RFC Clause: 3.2.2
Type: Mandatory
Applies to: IPsec host

Requirement:
Any integrity algorithm employed with ESP MUST make provisions to permit processing of packets that arrive out of order and to accommodate packet loss

RFC Text:
The integrity algorithm employed for the ICV computation is specified by the SA via which the packet is transmitted/received. As was the case for encryption algorithms, **any integrity algorithm employed with ESP must make provisions to permit processing of packets that arrive out of order and to accommodate packet loss.** The same admonition noted above applies to use of any plaintext data to facilitate receiver synchronization of integrity algorithms. Note that because the integrity service MAY be optional, this algorithm may be "NULL".

Identifier: RQ_002_3050
RFC Clause: 3.3.3
Type: Mandatory
Applies to: IPsec host

Requirement:
When establishing an ESP Security Association, an IPsec HOST MUST set the value zero (0) into the Sequence Number field of the ESP packet

RFC Text:
The sender's counter is initialized to 0 when an SA is established. The sender increments the sequence number (or ESN) counter for this SA and inserts the low-order 32 bits of the value into the Sequence Number field. Thus, the first packet sent using a given SA will contain a sequence number of 1.

Identifier: RQ_002_3051
RFC Clause: 3.3.3
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host increments an ESP packet sequence number to a value greater than can be held in 32 bits AND if Extended Sequence Numbers (ESN) are NOT employed, it MUST delete the corresponding Security Association and establish a new one to replace it.

RFC Text:
If anti-replay is enabled (the default), the sender checks to ensure that the counter has not cycled before inserting the new value in the Sequence Number field. In other words, **the sender MUST NOT send a packet on an SA if doing so would cause the sequence number to cycle.** An attempt to transmit a packet that would result in sequence number overflow is an auditable event. The audit log entry for this event SHOULD include the SPI value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.

Identifier: RQ_002_3052
RFC Clause: 3.3.3
Type: Recommended
Applies to: IPsec host

Requirement:
When an ESP sequence number overflows the IPsec host SHOULD record in the audit log the SPI value, current date/time, Source Address, Destination Address, and the cleartext Flow ID.

RFC Text:
If anti-replay is enabled (the default), the sender checks to ensure that the counter has not cycled before inserting the new value in the Sequence Number field. In other words, the sender MUST NOT send a packet on an SA if doing so would cause the sequence number to cycle. An attempt to transmit a packet that would result in sequence number overflow is an auditable event. **The audit log entry for this event SHOULD include the SPI value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.**

Identifier: RQ_002_3053
RFC Clause: 3.3.3
Type: Recommended
Applies to: IPsec host

Requirement:
If an IPsec host manually distributes the key used to compute an ESP ICV it SHOULD NOT provide anti-replay service.

RFC Text:
If the key used to compute an ICV is manually distributed, a compliant implementation SHOULD NOT provide anti-replay service. If a user chooses to employ anti-replay in conjunction with SAs that are manually keyed, the sequence number counter at the sender MUST be correctly maintained across local reboots, etc., until the key is replaced. (See Section 5.)

Identifier: RQ_002_3054
RFC Clause: 3.3.3
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host is configure to use anti-replay in conjunction with ESP SAs that are manually keyed, it MUST correctly maintain the sequence number counter across local reboots, etc., until the key is replaced.

RFC Text:
If the key used to compute an ICV is manually distributed, a compliant implementation SHOULD NOT provide anti-replay service. **If a user chooses to employ anti-replay in conjunction with SAs that are manually keyed, the sequence number counter at the sender MUST be correctly maintained across local reboots, etc., until the key is replaced. (See Section 5.)**

Identifier: RQ_002_3055
RFC Clause: 3.3.3
Type: Optional
Applies to: IPsec host

Requirement:
If the anti-replay service is disabled an IPsec host MAY not need to monitor or reset the Sequence Number counter used in ESP packets.

RFC Text:
If anti-replay is disabled (as noted above), the sender does not need to monitor or reset the counter. However, the sender still increments the counter and when it reaches the maximum value, the counter rolls over back to zero. (This behavior is recommended for multi-sender, multicast SAs, unless anti-replay mechanisms outside the scope of this standard are negotiated between the sender and receiver.)

Identifier: RQ_002_3056
RFC Clause: 3.3.3
Type: Recommended
Applies to: IPsec host

Requirement:
If an IPsec host receives a request to establish an ESP Security Association but is configured such that the anti-replay service is not enabled, it SHOULD NOT attempt to negotiate ESN in an SA management protocol

RFC Text:
Note: **If a receiver chooses to not enable anti-replay for an SA, then the receiver SHOULD NOT negotiate ESN in an SA management protocol**. Use of ESN creates a need for the receiver to manage the anti-replay window (in order to determine the correct value for the high-order bits of the ESN, which are employed in the ICV computation), which is generally contrary to the notion of disabling anti-replay for an SA.

Identifier: RQ_002_3057
RFC Clause: 3.3.4
Type: Optional
Applies to: IPsec host

Requirement:
An ESP implementation MAY choose not to support fragmentation and may mark transmitted packets with the DF bit to facilitate Path MTU discovery

RFC Text:
Fragmentation, whether performed by an IPsec implementation or by routers along the path between IPsec peers, significantly reduces performance. Moreover, the requirement for an ESP receiver to accept fragments for reassembly creates denial of service vulnerabilities. Thus, **an ESP implementation MAY choose to not support fragmentation and may mark transmitted packets with the DF bit, to facilitate Path MTU (PMTU) discovery**. In any case, an ESP implementation MUSTsupport generation of ICMP PMTU messages (or equivalent internal signaling for native host implementations) to minimize the likelihood of fragmentation. Details of the support required for MTU management are contained in the Security Architecture document.

Identifier: RQ_002_3058
RFC Clause: 3.3.4
Type: Mandatory
Applies to: IPsec host

Requirement:
An ESP implementation MUSTsupport generation of ICMP PMTU messages to minimize the likelihood of fragmentation.

RFC Text:
Fragmentation, whether performed by an IPsec implementation or by routers along the path between IPsec peers, significantly reduces performance. Moreover, the requirement for an ESP receiver to accept fragments for reassembly creates denial of service vulnerabilities. Thus, an ESP implementation MAY choose to not support fragmentation and may mark transmitted packets with the DF bit, to facilitate Path MTU (PMTU) discovery. In any case, **an ESP implementation MUSTsupport generation of ICMP PMTU messages (or equivalent internal signaling for native host implementations) to minimize the likelihood of fragmentation**. Details of the support required for MTU management are contained in the Security Architecture document.

Identifier: RQ_002_3059
RFC Clause: 3.4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
If a packet offered to ESP for processing appears to be an IP fragment the receiver MUST discard the packet

RFC Text:
If required, reassembly is performed prior to ESP processing. **If a packet offered to ESP for processing appears to be an IP fragment, i.e., the OFFSET field is non-zero or the MORE FRAGMENTS flag is set, the receiver MUST discard the packet**; this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time received, Source Address, Destination Address, Sequence Number, and (in IPv6) the Flow ID.

Identifier: RQ_002_3060
RFC Clause: 3.4.1
Type: Recommended
Applies to: IPsec host

Requirement:
An IPsec host SHOULD record in the audit log the SPI value, date/time received, Source Address, Destination Address, Sequence Number, and the Flow ID for each packet defragmentation event

RFC Text:
If required, reassembly is performed prior to ESP processing. If a packet offered to ESP for processing appears to be an IP fragment, i.e., the OFFSET field is non-zero or the MORE FRAGMENTS flag is set, the receiver MUST discard the packet; this is an auditable event. **The audit log entry for this event SHOULD include the SPI value, date/time received, Source Address, Destination Address, Sequence Number, and (in IPv6) the Flow ID**.

Identifier: RQ_002_3061
RFC Clause: 3.4.2
Type: Mandatory
Applies to: IPsec host

Requirement:
If no valid Security Association exists for a received ESP packet the receiving IPsec host MUST discard the packet

RFC Text:
If no valid Security Association exists for this packet, the receiver MUST discard the packet; this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time received, Source Address, Destination Address, Sequence Number, and (in IPv6) the cleartext Flow ID.

Identifier: RQ_002_3062
RFC Clause: 3.4.2
Type: Recommended
Applies to: IPsec host

Requirement:
An IPsec host SHOULD record in the audit log the SPI value, date/time received, Source Address, Destination Address, Sequence Number, and (in IPv6) the cleartext Flow ID for each Security Association lookup failure event

RFC Text:
If no valid Security Association exists for this packet, the receiver MUST discard the packet; this is an auditable event. **The audit log entry for this event SHOULD include the SPI value, date/time received, Source Address, Destination Address, Sequence Number, and (in IPv6) the cleartext Flow ID.**

Identifier: RQ_002_3063
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
All ESP implementations MUST support the anti-replay service

RFC Text:
All ESP implementations MUST support the anti-replay service, though its use may be enabled or disabled by the receiver on a per-SA basis. This service MUST NOT be enabled unless the ESP integrity service also is enabled for the SA, because otherwise the Sequence Number field has not been integrity protected. Anti-replay is applicable to unicast as well as multicast SAs. However, this standard specifies no mechanisms for providing anti-replay for a multi-sender SA (unicast or multicast). In the absence of negotiation (or manual configuration) of an anti-replay mechanism for such an SA, it is recommended that sender and receiver checking of the sequence number for the SA be disabled (via negotiation or manual configuration), as noted below.

Identifier: RQ_002_3064
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host ESP MUST NOT enable the anti-replay service unless the ESP integrity service also is enabled for the SA

RFC Text:
All ESP implementations MUST support the anti-replay service, though its use may be enabled or disabled by the receiver on a per-SA basis. **This service MUST NOT be enabled unless the ESP integrity service also is enabled for the SA**, because otherwise the Sequence Number field has not been integrity protected. Anti-replay is applicable to unicast as well as multicast SAs. However, this standard specifies no mechanisms for providing anti-replay for a multi-sender SA (unicast or multicast). In the absence of negotiation (or manual configuration) of an anti-replay mechanism for such an SA, it is recommended that sender and receiver checking of the sequence number for the SA be disabled (via negotiation or manual configuration), as noted below.

Identifier: RQ_002_3065
RFC Clause: 3.4.3
Type: Recommended
Applies to: IPsec host

Requirement:
An IPsec host SHOULD disable the checking of sequence numbers on multi-sender ESP SAs (unicast or multicast)

RFC Text:
All ESP implementations MUST support the anti-replay service, though its use may be enabled or disabled by the receiver on a per-SA basis. This service MUST NOT be enabled unless the ESP integrity service also is enabled for the SA, because otherwise the Sequence Number field has not been integrity protected. Anti-replay is applicable to unicast as well as multicast SAs. However, this standard specifies no mechanisms for providing anti-replay for a multi-sender SA (unicast or multicast). **In the absence of negotiation (or manual configuration) of an anti-replay mechanism for such an SA, it is recommended that sender and receiver checking of the sequence number for the SA be disabled** (via negotiation or manual configuration), as noted below.

Identifier: RQ_002_3066
RFC Clause: 3.4.3
Type: Recommended
Applies to: IPsec host

Requirement:
When an IPsec host receives a request to establish an ESP Security Association it SHOULD notify the initiator if it is unable to provide anti-replay protection.

RFC Text:
If the receiver does not enable anti-replay for an SA, no inbound checks are performed on the Sequence Number. However, from the perspective of the sender, the default is to assume that anti-replay is enabled at the receiver. To avoid having the sender do unnecessary sequence number monitoring and SA setup (see section 3.3.3), if an SA establishment protocol is employed, **the receiver SHOULD notify the sender, during SA establishment, if the receiver will not provide anti-replay protection.**

Identifier: RQ_002_3067
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP during sequence number verification if the receiver has enabled the anti-replay service for this SA the receive packet counter for the SA MUST be initialized to zero when the SA is established.

RFC Text:
If the receiver has enabled the anti-replay service for this SA, the receive packet counter for the SA MUST be initialized to zero when the SA is established. For each received packet, the receiver MUST verify that the packet contains a Sequence Number that does not duplicate the Sequence Number of any other packets received during the life of this SA. This SHOULD be the first ESP check applied to a packet after it has been matched to an SA, to speed rejection of duplicate packets.

Identifier: RQ_002_3068
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host that has anti-replay service enabled receives an ESP packet, it MUST verify that the value set in the Sequence Number field of the incoming ESP header does not duplicate the Sequence Number of any other packets received during the life of the corresponding SA

RFC Text:
If the receiver has enabled the anti-replay service for this SA, the receive packet counter for the SA MUST be initialized to zero when the SA is established. **For each received packet, the receiver MUST verify that the packet contains a Sequence Number that does not duplicate the Sequence Number of any other packets received during the life of this SA**. This SHOULD be the first ESP check applied to a packet after it has been matched to an SA, to speed rejection of duplicate packets.

Identifier: RQ_002_3070
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host that has the anti-replay service enabled MUST support a minimum window size of 32 packets when 32-bit sequence numbers are employed (i.e. when ESN is NOTenabled)

RFC Text:
A minimum window size of 32 packets MUST be supported when 32-bit sequence numbers are employed; a window size of 64 is preferred and SHOULD be employed as the default. Another window size (larger than the minimum) MAY be chosen by the receiver. (The receiver does NOT notify the sender of the window size.) The receive window size should be increased for higher-speed environments, irrespective of assurance issues. Values for minimum and recommended receive window sizes for very high-speed (e.g., multi-gigabit/second) devices are not specified by this standard.

Identifier: RQ_002_3071
RFC Clause: 3.4.3
Type: Recommended
Applies to: IPsec host

Requirement:
An IPsec host that has enabled the anti-replay service SHOULD implement a default window size of 64 packets

RFC Text:
A minimum window size of 32 packets MUST be supported when 32-bit sequence numbers are employed; **a window size of 64 is preferred and SHOULD be employed as the default**. Another window size (larger than the minimum) MAY be chosen by the receiver. (The receiver does NOT notify the sender of the window size.) The receive window size should be increased for higher-speed environments, irrespective of assurance issues. Values for minimum and recommended receive window sizes for very high-speed (e.g., multi-gigabit/second) devices are not specified by this standard.

Identifier: RQ_002_3072
RFC Clause: 3.4.3
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP during sequence number verification where anti-replay is enabled duplicate ESP packet MUST be detected and rejected through the use of a sliding receive window

RFC Text:
Duplicates are rejected through the use of a sliding receive window. How the window is implemented is a local matter, but the following text describes the functionality that the implementation must exhibit.

Identifier: RQ_002_3077
RFC Clause: 3.4.4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP during Integrity Check Value verification where separate confidentiality and integrity algorithms are employed the receiver MUST compute the ICV over the ESP packet minus the ICV (using the specified integrity algorithm) and verify that it is the same as the ICV carried in the packet

RFC Text:

```
3.4.4.  Integrity Check Value Verification

   As with outbound processing, there are several options for inbound
   processing, based on features of the algorithms employed.

3.4.4.1.  Separate Confidentiality and Integrity Algorithms
```

If separate confidentiality and integrity algorithms are employed processing proceeds as follows:

1. **If integrity has been selected, the receiver computes the ICV over the ESP packet minus the ICV, using the specified integrity algorithm and verifies that it is the same as the ICV carried in the packet. D**etails of the computation are provided below.

   If the computed and received ICVs match, then the datagram is valid, and it is accepted. If the test fails, then the receiver MUST discard the received IP datagram as invalid; this is an auditable event. The log data SHOULD include the SPI value, date/time received, Source Address, Destination Address, the Sequence Number, and (for IPv6) the cleartext Flow ID.

   Implementation Note:

   Implementations can use any set of steps that results in the same result as the following set of steps. Begin by removing and saving the ICV field. Next check the overall length of the ESP packet minus the ICV field. If implicit padding is required, based on the block size of the integrity algorithm, append zero-filled bytes to the end of the ESP packet directly after the Next Header field, or after the high-order 32 bits of the sequence number if ESN is selected. Perform the ICV computation and compare the result with the saved value, using the comparison rules defined by the algorithm specification.

2. The receiver decrypts the ESP Payload Data, Padding, Pad Length, and Next Header using the key, encryption algorithm, algorithm mode, and cryptographic synchronization data (if any), indicated by the SA. As in Section 3.3.2, we speak here in terms of encryption always being applied because of the formatting implications. This is done with the understanding that "no confidentiality" is offered by using the NULL encryption algorithm (RFC 2410).

   - If explicit cryptographic synchronization data, e.g., an IV, is indicated, it is taken from the Payload field and input to the decryption algorithm as per the algorithm specification.

   - If implicit cryptographic synchronization data is indicated, a local version of the IV is constructed and input to the decryption algorithm as per the algorithm specification.

3. The receiver processes any Padding as specified in the encryption algorithm specification. If the default padding scheme (see Section 2.4) has been employed, the receiver SHOULD inspect the Padding field before removing the padding prior to passing the decrypted data to the next layer.

4. The receiver checks the Next Header field. If the value is "59" (no next header), the (dummy) packet is discarded without further processing.

```
   5. The receiver reconstructs the original IP datagram from:

           - for transport mode -- outer IP header plus the
             original next layer protocol information in the ESP
             Payload field
           - for tunnel mode -- the entire IP datagram in the ESP
             Payload field.

      The exact steps for reconstructing the original datagram
      depend on the mode (transport or tunnel) and are described
      in the Security Architecture document.  At a minimum, in an
      IPv6 context, the receiver SHOULD ensure that the decrypted
      data is 8-byte aligned, to facilitate processing by the
      protocol identified in the Next Header field.  This
      processing "discards" any (optional) TFC padding that has
      been added for traffic flow confidentiality.  (If present,
      this will have been inserted after the IP datagram (or
      transport-layer frame) and before the Padding field (see
      Section 2.4).)

--------------------
```

Identifier: RQ_002_3078
RFC Clause: 3.4.4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host receives an ESP packet and is using separate confidentiality and integrity algorithms it MUST discard the received IP datagram if the value in the ICV field of the received packet does NOTmatch its own computed value

RFC Text:
```
3.4.4.  Integrity Check Value Verification

   As with outbound processing, there are several options for inbound
   processing, based on features of the algorithms employed.

3.4.4.1.  Separate Confidentiality and Integrity Algorithms

   If separate confidentiality and integrity algorithms are employed
   processing proceeds as follows:

         1. If integrity has been selected, the receiver computes the
            ICV over the ESP packet minus the ICV, using the specified
            integrity algorithm and verifies that it is the same as the
            ICV carried in the packet.  Details of the computation are
            provided below.

            If the computed and received ICVs match, then the datagram
            is valid, and it is accepted.  If the test fails, then the
            receiver MUST discard the received IP datagram as invalid;
            this is an auditable event.  The log data SHOULD include the
            SPI value, date/time received, Source Address, Destination
            Address, the Sequence Number, and (for IPv6) the cleartext
            Flow ID.

            Implementation Note:

            Implementations can use any set of steps that results in the
            same result as the following set of steps.  Begin by
            removing and saving the ICV field.  Next check the overall
            length of the ESP packet minus the ICV field.  If implicit
            padding is required, based on the block size of the
            integrity algorithm, append zero-filled bytes to the end of
            the ESP packet directly after the Next Header field, or
            after the high-order 32 bits of the sequence number if ESN
            is selected.  Perform the ICV computation and compare the
            result with the saved value, using the comparison rules
            defined by the algorithm specification.

         2. The receiver decrypts the ESP Payload Data, Padding, Pad
            Length, and Next Header using the key, encryption algorithm,
            algorithm mode, and cryptographic synchronization data (if
            any), indicated by the SA.  As in Section 3.3.2, we speak
            here in terms of encryption always being applied because of
            the formatting implications.  This is done with the
            understanding that "no confidentiality" is offered by using
            the NULL encryption algorithm (RFC 2410).

                 - If explicit cryptographic synchronization data, e.g.,
                   an IV, is indicated, it is taken from the Payload
                   field and input to the decryption algorithm as per
                   the algorithm specification.

                 - If implicit cryptographic synchronization data is
                   indicated, a local version of the IV is constructed
                   and input to the decryption algorithm as per the
                   algorithm specification.

         3. The receiver processes any Padding as specified in the
            encryption algorithm specification.  If the default padding
            scheme (see Section 2.4) has been employed, the receiver
            SHOULD inspect the Padding field before removing the padding
            prior to passing the decrypted data to the next layer.

         4. The receiver checks the Next Header field.  If the value is
            "59" (no next header), the (dummy) packet is discarded
            without further processing.
```

```
5. The receiver reconstructs the original IP datagram from:

        - for transport mode -- outer IP header plus the
          original next layer protocol information in the ESP
          Payload field
        - for tunnel mode -- the entire IP datagram in the ESP
          Payload field.

   The exact steps for reconstructing the original datagram
   depend on the mode (transport or tunnel) and are described
   in the Security Architecture document.  At a minimum, in an
   IPv6 context, the receiver SHOULD ensure that the decrypted
   data is 8-byte aligned, to facilitate processing by the
   protocol identified in the Next Header field.  This
   processing "discards" any (optional) TFC padding that has
   been added for traffic flow confidentiality.  (If present,
   this will have been inserted after the IP datagram (or
   transport-layer frame) and before the Padding field (see
   Section 2.4).)
```

Identifier: RQ_002_3079
RFC Clause: 3.4.4.1
Type: Recommended
Applies to: IPsec host

Requirement:
If an IPsec host receives an ESP packet and is using separate confidentiality and integrity algorithms it SHOULD record the SPI value, date/time received, Source Address, Destination Address, the Sequence Number, and the cleartext Flow ID in the audit log for each ICV mismatch event

RFC Text:

```
3.4.4.  Integrity Check Value Verification

   As with outbound processing, there are several options for inbound
   processing, based on features of the algorithms employed.

3.4.4.1.  Separate Confidentiality and Integrity Algorithms

   If separate confidentiality and integrity algorithms are employed
   processing proceeds as follows:

         1. If integrity has been selected, the receiver computes the
            ICV over the ESP packet minus the ICV, using the specified
            integrity algorithm and verifies that it is the same as the
            ICV carried in the packet.  Details of the computation are
            provided below.

            If the computed and received ICVs match, then the datagram
            is valid, and it is accepted.  If the test fails, then the
            receiver MUST discard the received IP datagram as invalid;
            this is an auditable event.  The log data SHOULD include the
            SPI value, date/time received, Source Address, Destination
            Address, the Sequence Number, and (for IPv6) the cleartext
            Flow ID.

            Implementation Note:

            Implementations can use any set of steps that results in the
            same result as the following set of steps.  Begin by
            removing and saving the ICV field.  Next check the overall
            length of the ESP packet minus the ICV field.  If implicit
            padding is required, based on the block size of the
            integrity algorithm, append zero-filled bytes to the end of
            the ESP packet directly after the Next Header field, or
            after the high-order 32 bits of the sequence number if ESN
            is selected.  Perform the ICV computation and compare the
            result with the saved value, using the comparison rules
            defined by the algorithm specification.
```

```
   2. The receiver decrypts the ESP Payload Data, Padding, Pad
      Length, and Next Header using the key, encryption algorithm,
      algorithm mode, and cryptographic synchronization data (if
      any), indicated by the SA.  As in Section 3.3.2, we speak
      here in terms of encryption always being applied because of
      the formatting implications.  This is done with the
      understanding that "no confidentiality" is offered by using
      the NULL encryption algorithm (RFC 2410).

              - If explicit cryptographic synchronization data, e.g.,
                an IV, is indicated, it is taken from the Payload
                field and input to the decryption algorithm as per
                the algorithm specification.

              - If implicit cryptographic synchronization data is
                indicated, a local version of the IV is constructed
                and input to the decryption algorithm as per the
                algorithm specification.

   3. The receiver processes any Padding as specified in the
      encryption algorithm specification.  If the default padding
      scheme (see Section 2.4) has been employed, the receiver
      SHOULD inspect the Padding field before removing the padding
      prior to passing the decrypted data to the next layer.

   4. The receiver checks the Next Header field.  If the value is
      "59" (no next header), the (dummy) packet is discarded
      without further processing.

   5. The receiver reconstructs the original IP datagram from:

              - for transport mode -- outer IP header plus the
                original next layer protocol information in the ESP
                Payload field
              - for tunnel mode -- the entire IP datagram in the ESP
                Payload field.

      The exact steps for reconstructing the original datagram
      depend on the mode (transport or tunnel) and are described
      in the Security Architecture document.  At a minimum, in an
      IPv6 context, the receiver SHOULD ensure that the decrypted
      data is 8-byte aligned, to facilitate processing by the
      protocol identified in the Next Header field.  This
      processing "discards" any (optional) TFC padding that has
      been added for traffic flow confidentiality.  (If present,
      this will have been inserted after the IP datagram (or
      transport-layer frame) and before the Padding field (see
      Section 2.4).)
```

Identifier: RQ_002_3080
RFC Clause: 3.4.4.1
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host receives an ESP packet and is using separate confidentiality and integrity algorithms it MUST complete integrity checking before the decrypted packet is passed on for further processing

RFC Text:

If integrity checking and encryption are performed in parallel, integrity checking MUST be completed before the decrypted packet is passed on for further processing. This order of processing facilitates rapid detection and rejection of replayed or bogus packets by the receiver, prior to decrypting the packet, hence potentially reducing the impact of denial of service attacks.

Note: If the receiver performs decryption in parallel with integrity checking, care must be taken to avoid possible race conditions with regard to packet access and extraction of the decrypted packet.

Identifier: RQ_002_3083
RFC Clause: 3.4.4.2
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host receives an ESP packet and is using combined confidentiality and integrity algorithms it MUST discard the received IP datagram if the value in the ICV field of the received packet DOES NOT match its own computed value

RFC Text:
```
3.4.4.2.  Combined Confidentiality and Integrity Algorithms

   If a combined confidentiality and integrity algorithm is employed,
   then the receiver proceeds as follows:

         1. Decrypts and integrity checks the ESP Payload Data, Padding,
            Pad Length, and Next Header, using the key, algorithm,
            algorithm mode, and cryptographic synchronization data (if
            any), indicated by the SA.  The SPI from the ESP header, and
            the (receiver) packet counter value (adjusted as required
            from the processing described in Section 3.4.3) are inputs
            to this algorithm, as they are required for the integrity
            check.

                 - If explicit cryptographic synchronization data, e.g.,
                   an IV, is indicated, it is taken from the Payload
                   field and input to the decryption algorithm as per
                   the algorithm specification.

                 - If implicit cryptographic synchronization data, e.g.,
                   an IV, is indicated, a local version of the IV is
                   constructed and input to the decryption algorithm as
                   per the algorithm specification.

         2. If the integrity check performed by the combined mode
            algorithm fails, the receiver MUST discard the received IP
            datagram as invalid; this is an auditable event.  The log
            data SHOULD include the SPI value, date/time received,
            Source Address, Destination Address, the Sequence Number,
            and (in IPv6) the cleartext Flow ID.

         3. Process any Padding as specified in the encryption algorithm
            specification, if the algorithm has not already done so.

         4. The receiver checks the Next Header field.  If the value is
            "59" (no next header), the (dummy) packet is discarded
            without further processing.

         5. Extract the original IP datagram (tunnel mode) or
            transport-layer frame (transport mode) from the ESP Payload
            Data field.  This implicitly discards any (optional) padding
            that has been added for traffic flow confidentiality.  (If
            present, the TFC padding will have been inserted after the
            IP payload and before the Padding field (see Section 2.4).)
```

Identifier: RQ_002_3084
RFC Clause: 3.4.4.2
Type: Recommended
Applies to: IPsec host

Requirement:

If an IPsec host receives an ESP packet and is using combined confidentiality and integrity algorithms it SHOULD record the SPI value, date/time received, Source Address, Destination Address, the Sequence Number, and the cleartext Flow ID values in the audit log for each ICV verification failure event

RFC Text:

```
3.4.4.2.  Combined Confidentiality and Integrity Algorithms

   If a combined confidentiality and integrity algorithm is employed,
   then the receiver proceeds as follows:

         1. Decrypts and integrity checks the ESP Payload Data, Padding,
            Pad Length, and Next Header, using the key, algorithm,
            algorithm mode, and cryptographic synchronization data (if
            any), indicated by the SA.  The SPI from the ESP header, and
            the (receiver) packet counter value (adjusted as required
            from the processing described in Section 3.4.3) are inputs
            to this algorithm, as they are required for the integrity
            check.

                 - If explicit cryptographic synchronization data, e.g.,
                   an IV, is indicated, it is taken from the Payload
                   field and input to the decryption algorithm as per
                   the algorithm specification.

                 - If implicit cryptographic synchronization data, e.g.,
                   an IV, is indicated, a local version of the IV is
                   constructed and input to the decryption algorithm as
                   per the algorithm specification.

         2. If the integrity check performed by the combined mode
            algorithm fails, the receiver MUST discard the received IP
            datagram as invalid; this is an auditable event.  The log
            data SHOULD include the SPI value, date/time received,
            Source Address, Destination Address, the Sequence Number,
            and (in IPv6) the cleartext Flow ID.

         3. Process any Padding as specified in the encryption algorithm
            specification, if the algorithm has not already done so.

         4. The receiver checks the Next Header field.  If the value is
            "59" (no next header), the (dummy) packet is discarded
            without further processing.

         5. Extract the original IP datagram (tunnel mode) or
            transport-layer frame (transport mode) from the ESP Payload
            Data field.  This implicitly discards any (optional) padding
            that has been added for traffic flow confidentiality.  (If
            present, the TFC padding will have been inserted after the
            IP payload and before the Padding field (see Section 2.4).)
```

Identifier: RQ_002_3085
RFC Clause: 4
Type: Mandatory
Applies to: IPsec host

Requirement:
If an IPsec host supports auditing then an ESP implementation within that host MUST also support auditing

RFC Text:
```
4.  Auditing

   Not all systems that implement ESP will implement auditing.  However,
   if ESP is incorporated into a system that supports auditing, then the
   ESP implementation MUST also support auditing and MUST allow a system
   administrator to enable or disable auditing for ESP.  For the most
   part, the granularity of auditing is a local matter.  However,
   several auditable events are identified in this specification and for
   each of these events a minimum set of information that SHOULD be
   included in an audit log is defined.

         - No valid Security Association exists for a session.  The
           audit log entry for this event SHOULD include the SPI value,
           date/time received, Source Address, Destination Address,
           Sequence Number, and (for IPv6) the cleartext Flow ID.

         - A packet offered to ESP for processing appears to be an IP
           fragment, i.e., the OFFSET field is non-zero or the MORE
           FRAGMENTS flag is set.  The audit log entry for this event
           SHOULD include the SPI value, date/time received, Source
           Address, Destination Address, Sequence Number, and (in IPv6)
           the Flow ID.

         - Attempt to transmit a packet that would result in Sequence
           Number overflow.  The audit log entry for this event SHOULD
           include the SPI value, current date/time, Source Address,
           Destination Address, Sequence Number, and (for IPv6) the
           cleartext Flow ID.

         - The received packet fails the anti-replay checks.  The audit
           log entry for this event SHOULD include the SPI value,
           date/time received, Source Address, Destination Address, the
           Sequence Number, and (in IPv6) the Flow ID.

         - The integrity check fails.  The audit log entry for this
           event SHOULD include the SPI value, date/time received,
           Source Address, Destination Address, the Sequence Number, and
           (for IPv6) the Flow ID.
```

Identifier: RQ_002_3086
RFC Clause: 4
Type: Mandatory
Applies to: IPsec host

Requirement:

An IPsec host that supports ESP auditing MUST allow a system administrator to enable or disable auditing for ESP.

RFC Text:

```
4.  Auditing

   Not all systems that implement ESP will implement auditing.  However,
   if ESP is incorporated into a system that supports auditing, then the
   ESP implementation MUST also support auditing and MUST allow a system
   administrator to enable or disable auditing for ESP.  For the most
   part, the granularity of auditing is a local matter.  However,
   several auditable events are identified in this specification and for
   each of these events a minimum set of information that SHOULD be
   included in an audit log is defined.

         - No valid Security Association exists for a session.  The
           audit log entry for this event SHOULD include the SPI value,
           date/time received, Source Address, Destination Address,
           Sequence Number, and (for IPv6) the cleartext Flow ID.

         - A packet offered to ESP for processing appears to be an IP
           fragment, i.e., the OFFSET field is non-zero or the MORE
           FRAGMENTS flag is set.  The audit log entry for this event
           SHOULD include the SPI value, date/time received, Source
           Address, Destination Address, Sequence Number, and (in IPv6)
           the Flow ID.

         - Attempt to transmit a packet that would result in Sequence
           Number overflow.  The audit log entry for this event SHOULD
           include the SPI value, current date/time, Source Address,
           Destination Address, Sequence Number, and (for IPv6) the
           cleartext Flow ID.

         - The received packet fails the anti-replay checks.  The audit
           log entry for this event SHOULD include the SPI value,
           date/time received, Source Address, Destination Address, the
           Sequence Number, and (in IPv6) the Flow ID.

         - The integrity check fails.  The audit log entry for this
           event SHOULD include the SPI value, date/time received,
           Source Address, Destination Address, the Sequence Number, and
           (for IPv6) the Flow ID.
```

Identifier: RQ_002_3087
RFC Clause: 4
Type: Recommended
Applies to: IPsec host

Requirement:

```
An IPsec host that supports ESP auditing SHOULD audit the following events:

         - No valid Security Association exists for a session.
         - A packet offered to ESP for processing appears to be an IP fragment,
         - Attempt to transmit a packet that would result in Sequence Number overflow.
         - The received packet fails the anti-replay checks.
         - The integrity check fails
```

RFC Text:

```
4.  Auditing

   Not all systems that implement ESP will implement auditing.  However,
   if ESP is incorporated into a system that supports auditing, then the
   ESP implementation MUST also support auditing and MUST allow a system
   administrator to enable or disable auditing for ESP.  For the most
   part, the granularity of auditing is a local matter.  However,
   several auditable events are identified in this specification and for
   each of these events a minimum set of information that SHOULD be
   included in an audit log is defined.

         - No valid Security Association exists for a session.  The
           audit log entry for this event SHOULD include the SPI value,
           date/time received, Source Address, Destination Address,
           Sequence Number, and (for IPv6) the cleartext Flow ID.

         - A packet offered to ESP for processing appears to be an IP
           fragment, i.e., the OFFSET field is non-zero or the MORE
           FRAGMENTS flag is set.  The audit log entry for this event
           SHOULD include the SPI value, date/time received, Source
           Address, Destination Address, Sequence Number, and (in IPv6)
           the Flow ID.

         - Attempt to transmit a packet that would result in Sequence
           Number overflow.  The audit log entry for this event SHOULD
           include the SPI value, current date/time, Source Address,
           Destination Address, Sequence Number, and (for IPv6) the
           cleartext Flow ID.

         - The received packet fails the anti-replay checks.  The audit
           log entry for this event SHOULD include the SPI value,
           date/time received, Source Address, Destination Address, the
           Sequence Number, and (in IPv6) the Flow ID.

         - The integrity check fails.  The audit log entry for this
           event SHOULD include the SPI value, date/time received,
           Source Address, Destination Address, the Sequence Number, and
           (for IPv6) the Flow ID.
```

Identifier: RQ_002_3088
RFC Clause: 2
Type: Mandatory
Applies to: IPsec host

Requirement:
An ESP packet MUST be formatted by concatenating the following named fields: Security Parameters Index (SPI) (4 Bytes); Sequence Number (4 Bytes); Payload Data (variable length); Padding (0-255 bytes); Pad Length (1 Byte); Next Header (1 Byte); optional Integrity Check Value (ICV) (Variable length).

```
  Octet                  Length (Octets)           Field
  ----------------------------------------------------------------------------------------------------
------
  1 to 4                  4                                  Security Parameters Index
  5 to 8                  4                                  Sequence Number
                             Varies                          Payload data
                             Varies                          Padding
                             1                                  Pad length
                             1                                  Next header
                             Varies                          Integrity check value
```

RFC Text:
The (outer) protocol header (IPv4, IPv6, or Extension) that immediately precedes the ESP header SHALL contain the value 50 in its Protocol (IPv4) or Next Header (IPv6, Extension) field (see IANA web page at http://www.iana.org/assignments/protocol-numbers). Figure 1 illustrates the top-level format of an ESP packet. **The packet begins with two 4-byte fields (Security Parameters Index (SPI) and Sequence Number). Following these fields is the Payload Data, which has substructure that depends on the choice of encryption algorithm and mode, and on the use of TFC padding, which is examined in more detail later. Following the Payload Data are Padding and Pad Length fields, and the Next Header field. The optional Integrity Check Value (ICV) field completes the packet.**

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ----
|               Security Parameters Index (SPI)                 | ^Int.
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
|                      Sequence Number                          | |ered
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | ----
|                    Payload Data* (variable)                   | |   ^
~                                                               ~ |   |
|                                                               | |Conf.
+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |Cov-
|               |     Padding (0-255 bytes)                     | |ered*
+-+-+-+-+-+-+-+-+               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |   |
|                               |  Pad Length   | Next Header   | v   v
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ------
|         Integrity Check Value-ICV   (variable)                |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

            Figure 1.  Top-Level Format of an ESP Packet
```

Identifier: RQ_002_3089
RFC Clause: 2
Type: Mandatory
Applies to: IPsec host

Requirement:
if there are concerns about backward compatibility they MUST be addressed by using a signaling mechanism between the two IPsec peers to ensure compatible versions of ESP (e.g., Internet Key Exchange (IKEv2) or an out-of-band configuration mechanism.

RFC Text:
ESP does not contain a version number, therefore if there are concerns about backward compatibility, they MUST be addressed by using a signaling mechanism between the two IPsec peers to ensure compatible versions of ESP (e.g., Internet Key Exchange (IKEv2) [Kau05]) or an out-of-band configuration mechanism.

Identifier: RQ_002_3091
RFC Clause: 2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
When an IPsec host receives an ESP packet, it MUST use the Security Parameter Index in the first four octets of the packet (and previously shared with the packet initiator using IKEv2 or another key exchange protocol) to identify the SA to which the incoming packet is bound

RFC Text:
The SPI is an arbitrary 32-bit value that is used by a receiver to identify the SA to which an incoming packet is bound. The SPI field is mandatory.

Identifier: RQ_002_3092
RFC Clause: 3.3
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP transport mode the sender MUST encapsulate the next layer protocol information between the ESP header and the ESP trailer fields, and retain the specified IP header (and any IP extension headers in the IPv6 context)

RFC Text:
In transport mode, the sender encapsulates the next layer protocol information between the ESP header and the ESP trailer fields, and retains the specified IP header (and any IP extension headers in the IPv6 context). In tunnel mode, the outer and inner IP header/extensions can be interrelated in a variety of ways. The construction of the outer IP header/extensions during the encapsulation process is described in the Security Architecture document.

Identifier: RQ_002_3093
RFC Clause: 2.1
Type: Optional
Applies to: IPsec host

Requirement:
For a unicast SA the SPI can be used by itself to specify an SA

RFC Text:
For a unicast SA, the SPI can be used by itself to specify an SA, or
it may be used in conjunction with the IPsec protocol type (in this
case ESP). Because the SPI value is generated by the receiver for a
unicast SA, whether the value is sufficient to identify an SA by
itself or whether it must be used in conjunction with the IPsec
protocol value is a local matter. This mechanism for mapping inbound
traffic to unicast SAs MUST be supported by all ESP implementations.

Identifier: RQ_002_3094
RFC Clause: 2.1
Type: Optional
Applies to: IPsec host

Requirement:
For a unicast SA, the SPI MAY be used in conjunction with the IPsec protocol type (in this case ESP)

RFC Text:
**For a unicast SA, the SPI can be used by itself to specify an SA, or
it may be used in conjunction with the IPsec protocol type (in this
case ESP).** Because the SPI value is generated by the receiver for a
unicast SA, whether the value is sufficient to identify an SA by
itself or whether it must be used in conjunction with the IPsec
protocol value is a local matter. This mechanism for mapping inbound
traffic to unicast SAs MUST be supported by all ESP implementations.

Identifier: RQ_002_3095
RFC Clause: 2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
The mechanism for mapping inbound traffic to unicast SAs using either 'SPI' or 'SPI+IPsec protocol type' MUST be supported by all ESP implementations

RFC Text:
For a unicast SA, the SPI can be used by itself to specify an SA, or
it may be used in conjunction with the IPsec protocol type (in this
case ESP). Because the SPI value is generated by the receiver for a
unicast SA, whether the value is sufficient to identify an SA by
itself or whether it must be used in conjunction with the IPsec
protocol value is a local matter. **This mechanism for mapping inbound
traffic to unicast SAs MUST be supported by all ESP implementations.**

Identifier: RQ_002_3099
RFC Clause: 2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
The indication of whether source and destination address matching is required to map inbound IPsec traffic to SAs MUST be set either as a side effect of manual SA configuration or via negotiation using an SA management protocol

RFC Text:
The indication of whether source and destination address matching is required to map inbound IPsec traffic to SAs MUST be set either as a side effect of manual SA configuration or via negotiation using an SA management protocol, e.g., IKE or Group Domain of Interpretation (GDOI) [RFC3547]. Typically, Source-Specific Multicast (SSM) [HC03] groups use a 3-tuple SA identifier composed of an SPI, a destination multicast address, and source address. An Any-Source Multicast group SA requires only an SPI and a destination multicast address as an identifier.

Identifier: RQ_002_3100
RFC Clause: 2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP a reserved SPI value (in the range 1 through 255) MUST NOT be used unless its use is specified in an RFC for a specific protocol

RFC Text:
The set of SPI values in the range 1 through 255 are reserved by the Internet Assigned Numbers Authority (IANA) for future use; a reserved SPI value will not normally be assigned by IANA unless the use of the assigned SPI value is specified in an RFC. The SPI value of zero (0) is reserved for local, implementation-specific use and MUST NOT be sent on the wire. (For example, a key management implementation might use the zero SPI value to mean "No Security Association Exists" during the period when the IPsec implementation has requested that its key management entity establish a new SA, but the SA has not yet been established.)

Identifier: RQ_002_3102
RFC Clause: 3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:

In IPsec ESP transport mode where separate confidentiality and integrity algorithms are employed the Sender MUST proceed for encryption as follows:

1. Encapsulate (into the ESP Payload field) the original next layer protocol information.
2. Add any necessary padding (both pptional TFC padding and (encryption) Padding)
3. Encrypt the result using the key, encryption algorithm, and algorithm mode specified for the SA and using any required cryptographic synchronization data.

RFC Text:

```
If separate confidentiality and integrity algorithms are employed,
   the Sender proceeds as follows:

         1. Encapsulate (into the ESP Payload field):
                 - for transport mode -- just the original next layer
                   protocol information.
                 - for tunnel mode -- the entire original IP datagram.

         2. Add any necessary padding -- Optional TFC padding and
            (encryption) Padding

         3. Encrypt the result using the key, encryption algorithm,
            and algorithm mode specified for the SA and using any
            required cryptographic synchronization data.
                 - If explicit cryptographic synchronization data,
                   e.g., an IV, is indicated, it is input to the
                   encryption algorithm per the algorithm specification
                   and placed in the Payload field.
                 - If implicit cryptographic synchronization data is
                   employed, it is constructed and input to the
                   encryption algorithm as per the algorithm
                   specification.
                 - If integrity is selected, encryption is performed
                   first, before the integrity algorithm is applied, and
                   the encryption does not encompass the ICV field.
                   This order of processing facilitates rapid detection
                   and rejection of replayed or bogus packets by the
                   receiver, prior to decrypting the packet, hence
                   potentially reducing the impact of denial of service
                   (DoS) attacks.  It also allows for the possibility of
                   parallel processing of packets at the receiver, i.e.,
                   decryption can take place in parallel with integrity
                   checking.  Note that because the ICV is not protected
                   by encryption, a keyed integrity algorithm must be
                   employed to compute the ICV.

         4. Compute the ICV over the ESP packet minus the ICV field.
            Thus, the ICV computation encompasses the SPI, Sequence
            Number, Payload Data, Padding (if present), Pad Length, and
            Next Header.  (Note that the last 4 fields will be in
            ciphertext form, because encryption is performed first.)  If
            the ESN option is enabled for the SA, the high-order 32
            bits of the sequence number are appended after the Next
            Header field for purposes of this computation, but are not
            transmitted.
```

Identifier: RQ_002_3103
RFC Clause: 3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:

In IPsec ESP tunnel mode where separate confidentiality and integrity algorithms are employed the Sender MUST proceed for encryption as follows:

1. Encapsulate (into the ESP Payload field) the original IP datagram.
2. Add any necessary padding (both pptional TFC padding and (encryption) Padding)
3. Encrypt the result using the key, encryption algorithm, and algorithm mode specified for the SA and using any required cryptographic synchronization data.

RFC Text:

```
If separate confidentiality and integrity algorithms are employed,
   the Sender proceeds as follows:

         1. Encapsulate (into the ESP Payload field):
                 - for transport mode -- just the original next layer
                   protocol information.
                 - for tunnel mode -- the entire original IP datagram.

         2. Add any necessary padding -- Optional TFC padding and
            (encryption) Padding

         3. Encrypt the result using the key, encryption algorithm,
            and algorithm mode specified for the SA and using any
            required cryptographic synchronization data.
                 - If explicit cryptographic synchronization data,
                   e.g., an IV, is indicated, it is input to the
                   encryption algorithm per the algorithm specification
                   and placed in the Payload field.
                 - If implicit cryptographic synchronization data is
                   employed, it is constructed and input to the
                   encryption algorithm as per the algorithm
                   specification.
                 - If integrity is selected, encryption is performed
                   first, before the integrity algorithm is applied, and
                   the encryption does not encompass the ICV field.
                   This order of processing facilitates rapid detection
                   and rejection of replayed or bogus packets by the
                   receiver, prior to decrypting the packet, hence
                   potentially reducing the impact of denial of service
                   (DoS) attacks.  It also allows for the possibility of
                   parallel processing of packets at the receiver, i.e.,
                   decryption can take place in parallel with integrity
                   checking.  Note that because the ICV is not protected
                   by encryption, a keyed integrity algorithm must be
                   employed to compute the ICV.

         4. Compute the ICV over the ESP packet minus the ICV field.
            Thus, the ICV computation encompasses the SPI, Sequence
            Number, Payload Data, Padding (if present), Pad Length, and
            Next Header.  (Note that the last 4 fields will be in
            ciphertext form, because encryption is performed first.)  If
            the ESN option is enabled for the SA, the high-order 32
            bits of the sequence number are appended after the Next
            Header field for purposes of this computation, but are not
            transmitted.
```

Identifier: RQ_002_3104
RFC Clause: 3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:

In IPsec ESP where separate confidentiality and integrity algorithms are employed the Sender MUST proceed for integrity as follows: Compute the ICV over the SPI, Sequence Number, Payload Data, Padding (if present), Pad Length, and Next Header.

RFC Text:

```
If separate confidentiality and integrity algorithms are employed,
   the Sender proceeds as follows:

         1. Encapsulate (into the ESP Payload field):
                 - for transport mode -- just the original next layer
                   protocol information.
                 - for tunnel mode -- the entire original IP datagram.

         2. Add any necessary padding -- Optional TFC padding and
            (encryption) Padding

         3. Encrypt the result using the key, encryption algorithm,
            and algorithm mode specified for the SA and using any
            required cryptographic synchronization data.
                 - If explicit cryptographic synchronization data,
                   e.g., an IV, is indicated, it is input to the
                   encryption algorithm per the algorithm specification
                   and placed in the Payload field.
                 - If implicit cryptographic synchronization data is
                   employed, it is constructed and input to the
                   encryption algorithm as per the algorithm
                   specification.
                 - If integrity is selected, encryption is performed
                   first, before the integrity algorithm is applied, and
                   the encryption does not encompass the ICV field.
                   This order of processing facilitates rapid detection
                   and rejection of replayed or bogus packets by the
                   receiver, prior to decrypting the packet, hence
                   potentially reducing the impact of denial of service
                   (DoS) attacks.  It also allows for the possibility of
                   parallel processing of packets at the receiver, i.e.,
                   decryption can take place in parallel with integrity
                   checking.  Note that because the ICV is not protected
                   by encryption, a keyed integrity algorithm must be
                   employed to compute the ICV.

         4. Compute the ICV over the ESP packet minus the ICV field.
            Thus, the ICV computation encompasses the SPI, Sequence
            Number, Payload Data, Padding (if present), Pad Length, and
            Next Header.  (Note that the last 4 fields will be in
            ciphertext form, because encryption is performed first.)  If
            the ESN option is enabled for the SA, the high-order 32
            bits of the sequence number are appended after the Next
            Header field for purposes of this computation, but are not
            transmitted.
```

Identifier: RQ_002_3105
RFC Clause: 3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:

In IPsec ESP where separate confidentiality and integrity algorithms are employed with the ESN option enabled the high-order 32 bits of the sequence number MUST be appended after the Next Header field for purposes of ICV computation

RFC Text:

```
If separate confidentiality and integrity algorithms are employed,
   the Sender proceeds as follows:

         1. Encapsulate (into the ESP Payload field):
                - for transport mode -- just the original next layer
                  protocol information.
                - for tunnel mode -- the entire original IP datagram.

         2. Add any necessary padding -- Optional TFC padding and
            (encryption) Padding

         3. Encrypt the result using the key, encryption algorithm,
            and algorithm mode specified for the SA and using any
            required cryptographic synchronization data.
                - If explicit cryptographic synchronization data,
                  e.g., an IV, is indicated, it is input to the
                  encryption algorithm per the algorithm specification
                  and placed in the Payload field.
                - If implicit cryptographic synchronization data is
                  employed, it is constructed and input to the
                  encryption algorithm as per the algorithm
                  specification.
                - If integrity is selected, encryption is performed
                  first, before the integrity algorithm is applied, and
                  the encryption does not encompass the ICV field.
                  This order of processing facilitates rapid detection
                  and rejection of replayed or bogus packets by the
                  receiver, prior to decrypting the packet, hence
                  potentially reducing the impact of denial of service
                  (DoS) attacks.  It also allows for the possibility of
                  parallel processing of packets at the receiver, i.e.,
                  decryption can take place in parallel with integrity
                  checking.  Note that because the ICV is not protected
                  by encryption, a keyed integrity algorithm must be
                  employed to compute the ICV.

         4. Compute the ICV over the ESP packet minus the ICV field.
            Thus, the ICV computation encompasses the SPI, Sequence
            Number, Payload Data, Padding (if present), Pad Length, and
            Next Header.  (Note that the last 4 fields will be in
            ciphertext form, because encryption is performed first.)  If
            the ESN option is enabled for the SA, the high-order 32
            bits of the sequence number are appended after the Next
            Header field for purposes of this computation, but are not
            transmitted.
```

Identifier: RQ_002_3106
RFC Clause: 3.3.2.1
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP where separate confidentiality and integrity algorithms are employed with the ESN option enabled the high-order 32 bits of the sequence number MUST NOT be transmitted

RFC Text:
```
If separate confidentiality and integrity algorithms are employed,
   the Sender proceeds as follows:

         1. Encapsulate (into the ESP Payload field):
                 - for transport mode -- just the original next layer
                   protocol information.
                 - for tunnel mode -- the entire original IP datagram.

         2. Add any necessary padding -- Optional TFC padding and
            (encryption) Padding

         3. Encrypt the result using the key, encryption algorithm,
            and algorithm mode specified for the SA and using any
            required cryptographic synchronization data.
                 - If explicit cryptographic synchronization data,
                   e.g., an IV, is indicated, it is input to the
                   encryption algorithm per the algorithm specification
                   and placed in the Payload field.
                 - If implicit cryptographic synchronization data is
                   employed, it is constructed and input to the
                   encryption algorithm as per the algorithm
                   specification.
                 - If integrity is selected, encryption is performed
                   first, before the integrity algorithm is applied, and
                   the encryption does not encompass the ICV field.
                   This order of processing facilitates rapid detection
                   and rejection of replayed or bogus packets by the
                   receiver, prior to decrypting the packet, hence
                   potentially reducing the impact of denial of service
                   (DoS) attacks.  It also allows for the possibility of
                   parallel processing of packets at the receiver, i.e.,
                   decryption can take place in parallel with integrity
                   checking.  Note that because the ICV is not protected
                   by encryption, a keyed integrity algorithm must be
                   employed to compute the ICV.

         4. Compute the ICV over the ESP packet minus the ICV field.
            Thus, the ICV computation encompasses the SPI, Sequence
            Number, Payload Data, Padding (if present), Pad Length, and
            Next Header.  (Note that the last 4 fields will be in
            ciphertext form, because encryption is performed first.)  If
            the ESN option is enabled for the SA, the high-order 32
            bits of the sequence number are appended after the Next
            Header field for purposes of this computation, but are not
            transmitted.
```

Identifier: RQ_002_3107
RFC Clause: 3.3.2.2
Type: Mandatory
Applies to: IPsec host

Requirement:

In IPsec ESP transport mode where combined confidentiality and integrity algorithms are employed the Sender MUST proceed for encryption as follows:

1. Encapsulate (into the ESP Payload field) the original next layer protocol information.
2. Add any necessary padding (both pptional TFC padding and (encryption) Padding)
3. Encrypt the result using the key, encryption algorithm, and algorithm mode specified for the SA and using any required cryptographic synchronization data.

RFC Text:

```
If a combined confidentiality/integrity algorithm is employed, the
   Sender proceeds as follows:

         1. Encapsulate into the ESP Payload Data field:
                 - for transport mode -- just the original next layer
                   protocol information.
                 - for tunnel mode -- the entire original IP datagram.

         2. Add any necessary padding -- includes optional TFC padding
            and (encryption) Padding.

         3. Encrypt and integrity protect the result using the key
            and combined mode algorithm specified for the SA and using
            any required cryptographic synchronization data.
                 - If explicit cryptographic synchronization data,
                   e.g., an IV, is indicated, it is input to the
                   combined mode algorithm per the algorithm
                   specification and placed in the Payload field.
                 - If implicit cryptographic synchronization data is
                   employed, it is constructed and input to the
                   encryption algorithm as per the algorithm
                   specification.
                 - The Sequence Number (or Extended Sequence Number, as
                   appropriate) and the SPI are inputs to the
                   algorithm, as they must be included in the integrity
                   check computation.  The means by which these values
                   are included in this computation are a function of
                   the combined mode algorithm employed and thus not
                   specified in this standard.
                 - The (explicit) ICV field MAY be a part of the ESP
                   packet format when a combined mode algorithm is
                   employed.  If one is not used, an analogous field
                   usually will be a part of the ciphertext payload.
                   The location of any integrity fields, and the means
                   by which the Sequence Number and SPI are included in
                   the integrity computation, MUST be defined in an RFC
                   that defines the use of the combined mode algorithm
                   with ESP.
```

Identifier: RQ_002_3108
RFC Clause: 3.3.2.2
Type: Mandatory
Applies to: IPsec host

Requirement:

In IPsec ESP tunnel mode where combined confidentiality and integrity algorithms are employed the Sender MUST proceed for encryption as follows:

1. Encapsulate (into the ESP Payload field) the original IP datagram.
2. Add any necessary padding (both pptional TFC padding and (encryption) Padding)
3. Encrypt the result using the key, encryption algorithm, and algorithm mode specified for the SA and using any required cryptographic synchronization data.

RFC Text:

If a combined confidentiality/integrity algorithm is employed, **the Sender proceeds as follows:**

1. **Encapsulate into the ESP Payload Data field:**
   - **for transport mode -- just the original next layer protocol information.**
   - **for tunnel mode -- the entire original IP datagram.**

2. **Add any necessary padding -- includes optional TFC padding and (encryption) Padding.**

3. **Encrypt and integrity protect the result using the key and combined mode algorithm specified for the SA and using any required cryptographic synchronization data.**
   - If explicit cryptographic synchronization data, e.g., an IV, is indicated, it is input to the combined mode algorithm per the algorithm specification and placed in the Payload field.
   - If implicit cryptographic synchronization data is employed, it is constructed and input to the encryption algorithm as per the algorithm specification.
   - The Sequence Number (or Extended Sequence Number, as appropriate) and the SPI are inputs to the algorithm, as they must be included in the integrity check computation. The means by which these values are included in this computation are a function of the combined mode algorithm employed and thus not specified in this standard.
   - The (explicit) ICV field MAY be a part of the ESP packet format when a combined mode algorithm is employed. If one is not used, an analogous field usually will be a part of the ciphertext payload. The location of any integrity fields, and the means by which the Sequence Number and SPI are included in the integrity computation, MUST be defined in an RFC that defines the use of the combined mode algorithm with ESP.

Identifier: RQ_002_3109
RFC Clause: 3.3.2.2
Type: Optional
Applies to: IPsec host

Requirement:

In IPsec ESP where combined confidentiality and integrity algorithms are employed the explicit ICV field MAY be a part of the ESP packet format.

RFC Text:

```
If a combined confidentiality/integrity algorithm is employed, the
   Sender proceeds as follows:

         1. Encapsulate into the ESP Payload Data field:
                 - for transport mode -- just the original next layer
                   protocol information.
                 - for tunnel mode -- the entire original IP datagram.

         2. Add any necessary padding -- includes optional TFC padding
            and (encryption) Padding.

         3. Encrypt and integrity protect the result using the key
            and combined mode algorithm specified for the SA and using
            any required cryptographic synchronization data.
                 - If explicit cryptographic synchronization data,
                   e.g., an IV, is indicated, it is input to the
                   combined mode algorithm per the algorithm
                   specification and placed in the Payload field.
                 - If implicit cryptographic synchronization data is
                   employed, it is constructed and input to the
                   encryption algorithm as per the algorithm
                   specification.
                 - The Sequence Number (or Extended Sequence Number, as
                   appropriate) and the SPI are inputs to the
                   algorithm, as they must be included in the integrity
                   check computation.  The means by which these values
                   are included in this computation are a function of
                   the combined mode algorithm employed and thus not
                   specified in this standard.
                 - The (explicit) ICV field MAY be a part of the ESP
                   packet format when a combined mode algorithm is
                   employed.  If one is not used, an analogous field
                   usually will be a part of the ciphertext payload.
                   The location of any integrity fields, and the means
                   by which the Sequence Number and SPI are included in
                   the integrity computation, MUST be defined in an RFC
                   that defines the use of the combined mode algorithm
                   with ESP.
```

Identifier: RQ_002_3110
RFC Clause: 3.3.2.2
Type: Optional
Applies to: IPsec host

Requirement:

In IPsec ESP where combined confidentiality and integrity algorithms are employed and where the explicit ICV field is not provided the ICV MAY be a part of the ESP payload field.

RFC Text:

```
If a combined confidentiality/integrity algorithm is employed, the
   Sender proceeds as follows:

         1. Encapsulate into the ESP Payload Data field:
                - for transport mode -- just the original next layer
                  protocol information.
                - for tunnel mode -- the entire original IP datagram.

         2. Add any necessary padding -- includes optional TFC padding
            and (encryption) Padding.

         3. Encrypt and integrity protect the result using the key
            and combined mode algorithm specified for the SA and using
            any required cryptographic synchronization data.
                - If explicit cryptographic synchronization data,
                  e.g., an IV, is indicated, it is input to the
                  combined mode algorithm per the algorithm
                  specification and placed in the Payload field.
                - If implicit cryptographic synchronization data is
                  employed, it is constructed and input to the
                  encryption algorithm as per the algorithm
                  specification.
                - The Sequence Number (or Extended Sequence Number, as
                  appropriate) and the SPI are inputs to the
                  algorithm, as they must be included in the integrity
                  check computation.  The means by which these values
                  are included in this computation are a function of
                  the combined mode algorithm employed and thus not
                  specified in this standard.
                - The (explicit) ICV field MAY be a part of the ESP
                  packet format when a combined mode algorithm is
                  employed.  If one is not used, an analogous field
                  usually will be a part of the ciphertext payload.
                  The location of any integrity fields, and the means
                  by which the Sequence Number and SPI are included in
                  the integrity computation, MUST be defined in an RFC
                  that defines the use of the combined mode algorithm
                  with ESP.
```

Identifier: RQ_002_3111
RFC Clause: 3.3.2.2
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP where combined confidentiality and integrity algorithms are employed and where either RQ_SEC_3109 or RQ_SEC_3110 apply the location of any integrity fields, and the means by which the Sequence Number and SPI are included in the integrity computation, MUST be defined in an RFC that defines the use of the combined mode algorithm with ESP .

RFC Text:
```
If a combined confidentiality/integrity algorithm is employed, the
   Sender proceeds as follows:

         1. Encapsulate into the ESP Payload Data field:
                 - for transport mode -- just the original next layer
                   protocol information.
                 - for tunnel mode -- the entire original IP datagram.

         2. Add any necessary padding -- includes optional TFC padding
            and (encryption) Padding.

         3. Encrypt and integrity protect the result using the key
            and combined mode algorithm specified for the SA and using
            any required cryptographic synchronization data.
                 - If explicit cryptographic synchronization data,
                   e.g., an IV, is indicated, it is input to the
                   combined mode algorithm per the algorithm
                   specification and placed in the Payload field.
                 - If implicit cryptographic synchronization data is
                   employed, it is constructed and input to the
                   encryption algorithm as per the algorithm
                   specification.
                 - The Sequence Number (or Extended Sequence Number, as
                   appropriate) and the SPI are inputs to the
                   algorithm, as they must be included in the integrity
                   check computation.  The means by which these values
                   are included in this computation are a function of
                   the combined mode algorithm employed and thus not
                   specified in this standard.
                 - The (explicit) ICV field MAY be a part of the ESP
                   packet format when a combined mode algorithm is
                   employed.  If one is not used, an analogous field
                   usually will be a part of the ciphertext payload.
                   The location of any integrity fields, and the means
                   by which the Sequence Number and SPI are included in
                   the integrity computation, MUST be defined in an RFC
                   that defines the use of the combined mode algorithm
                   with ESP.
```

Identifier: RQ_002_3112
RFC Clause: 3.3.3
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP the sender MUST increment the sequence number (or ESN) counter for each packet in each SA and insert the low-order 32 bits of the value into the Sequence Number field

RFC Text:
```
   The sender's counter is initialized to 0 when an SA is established.
   The sender increments the sequence number (or ESN) counter for this
   SA and inserts the low-order 32 bits of the value into the Sequence
   Number field.  Thus, the first packet sent using a given SA will
   contain a sequence number of 1.
```

Identifier: RQ_002_3113
RFC Clause: 3.3.3
Type: Mandatory
Applies to: IPsec host

Requirement:
In IPsec ESP the first packet sent using a given SA MUST contain a sequence number of 1

RFC Text:

```
The sender's counter is initialized to 0 when an SA is established.
The sender increments the sequence number (or ESN) counter for this
SA and inserts the low-order 32 bits of the value into the Sequence
Number field.  Thus, the first packet sent using a given SA will
contain a sequence number of 1.
```

4.4 Requirements extracted from RFC 4305

Identifier: RQ_002_5000
RFC Clause: 3
Type: Mandatory
Applies to: IPsec host

Requirement:
For IPsec implementations to interoperate they MUST have at least one security algorithms in common

RFC Text:

```
For IPsec implementations to interoperate, they must support one or
   more security algorithms in common.  This section specifies the
   security algorithm implementation requirements for standards-
   conformant ESP and AH implementations.  The security algorithms
   actually used for any particular ESP or AH security association are
   determined by a negotiation mechanism, such as the Internet Key
   Exchange (IKE [RFC2409, IKEv2]) or pre-establishment.
```

Identifier: RQ_002_5001
RFC Clause: 3
Type: Mandatory
Applies to: IPsec host

Requirement:
The security algorithms used for ESP or AH security association MUST be determined by negotiation (examples of negotiation mechanisms inlcude the Internet Key Exchange)

RFC Text:

```
For IPsec implementations to interoperate, they must support one or
   more security algorithms in common.  This section specifies the
   security algorithm implementation requirements for standards-
   conformant ESP and AH implementations.  The security algorithms
   actually used for any particular ESP or AH security association are
   determined by a negotiation mechanism, such as the Internet Key
   Exchange (IKE [RFC2409, IKEv2]) or pre-establishment.
```

Identifier: RQ_002_5002
RFC Clause: 3.1.1
Type: Mandatory
Applies to: IPsec host

Requirement:

```
An IPsec host supporting an ESP Security Association MUST support the NULL encryption algorithm
```

RFC Text:

```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.

--------------------
```

Identifier: RQ_002_5003
RFC Clause: 3.1.1.
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host supporting an ESP Security Association MUST support the TripleDES-CBC encryption algorithm

RFC Text:
```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.

--------------------
```

Identifier: RQ_002_5004
RFC Clause: 3.1.1
Type: Recommended
Applies to: IPsec host

Requirement:

An IPsec host supporting an ESP Security Association SHOULD support the AES-CBC encryption algorithm with 128-bit key length

RFC Text:

```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.

--------------------
```

Identifier: RQ_002_5005
RFC Clause: 3.1.1
Type: Recommended
Applies to: IPsec host

Requirement:
An IPsec host supporting an ESP Security Association SHOULD support the AES-CTR encryption algorithm

RFC Text:
```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.

--------------------
```

Identifier: RQ_002_5006
RFC Clause: 3.1.1
Type: Recommended
Applies to: IPsec host

Requirement:
An IPsec host supporting an ESP Security Association SHOULD NOT support the DES-CBC encryption algorithm

RFC Text:
```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.
```

Identifier: RQ_002_5007
RFC Clause: 3.1.1
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host supporting an ESP Security Association MUST support the HMAC-SHA1-96 authentication algorithm

RFC Text:

```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.

--------------------
```

Identifier: RQ_002_5008
RFC Clause: 3.1.1
Type: Mandatory
Applies to: IPsec host

Requirement:

An IPsec host supporting an ESP Security Association MUST support the NULL authentication algorithm

RFC Text:

```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.

--------------------
```

Identifier: RQ_002_5009
RFC Clause: 3.1.1
Type: Recommended
Applies to: IPsec host

Requirement:

```
An IPsec host supporting an ESP Security Association SHOULD support the AES-XCBC-MAC-96
authentication algorithm
```

RFC Text:

```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.

--------------------
```

Identifier: RQ_002_5010
RFC Clause: 3.1.1
Type: Optional
Applies to: IPsec host

Requirement:

An IPsec host supporting an ESP Security Association MAY support the HMAC-MD5-96 authentication algorithm

RFC Text:

```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403]  (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.
```

Identifier: RQ_002_5011
RFC Clause: 3.1.1
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host supporting an ESP Security Association MUST NOT deploy both the NULL authentication algorithm and the NULL encryption algorithm

RFC Text:
```
These tables list encryption and authentication algorithms for the
   IPsec Encapsulating Security Payload protocol.

      Requirement    Encryption Algorithm (notes)
      -----------    --------------------
      MUST           NULL (1)
      MUST-          TripleDES-CBC [RFC2451]
      SHOULD+        AES-CBC with 128-bit keys [RFC3602]
      SHOULD         AES-CTR [RFC3686]
      SHOULD NOT     DES-CBC [RFC2405] (3)

      Requirement    Authentication Algorithm (notes)
      -----------    ------------------------
      MUST           HMAC-SHA1-96 [RFC2404]
      MUST           NULL (1)
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403]  (2)

   Notes:

   (1) Since ESP encryption and authentication are optional, support for
       the two "NULL" algorithms is required to maintain consistency
       with the way these services are negotiated.  Note that while
       authentication and encryption can each be "NULL", they MUST NOT
       both be "NULL".
   (2) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
   (3) DES, with its small key size and publicly demonstrated and open-
       design special-purpose cracking hardware, is of questionable
       security for general use.
```

Identifier: RQ_002_5012
RFC Clause: 3.2
Type: Mandatory
Applies to: IPsec host

Requirement:
An IPsec host supporting an AH Security Association MUST support the HMAC-SHA1-96 authentication algorithm

RFC Text:
```
The implementation conformance requirements for security algorithms
   for AH are given below.  See Section 2 for definitions of the values
   in the "Requirement" column.  As you would suspect, all of these
   algorithms are authentication algorithms.

      Requirement    Algorithm (notes)
      -----------    ---------
      MUST           HMAC-SHA1-96 [RFC2404]
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (1)

   Note:

   (1) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
```

Identifier: RQ_002_5013
RFC Clause: 3.2
Type: Recommended
Applies to: IPsec host

Requirement:
An IPsec host supporting an AH Security Association SHOULD support the AES-XCBC-MAC-96 authentication algorithm

RFC Text:
```
The implementation conformance requirements for security algorithms
   for AH are given below.  See Section 2 for definitions of the values
   in the "Requirement" column.  As you would suspect, all of these
   algorithms are authentication algorithms.

      Requirement    Algorithm (notes)
      -----------    ---------
      MUST           HMAC-SHA1-96 [RFC2404]
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (1)

   Note:

   (1) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
```

Identifier: RQ_002_5014
RFC Clause: 3.2
Type: Optional
Applies to: IPsec host

Requirement:
An IPsec host supporting an AH Security Association MAY support the HMAC-MAC-96 authentication algorithm

RFC Text:
```
The implementation conformance requirements for security algorithms
   for AH are given below.  See Section 2 for definitions of the values
   in the "Requirement" column.  As you would suspect, all of these
   algorithms are authentication algorithms.

      Requirement    Algorithm (notes)
      -----------    ---------
      MUST           HMAC-SHA1-96 [RFC2404]
      SHOULD+        AES-XCBC-MAC-96 [RFC3566]
      MAY            HMAC-MD5-96 [RFC2403] (1)

   Note:

   (1) Weaknesses have become apparent in MD5; however, these should not
       affect the use of MD5 with HMAC.
```

4.5 Requirements extracted from RFC 4306

Identifier: RQ_002_6000
RFC Clause: 1
Type: Mandatory
Applies to: Host

Requirement:
When establishing an IKE Security Association, an IKE implementation MUST complete all IKE_SA_INIT exchanges before initiating any other exchange type on that SA.

RFC Text:
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. 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. 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.

Identifier: RQ_002_6001
RFC Clause: 1
Type: Mandatory
Applies to: Host

Requirement:
When establishing an IKE Security Association, an IKE implementation MUST complete all IKE_AUTH exchanges before initiating any CREATE_CHILD_SA and INFORMATIONAL exchanges on that SA

RFC Text:
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. 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. 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.

Identifier: RQ_002_6002
RFC Clause: 1
Type: Optional
Applies to: Host

Requirement:
When establishing an IKE Security Association, an IKE implementation may initiate any number of CREATE_CHILD_SA and INFORMATIONAL exchanges in any order.

RFC Text:
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. 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**. 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.

Identifier: RQ_002_6003
RFC Clause: 1.4.
Type: Mandatory
Applies to: Host

Requirement:
IKE INFORMATIONAL exchanges MUST ONLY occur after the initial exchanges to establish the relevant IKE Security Associations

RFC Text:
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. **INFORMATIONAL exchanges MUST ONLY occur after the initial exchanges** and are cryptographically protected with the negotiated keys.

Identifier: RQ_002_6004
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
An IKE endpoint in an established Security Association MUST cryptographically protect all IKE INFORMATIONAL exchanges sent across that SA using the keys negotiated during the establishment of the SA

RFC Text:
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. **INFORMATIONAL exchanges MUST ONLY occur after the initial exchanges and are cryptographically protected with the negotiated keys**.

Identifier: RQ_002_6005
RFC Clause: 1.4.
Type: Mandatory
Applies to: Host

Requirement:
An endpoint in an established IKE Security Association MUST use that SA to send any Informational exchanges pertaining to the control of the SA.

RFC Text:
Control messages that pertain to an IKE_SA MUST be sent under that IKE_SA. 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)

Identifier: RQ_002_6006
RFC Clause: 1.4.
Type: Mandatory
Applies to: Host

Requirement:
An endpoint in an established IKE Security Association MUST send any Informational exchanges that pertain to the control of an associated CHILD_SA under the protection of either the IKE_SA which generated them or its successor if the IKE_SA was replaced for the purpose of rekeying

RFC Text:
Control messages that pertain to an IKE_SA MUST be sent under that IKE_SA. **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)**.

Identifier: RQ_002_6007
RFC Clause: 1.4.
Type: Mandatory
Applies to: Host

Requirement:
To avoid retransmission of Notification, Delete and Configuration messages, the recipient of an IKE INFORMATIONAL exchange request MUST send a valid response

RFC Text:
Messages in an INFORMATIONAL exchange contain zero or more Notification, Delete, and Configuration payloads. 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). 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

----

Identifier: RQ_002_6008
RFC Clause: 1.4.
Type: Optional
Applies to: Host

Requirement:
The response to an IKE INFORMATIONAL exchange request MAY be a message with no payloads

RFC Text:
Messages in an INFORMATIONAL exchange contain zero or more Notification, Delete, and Configuration payloads. 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). **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.

Identifier: RQ_002_6009
RFC Clause: 1.4.
Type: Optional
Applies to: Host

Requirement:
As a means of verifying that the other endpoint in an IKE Security Association is alive, the request message in an INFORMATIONAL exchange MAY contain no payloads.

RFC Text:
Messages in an INFORMATIONAL exchange contain zero or more Notification, Delete, and Configuration payloads. 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). 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**.

Identifier: RQ_002_6010
RFC Clause: 1.4.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE Security Association endpoint receives an INFORMATION exchange request with one or more Delete payloads, it MUST close the designated Security Associations

RFC Text:

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. 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. Each endpoint MUST close its incoming SAs and allow the other endpoint to close the other SA in each pair. **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**. Normally, the reply in the INFORMATIONAL exchange will contain delete payloads for the paired SAs going in the other direction. There is one exception. If by chance both ends of a set of SAs independently decide to close them, each may send a delete payload and the two requests may cross in the network. If a node receives a delete request for SAs for which it has already issued a delete request, it MUST delete the outgoing SAs while processing the request and the incoming SAs while processing the response. In that case, the responses MUST NOT include delete payloads for the deleted SAs, since that would result in duplicate deletion and could in theory delete the wrong SA.

Identifier: RQ_002_6011
RFC Clause: 1.4
Type: Mandatory
Applies to: Host

Requirement:

When a request is received to delete a Security Association which has other nested Security Associations, the receiving endpoint MUST delete all of these related Security Associations together

RFC Text:

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. **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**. Each endpoint MUST close its incoming SAs and allow the other endpoint to close the other SA in each pair. 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. Normally, the reply in the INFORMATIONAL exchange will contain delete payloads for the paired SAs going in the other direction. There is one exception. If by chance both ends of a set of SAs independently decide to close them, each may send a delete payload and the two requests may cross in the network. If a node receives a delete request for SAs for which it has already issued a delete request, it MUST delete the outgoing SAs while processing the request and the incoming SAs while processing the response. In that case, the responses MUST NOT include delete payloads for the deleted SAs, since that would result in duplicate deletion and could in theory delete the wrong SA.

Identifier: RQ_002_6012
RFC Clause: 1.4.
Type: Optional
Applies to: Host

Requirement:

An endpoint in an established IKE Security Association MAY send an IKE INFORMATIONAL exchange request or response containing no payloads

RFC Text:

Messages in an INFORMATIONAL exchange contain zero or more Notification, Delete, and Configuration **payloads**. 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). 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.

Identifier: RQ_002_6013
RFC Clause: 1.4.
Type: Optional
Applies to: Host

Requirement:
An endpoint in an established IKE Security Association MAY send an IKE INFORMATIONAL exchange request or response containing zero or more Notification payloads

RFC Text:
Messages in an INFORMATIONAL exchange contain zero or more Notification, Delete, and Configuration **payloads**. 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). 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.

Identifier: RQ_002_6014
RFC Clause: 1.4.
Type: Mandatory
Applies to: Host

Requirement:
An endpoint in an established IKE Security Association MAY send an IKE INFORMATIONAL exchange request or response containing zero or more Delete payloads

RFC Text:
Messages in an INFORMATIONAL exchange contain zero or more Notification, **Delete**, and Configuration **payloads**. 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). 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.

Identifier: RQ_002_6015
RFC Clause: 1.4.
Type: Optional
Applies to: Host

Requirement:
An endpoint in an established IKE Security Association MAY send an IKE INFORMATIONAL exchange request or response containing zero or more Configuration payloads

RFC Text:
Messages in an INFORMATIONAL exchange contain zero or more Notification, Delete, and **Configuration payloads**. 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). 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.

Identifier: RQ_002_6016
RFC Clause: 1.4
Type: Mandatory
Applies to: Host

Requirement:
When an IKE Security Association endpoint receives an INFORMATIONAL exchange request with one or more delete payloads, it MUST send an INFORMATION exchange response containing Delete payloads for the corresponding Security Associations in the reverse direction but excluding any that have already been sent in a coincidental simultaneous INFORMATION exchange request to the requesting endpoint.

RFC Text:
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. 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. Each endpoint MUST close its incoming SAs and allow the other endpoint to close the other SA in each pair. 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. **Normally, the reply in the INFORMATIONAL exchange will contain delete payloads for the paired SAs going in the other direction. There is one exception. If by chance both ends of a set of SAs independently decide to close them, each may send a delete payload and the two requests may cross in the network. If a node receives a delete request for SAs for which it has already issued a delete request, it MUST delete the outgoing SAs while processing the request and the incoming SAs while processing the response. In that case, the responses MUST NOT include delete payloads for the deleted SAs**, since that would result in duplicate deletion and could in theory delete the wrong SA.

Identifier: RQ_002_6017
RFC Clause: 1.4.
Type: Recommended
Applies to: Host

Requirement:
In the event that one ESP or AH Security Association in a pair is closed but the other one is not, an IKE implementation SHOULD record this fact in a log after it has persisted for a predefined period..

RFC Text:
A node SHOULD regard half-closed connections as anomalous and audit their existence should they persist. Note that this specification nowhere specifies time periods, so it is up to individual endpoints to decide how long to wait}}. A node MAY refuse to accept incoming data on half-closed connections but MUST NOT unilaterally close them and reuse the SPIs. If connection state becomes sufficiently messed up, a node MAY close the IKE_SA; doing so will implicitly close all SAs negotiated under it. It can then rebuild the SAs it needs on a clean base under a new IKE_SA.

Identifier: RQ_002_6018
RFC Clause: 1.4.
Type: Optional
Applies to: Host

Requirement:
In the event that one ESP or AH Security Association in a pair is closed but the other one is not, an IKE implementation MAY refuse to accept incoming data on half-closed Security Associations.

RFC Text:
A node SHOULD regard half-closed connections as anomalous and audit their existence should they persist. Note that this specification nowhere specifies time periods, so it is up to individual endpoints to decide how long to wait. **A node MAY refuse to accept incoming data on half-closed connections** but MUST NOT unilaterally close them and reuse the SPIs. If connection state becomes sufficiently messed up, a node MAY close the IKE_SA; doing so will implicitly close all SAs negotiated under it. It can then rebuild the SAs it needs on a clean base under a new IKE_SA.

Identifier: RQ_002_6019
RFC Clause: 1.4.
Type: Mandatory
Applies to: Host

Requirement:
In the event that one ESP or AH Security Association in a pair is closed but the other one is not, an IKE implementation MUST NOT unilaterally close the half-closed Security Associations and reuse the SPIs

RFC Text:

A node SHOULD regard half-closed connections as anomalous and audit their existence should they persist. Note that this specification nowhere specifies time periods, so it is up to individual endpoints to decide how long to wait. A node MAY refuse to accept incoming data on half-closed connections **but MUST NOT unilaterally close them** and reuse the SPIs. If connection state becomes sufficiently messed up, a node MAY close the IKE_SA; doing so will implicitly close all SAs negotiated under it. It can then rebuild the SAs it needs on a clean base under a new IKE_SA.

Identifier: RQ_002_6020
RFC Clause: 1.4.
Type: Optional
Applies to: Host

Requirement:

In the event that one ESP or AH Security Association in a pair is closed but the other one is not, an IKE implementation MAY close the IKE Security Association

RFC Text:

A node SHOULD regard half-closed connections as anomalous and audit their existence should they persist. Note that this specification nowhere specifies time periods, so it is up to individual endpoints to decide how long to wait. A node MAY refuse to accept incoming data on half-closed connections but MUST NOT unilaterally close them and reuse the SPIs. **If connection state becomes sufficiently messed up, a node MAY close the IKE_SA**; doing so will implicitly close all SAs negotiated under it. It can then rebuild the SAs it needs on a clean base under a new IKE_SA.

Identifier: RQ_002_6021
RFC Clause: 1.5.
Type: Optional
Applies to: Host

Requirement:

If an endpoint in an established IKE Security Association receives an encrypted IKE packet on port 500 or 4500 with the source IP address of the other endpoint in the SA but with an unrecognized SPI, it MAY send a notification of the wayward packet over that IKE_SA in an INFORMATIONAL exchange containing a Notify payload set to INVALID_IKE_SPI

RFC Text:

If an encrypted IKE packet arrives on port 500 or 4500 with an unrecognized SPI, it could be because the receiving node has recently crashed and lost state or because of some other system malfunction or attack. If the receiving node has an active IKE_SA to the IP address from whence the packet came, it MAY send a notification of the wayward packet over that IKE_SA in an INFORMATIONAL exchange. If it does not have such an IKE_SA, it MAY send an Informational message without cryptographic protection to the source IP address. Such a message is not part of an informational exchange, and the receiving node MUST NOT respond to it. Doing so could cause a message loop.

Identifier: RQ_002_6022
RFC Clause: 1.5.
Type: Optional
Applies to: Host

Requirement:

If an encrypted IKE packet arrives on port 500 or 4500 with an unrecognized SPI, and the receiving node does not have an active IKE_SA to the IP address from whence the packet came, it MAY send an INFORMATIONAL exchange without cryptographic protection to the source IP address containing a Notify payload set to INVALID_IKE_SPI

RFC Text:

If an encrypted IKE packet arrives on port 500 or 4500 with an unrecognized SPI, it could be because the receiving node has recently crashed and lost state or because of some other system malfunction or attack. If the receiving node has an active IKE_SA to the IP address from whence the packet came, it MAY send a notification of the wayward packet over that IKE_SA in an INFORMATIONAL exchange. **If it does not have such an IKE_SA, it MAY send an Informational message without cryptographic protection to the source IP address**. Such a message is not part of an informational exchange, and the receiving node MUST NOT respond to it. Doing so could cause a message loop

----

Identifier: RQ_002_6023
RFC Clause: 1.5
Type: Mandatory
Applies to: Host

Requirement:
If a node receives an IKE INFORMATIONAL exchange message containing a NOTIFY payload indicating an INVALID_IKE_SPI error message, it MUST NOT send a response

RFC Text:
If an encrypted IKE packet arrives on port 500 or 4500 with an unrecognized SPI, it could be because the receiving node has recently crashed and lost state or because of some other system malfunction or attack. If the receiving node has an active IKE_SA to the IP address from whence the packet came, it MAY send a notification of the wayward packet over that IKE_SA in an INFORMATIONAL exchange. If it does not have such an IKE_SA, it MAY send an Informational message without cryptographic protection to the source IP address. **Such a message is not part of an informational exchange, and the receiving node MUST NOT respond to it**. Doing so could cause a message loop

----

Identifier: RQ_002_6024
RFC Clause: 2.
Type: Mandatory
Applies to: Host

Requirement:
IKE Implementations MUST be able to send IKE messages that are up to 1280 bytes long

RFC Text:
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. 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. 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.

Identifier: RQ_002_6025
RFC Clause: 2.
Type: Mandatory
Applies to: Host

Requirement:
IKE implementations MUST be able to receive and process IKE messages that are up to 1280 bytes long

RFC Text:
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. 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. 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.

Identifier: RQ_002_6026
RFC Clause: 2.
Type: Recommended
Applies to: Host

Requirement:
IKE implementations SHOULD be able to send IKE messages that are up to 3000 bytes long

RFC Text:

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**. 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. 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.

Identifier: RQ_002_6027
RFC Clause: 2.
Type: Recommended
Applies to: Host

Requirement:

IKE implementations SHOULD be able to receive and process IKE messages that are up to 3000 bytes long

RFC Text:

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**. 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. 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.

Identifier: RQ_002_6028
RFC Clause: 2.
Type: Optional
Applies to: Host

Requirement:

IKE implementations MAY shorten IKE messages by leaving out some certificates or cryptographic suite proposals in order to keep messages below the maximum UDP message size supported.

RFC Text:

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. **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**. 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.

Identifier: RQ_002_6029
RFC Clause: 2.1.
Type: Mandatory
Applies to: Host

Requirement:

An IKE implementation MUST NOT retransmit a response to an IKE request unless it receives a retransmission of the request

RFC Text:

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.

For every pair of IKE messages, the initiator is responsible for retransmission in the event of a timeout. **The responder MUST never retransmit a response unless it receives a retransmission of the request**. In that event, the responder MUST ignore the retransmitted request except insofar as it triggers a retransmission of the response. The initiator MUST remember each request until it receives the corresponding response. 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)

Identifier: RQ_002_6030
RFC Clause: 2.1.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives a retransmitted request, it MUST retransmit the associated response but, otherwise, ignore the request.

RFC Text:
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.

For every pair of IKE messages, the initiator is responsible for retransmission in the event of a timeout. The responder MUST never retransmit a response unless it receives a retransmission of the request. **In that event, the responder MUST ignore the retransmitted request except insofar as it triggers a retransmission of the response**. The initiator MUST remember each request until it receives the corresponding response. 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)

Identifier: RQ_002_6031
RFC Clause: 2.1.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation initiates a request, it must maintain sufficient information internally to enable it to process the corresponding response correctly when it receives it

RFC Text:
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.

For every pair of IKE messages, the initiator is responsible for retransmission in the event of a timeout. The responder MUST never retransmit a response unless it receives a retransmission of the request. In that event, the responder MUST ignore the retransmitted request except insofar as it triggers a retransmission of the response. **The initiator MUST remember each request until it receives the corresponding response**. 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)

Identifier: RQ_002_6032
RFC Clause: 2.1.
Type: Mandatory
Applies to: Host

Requirement:
When responding to a request from another node, an IKE implementation MUST maintain sufficient information internally to enable it to process any subsequent retransmissions of the request correctly until it receives a further request whose sequence number is larger than the sequence number in the response plus its window size

RFC Text:
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.

For every pair of IKE messages, the initiator is responsible for retransmission in the event of a timeout. The responder MUST never retransmit a response unless it receives a retransmission of the request. In that event, the responder MUST ignore the retransmitted request except insofar as it triggers a retransmission of the response. The initiator MUST remember each request until it receives the corresponding response. **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)

Identifier: RQ_002_6033
RFC Clause: 2.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation initiates a request, it MUST continue to retransmit the request until either it receives a corresponding reply OR it deems the IKE security association to have failed

RFC Text:
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

Identifier: RQ_002_6034
RFC Clause: 2.2.
Type: Mandatory
Applies to: Host

Requirement:
The IKE_SA initial setup messages MUST use the values 0 (for the first exchange) and 1 (for the second exchange) in the Message Identifier field

RFC Text:
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.

The Message ID is a 32-bit quantity, which is zero for the first IKE request in each direction. **The IKE_SA initial setup messages will always be numbered 0 and 1**. 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. These counters increment as requests are generated and received. Responses always contain the same message ID as the corresponding request. 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.

Identifier: RQ_002_6035
RFC Clause: 2.2.
Type: Mandatory
Applies to: Host

Requirement:
Following the initial exchange values of 0 and 1, the IKE Message ID MUST be incremented with each new request message and included in its IKE header.

RFC Text:
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.

The Message ID is a 32-bit quantity, which is zero for the first IKE request in each direction. The IKE_SA initial setup messages will always be numbered 0 and 1. **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. **These counters increment as requests are generated** and received. Responses always contain the same message ID as the corresponding request. 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.

Identifier: RQ_002_6036
RFC Clause: 2.2.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE endpoint sends a response to a received IKE request it MUST set the Message ID field in the IKE Header to the value set in the IKE Header of the incoming request.

RFC Text:
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.

The Message ID is a 32-bit quantity, which is zero for the first IKE request in each direction. The IKE_SA initial setup messages will always be numbered 0 and 1. 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. These counters increment as requests are generated and received. **Responses always contain the same message ID as the corresponding request**. 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.

Identifier: RQ_002_6037
RFC Clause: 2.2.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE request message with a Message ID which is out of the expected incrementing sequence, it MUST send a NOTIFY message containing the error value, INVALID_MESSAGE_ID.

RFC Text:
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.

The Message ID is a 32-bit quantity, which is zero for the first IKE request in each direction. The IKE_SA initial setup messages will always be numbered 0 and 1. **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**. These counters increment as requests are generated and received. Responses always contain the same message ID as the corresponding request. 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.

Identifier: RQ_002_6038
RFC Clause: 2.2.
Type: Mandatory
Applies to: Host

Requirement:
If incrementing an IKE Message ID causes it to grow too large to fit in 32 bits, an IKE endpoint MUST close the corresponding IKE Security Association.

RFC Text:
The Message ID is a 32-bit quantity, which is zero for the first IKE request in each direction. The IKE_SA initial setup messages will always be numbered 0 and 1. 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. These counters increment as requests are generated and received. Responses always contain the same message ID as the corresponding request. 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.

Note that Message IDs are cryptographically protected and provide protection against message replays. **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.

Identifier: RQ_002_6039
RFC Clause: 2.3.
Type: Optional
Applies to: Host

Requirement:
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 using the SET_WINDOW_SIZE status type in a NOTIFY message.

RFC Text:
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**. 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.

Identifier: RQ_002_6040
RFC Clause: 2.3.
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY wait for a response to one request before issuing another

RFC Text:
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. 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.

Identifier: RQ_002_6041
RFC Clause: 2.3.
Type: Mandatory
Applies to: Host

Requirement:
An IKE endpoint MUST accept and process a request while it is waiting for a response to one or more of its own requests

RFC Text:
After an IKE_SA is set up, either end can initiate one or more requests. These requests may pass one another over the network. **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**. An IKE endpoint SHOULD be prepared to accept and process multiple requests while it has a request outstanding

----

Identifier: RQ_002_6042
RFC Clause: 2.3.
Type: Recommended
Applies to: Host

Requirement:
An IKE endpoint SHOULD be prepared to accept and process multiple requests while it is waiting for a response to one or more of its own requests

RFC Text:
After an IKE_SA is set up, either end can initiate one or more requests. These requests may pass one another over the network. 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. **An IKE endpoint SHOULD be prepared to accept and process multiple requests while it has a request outstanding**

Identifier: RQ_002_6043
RFC Clause: 2.3.
Type: Mandatory
Applies to: Host

Requirement:
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

RFC Text:
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.

Identifier: RQ_002_6044
RFC Clause: 2.3.
Type: Mandatory
Applies to: Host

Requirement:
An IKE endpoint MUST NOT send further IKE requests while the number of requests for which a response has not been received is greater than its peer's window size declared in a NOTIFY message with SET_WINDOW_SIZE status type

RFC Text:
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. 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. 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.

Identifier: RQ_002_6045
RFC Clause: 2.3.
Type: Mandatory
Applies to: Host

Requirement:
An IKE endpoint MUST be able to regenerate exactly each request it has sent until it receives the corresponding response

RFC Text:
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. **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**. 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.

Identifier: RQ_002_6046
RFC Clause: 2.3.
Type: Mandatory
Applies to: Host

Requirement:
An IKE endpoint MUST be able to regenerate exactly the number of previous responses equal to its declared window size if requested to do so by its peer endpoint

RFC Text:
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. **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. 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**.

Identifier: RQ_002_6047
RFC Clause: 2.3.
Type: Recommended
Applies to: Host

Requirement:
An IKE endpoint supporting a window size greater than one SHOULD be capable of processing incoming requests in any order

RFC Text:
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.

Identifier: RQ_002_6048
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:
An endpoint in an established IKE Security Association MUST conclude that the other endpoint in the SA has failed when repeated attempts to contact it have gone unanswered for a timeout period

RFC Text:
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). 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. 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. 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). 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.

Identifier: RQ_002_6049
RFC Clause: 2.4.
Type: Recommended
Applies to: Host

Requirement:
If routing information indicates to an endpoint that the other endpoint in an IKE Security Association has failed, it SHOULD initiate an empty INFORMATIONAL message to the other endpoint to determine whether it is alive.

RFC Text:

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). 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. **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. 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)**. 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.

Identifier: RQ_002_6050
RFC Clause: 2.4.
Type: Optional
Applies to: Host

Requirement:

If an endpoint in an established IKE Security Association has recently received a cryptographically protected message from the other endpoint in the SA, unprotected notifications from the same endpoint MAY be ignored

RFC Text:

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). 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. {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. 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). **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.

Identifier: RQ_002_6051
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:

IKE implementations MUST limit the rate at which they take actions based on unprotected messages

RFC Text:

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). 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. {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. 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). 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**.

Identifier: RQ_002_6052
RFC Clause: 2.4.
Type: Recommended
Applies to: Host

Requirement:
An endpoint in an established IKE Security Association SHOULD retransmit IKE messages at least twelve (12) times over a period of at least several minutes before it determines that the other endpoint has failed

RFC Text:
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. To be a good network citizen, retransmission times MUST increase exponentially to avoid flooding the network and making an existing congestion situation worse. 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. 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. 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. Note that this places requirements on the failure modes of an IKE endpoint. An implementation MUST NOT continue sending on any SA if some failure prevents it from receiving on all of the associated SAs. 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.

Identifier: RQ_002_6053
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE endpoint resends a request to which it has received no response, the time between retransmissions MUST increase exponentially

RFC Text:
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. **To be a good network citizen, retransmission times MUST increase exponentially to avoid flooding the network and making an existing congestion situation worse**. 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. 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. 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. Note that this places requirements on the failure modes of an IKE endpoint. An implementation MUST NOT continue sending on any SA if some failure prevents it from receiving on all of the associated SAs. 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.

Identifier: RQ_002_6054
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint determines that there has only been outgoing traffic on all of the Security Associations encompassed by a particular IKE_SA, it SHOULD initiate an empty INFORMATIONAL message to the other endpoint in the IKE_SA to determine whether it is alive.

RFC Text:

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. To be a good network citizen, retransmission times MUST increase exponentially to avoid flooding the network and making an existing congestion situation worse. **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**. 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. 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. Note that this places requirements on the failure modes of an IKE endpoint. An implementation MUST NOT continue sending on any SA if some failure prevents it from receiving on all of the associated SAs. 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.

Identifier: RQ_002_6055
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:

If an IKE endpoint receives no cryptographically protected messages on a specific IKE_SA or any of its CHILD_SAs within a predefined period, it SHOULD initiate an empty INFORMATIONAL message to the other endpoint in the IKE_SA to determine whether it is alive.

RFC Text:

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. To be a good network citizen, retransmission times MUST increase exponentially to avoid flooding the network and making an existing congestion situation worse. 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. **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**. 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. Note that this places requirements on the failure modes of an IKE endpoint. An implementation MUST NOT continue sending on any SA if some failure prevents it from receiving on all of the associated SAs. 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.

Identifier: RQ_002_6056
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:

An IKE implementation MUST NOT continue sending messages on any Security Association if a failure prevents it from receiving messages on all of the related Security Associations.

RFC Text:

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. To be a good network citizen, retransmission times MUST increase exponentially to avoid flooding the network and making an existing congestion situation worse. 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. 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. 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. Note that this places requirements on the failure modes of an IKE endpoint. **An implementation MUST NOT continue sending on any SA if some failure prevents it from receiving on all of the associated Sas**. 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.

Identifier: RQ_002_6057
RFC Clause: 2.4.
Type: Optional
Applies to: Host

Requirement:
The initiator of an IKE Security Association MAY 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.

RFC Text:
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

----

Identifier: RQ_002_6058
RFC Clause: 2.4.
Type: Recommended
Applies to: Host

Requirement:
If the initiator of an IKE Security Association is accepting multiple responses to its first message then when one of those responses is found to be cryptographically valid, all subsequent responses SHOULD be ignored

RFC Text:
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**

Identifier: RQ_002_6059
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint determines that the other endpoint in a Security Association is not operational, then the IKE SA and all CHILD_SAs set up through that IKE_SA MUST be deleted

RFC Text:
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.

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.**

Identifier: RQ_002_6060
RFC Clause: 2.4.
Type: Optional
Applies to: Host

Requirement:
An IKE endpoint MAY delete an inactive CHILD_SA at any time

RFC Text:
An IKE endpoint may at any time delete inactive CHILD_SAs to recover resources used to hold their state. If an IKE endpoint chooses to delete CHILD_SAs, it MUST send Delete payloads to the other end notifying it of the deletion. It MAY similarly time out the IKE_SA. 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.

Identifier: RQ_002_6061
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint chooses to delete a CHILD_SA, it MUST send a Delete payload to the other endpoint notifying it of the deletion

RFC Text:
An IKE endpoint may at any time delete inactive CHILD_SAs to recover resources used to hold their state. **If an IKE endpoint chooses to delete CHILD_SAs, it MUST send Delete payloads to the other end notifying it of the deletion**. It MAY similarly time out the IKE_SA. 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.

Identifier: RQ_002_6062
RFC Clause: 2.4.
Type: Optional
Applies to: Host

Requirement:
An IKE endpoint MAY delete an inactive IKE_SA at any time

RFC Text:
An IKE endpoint may at any time delete inactive CHILD_SAs to recover resources used to hold their state. If an IKE endpoint chooses to delete CHILD_SAs, it MUST send Delete payloads to the other end notifying it of the deletion. **It MAY similarly time out the IKE_SA**. 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.

Identifier: RQ_002_6063
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:
Closing an IKE_SA MUST also cause its associated CHILD_SAs to be closed

RFC Text:
An IKE endpoint may at any time delete inactive CHILD_SAs to recover resources used to hold their state. If an IKE endpoint chooses to delete CHILD_SAs, it MUST send Delete payloads to the other end notifying it of the deletion. It MAY similarly time out the IKE_SA. **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.

Identifier: RQ_002_6064
RFC Clause: 2.4.
Type: Recommended
Applies to: Host

Requirement:
When an IKE endpoint closes an IKE_SA it SHOULD send a Delete payload to the other endpoint in the Security Association

RFC Text:
An IKE endpoint may at any time delete inactive CHILD_SAs to recover resources used to hold their state. If an IKE endpoint chooses to delete CHILD_SAs, it MUST send Delete payloads to the other end notifying it of the deletion. It MAY similarly time out the IKE_SA. 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**.

Identifier: RQ_002_6065
RFC Clause: 2.5.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint receives a message with a higher major IKE version number than its own, it MUST drop the message

RFC Text:
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. 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.

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. If an endpoint supports major version n, and major version m, it MUST support all versions between n and m. If it receives a message with a major version that it supports, it MUST respond with that version number. 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.

Identifier: RQ_002_6066
RFC Clause: 2.5.
Type: Recommended
Applies to: Host

Requirement:
If an IKE endpoint receives a message with a higher major IKE version number than its own, it SHOULD send an unauthenticated notification message indicating INVALID_MAJOR_VERSION and containing the highest version number it supports ((MjVer field in the IKE Header)

RFC Text:
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. 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.

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**. If an endpoint supports major version n, and major version m, it MUST support all versions between n and m. If it receives a message with a major version that it supports, it MUST respond with that version number. 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.

Identifier: RQ_002_6067
RFC Clause: 2.5.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint supports major IKE version n and major IKE version m, it MUST support all IKE versions between n and m

RFC Text:
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. 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.

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. **If an endpoint supports major version n, and major version m, it MUST support all versions between n and m**. If it receives a message with a major version that it supports, it MUST respond with that version number. 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.

Identifier: RQ_002_6068
RFC Clause: 2.5.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint receives a message with a major IKE version that it supports in the IKE Header, it MUST respond with that version number in the IKE Header of the response

RFC Text:

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. 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.

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. If an endpoint supports major version n, and major version m, it MUST support all versions between n and m. **If it receives a message with a major version that it supports, it MUST respond with that version number**. 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.

Identifier: RQ_002_6069
RFC Clause: 2.5.
Type: Mandatory
Applies to: Host

Requirement:

When an IKE endpoint sends an IKE message and it is able to support a higher major IKE version number than the version indicated in the header of the message, it MUST set the V(ersion) flag in the header.

RFC Text:

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. If an endpoint supports major version n, and major version m, it MUST support all versions between n and m. If it receives a message with a major version that it supports, it MUST respond with that version number. 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.**

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. 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.

Identifier: RQ_002_6070
RFC Clause: 2.5.
Type: Mandatory
Applies to: Host

Requirement:

If an IKE endpoint has used IKE Informational messages to establish the use of a lower major IKE version number than it is able to support on a particular Security Association and then receives a message from the other endpoint in that Security Association with the V(ersion) Flag set in the IKE Header, it MUST break the connection to the other endpoint and reconnect using a higher major version

RFC Text:

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. If an endpoint supports major version n, and major version m, it MUST support all versions between n and m. If it receives a message with a major version that it supports, it MUST respond with that version number. 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.

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. **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**.

Identifier: RQ_002_6071
RFC Clause: 2.5.
Type: Recommended
Applies to: Host

Requirement:
Whenever an IKE Version 2 implementation establishes that IKE Version 1 is to be used on a particular Security Association it SHOULD note that fact in its logs

RFC Text:
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. 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.

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**.

Identifier: RQ_002_6072
RFC Clause: 2.5.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE Security Association endpoint receives an IKE request from the other endpoint with the Critical flag set in the IKE payload Header but the payload type is unrecognized, the payload MUST be rejected and the response to the request MUST include a Notify payload with the Error type set to UNSUPPORTED_CRITICAL_PAYLOAD

RFC Text:
IKEv2 adds a "critical" flag to each payload header for further flexibility for forward compatibility. **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**. If the critical flag is not set and the payload type is unsupported, that payload MUST be ignored

----

Identifier: RQ_002_6073
RFC Clause: 2.5.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE Security Association endpoint receives an IKE request from the other endpoint with the Critical flag not set in the IKE payload Header but the payload type is unrecognized, that payload MUST be ignored

RFC Text:
IKEv2 adds a "critical" flag to each payload header for further flexibility for forward compatibility. **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**. If the critical flag is not set and the payload type is unsupported, that payload MUST be ignored

----

Identifier: RQ_002_6074
RFC Clause: 2.6.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE Security Association is first established, the initiating endpoint MUST assign an Initiator's Security Parameters Index (SPI) as an identifier of the Security Association

RFC Text:
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. **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.

Identifier: RQ_002_6075
RFC Clause: 2.6.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE Security Association is first established, the receiving endpoint MUST assign a Responder's Security Parameters Index (SPI) as an identifier of the Security Association

RFC Text:
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. **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.

Identifier: RQ_002_6076
RFC Clause: 2.6.
Type: Recommended
Applies to: Host

Requirement:
The Security Parameters Index (SPI) assigned by an IKE endpoint to identify a Security Association SHOULD be unique within the context of the endpoint

RFC Text:
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. **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.

Identifier: RQ_002_6077
RFC Clause: 2.6.
Type: Mandatory
Applies to: Host

Requirement:
A Security Parameters Index (SPI) of zero (0) MUST only be used in the IKE Responder's SPI field in the IKE Header when the remote SPI value is not yet known by the sender

RFC Text:
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. 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**.

Identifier: RQ_002_6078
RFC Clause: 2.6.
Type: Mandatory
Applies to: Host

Requirement:
In the first message of an initial IKE exchange the initiator MUST set the IKE Responder's SPI field to zero (0).

RFC Text:
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.

Identifier: RQ_002_6079
RFC Clause: 2.6.
Type: Recommended
Applies to: Host

Requirement:

When an IKE endpoint detects a large number of half-open IKE_SAs it SHOULD reject further initial IKE messages unless they contain a Notify payload of type COOKIE

RFC Text:

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. **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. 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:

```
  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}
```

The first two messages do not affect any initiator or responder state except for communicating the cookie. In particular, the message sequence numbers in the first four messages will all be zero and the message sequence numbers in the last two messages will be one. 'A' is the SPI assigned by the initiator, while 'B' is the SPI assigned by the responder.

Identifier: RQ_002_6080
RFC Clause: 2.6.
Type: Recommended
Applies to: Host

Requirement:

When an IKE endpoint detects a large number of half-open IKE_SAs it SHOULD send an unprotected IKE message as a response and include COOKIE Notify payload with the cookie data to be returned

RFC Text:

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. 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**. 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:

```
  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}
```

The first two messages do not affect any initiator or responder state except for communicating the cookie. In particular, the message sequence numbers in the first four messages will all be zero and the message sequence numbers in the last two messages will be one. 'A' is the SPI assigned by the initiator, while 'B' is the SPI assigned by the responder.

Identifier: RQ_002_6081
RFC Clause: 2.6.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE endpoint receives an unprotected response to one of its IKE_SA_INIT requests and this response includes a COOKIE Notify payload with cookie data to be returned, it MUST retry the IKE_SA_INIT with a Notify payload of type COOKIE containing the responder supplied cookie data as the first payload but with all other payloads unchanged

RFC Text:
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. 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. **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:

```
  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}
```

The first two messages do not affect any initiator or responder state except for communicating the cookie. In particular, the message sequence numbers in the first four messages will all be zero and the message sequence numbers in the last two messages will be one. 'A' is the SPI assigned by the initiator, while 'B' is the SPI assigned by the responder.

Identifier: RQ_002_6082
RFC Clause: 2.6.
Type: Optional
Applies to: Host

Requirement:
An endpoint in an IKE Security Association should frequently change the value of the randomly generated secret that is used to compute cookies to be included in its responses

RFC Text:
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.

A good way to do this is to set the responder cookie to be:

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

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. 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.

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.

Identifier: RQ_002_6083
RFC Clause: 2.6.
Type: Recommended
Applies to: Host

Requirement:
If an IKE endpoint in a Security Association receives an IKE_SA_INIT request with a NOTIFY payload of type COOKIE containing a version identifier of the randomly generated secret (VersionIdofSecret) that is not the same as its own current value, it MAY reject the message by sending another response with a new cookie or it MAY keep the old value of the secret for a short time and accept cookies computed from either one

RFC Text:
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.

A good way to do this is to set the responder cookie to be:

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

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. 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.

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**.

Identifier: RQ_002_6084
RFC Clause: 2.7.
Type: Mandatory
Applies to: Host

Requirement:
When proposing a set of choices of IPSec protocols for use on an IKE Security Association, an IKE implementation MUST include one or more proposals in an IKE SA payload.

RFC Text:

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).

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:

* 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).

* 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.

Identifier: RQ_002_6085
RFC Clause: 2.7.
Type: Mandatory
Applies to: Host

Requirement:

When proposing a set of choices of IPSec protocols for use on an IKE Security Association, an IKE implementation MUST include one or more protocols in each proposal in an IKE SA payload.

RFC Text:

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).

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:

* 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).

* 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.

Identifier: RQ_002_6086
RFC Clause: 2.7.
Type: Mandatory
Applies to: Host

Requirement:
When proposing a set of choices of IPSec protocols for use on an IKE Security Association, an IKE implementation MUST include one or more transforms for each protocol in an IKE SA payload.

RFC Text:
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).

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:

* 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).

* 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.

Identifier: RQ_002_6087
RFC Clause: 2.7.
Type: Optional
Applies to: Host

Requirement:
When proposing a set of choices of IPSec protocols for use on an IKE Security Association, an IKE implementation MAY include one or more attributes for each transform in an IKE SA payload.

RFC Text:
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)**.

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:

* 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).

* 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.

Identifier: RQ_002_6088
RFC Clause: 2.7.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives a proposal for a set of choices of IPsec protocols to be used within a Security Association, it must select a single suite which may be any subset of the received proposal

RFC Text:
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).

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:

* 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).

* 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.

Identifier: RQ_002_6089
RFC Clause: 2.7.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation selects a proposal from a received set of choices of IPsec protocols to be used within a Security Association, its SA response MUST contain the same protocols in the same order as the proposal

RFC Text:
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).

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:

* 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).

* 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.

Identifier: RQ_002_6090
RFC Clause: 2.7.
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation receives a proposal for a set of choices of IPsec protocols to be used within a Security Association, it MUST select a single proposal or reject them all and return an IKE INFORMATIONAL message containing a Notify payload with the Error Type set to NO_PROPOSAL_CHOSEN

RFC Text:

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).

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:

* 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).

* 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.

Identifier: RQ_002_6091
RFC Clause: 2.7.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation selects a proposal from a received set of choices of IPsec protocols to be used within a Security Association, the selected cryptographic suite MUST contain exactly one transform of each type included in the proposal.

RFC Text:
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).

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:

* 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).

* 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.

Identifier: RQ_002_6092
RFC Clause: 2.7.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE_SA_INIT request containing an invalid Diffie-Hellman value, it must send an IKE_SA_INIT response containing a NOTIFY payload with the Error Type set to INVALID_KE_PAYLOAD indicating the correct Diffie-Hellman group

RFC Text:
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. **If the initiator guesses wrong, the responder will respond with a Notify payload of type INVALID_KE_PAYLOAD indicating the selected group**. In this case, the initiator MUST retry the IKE_SA_INIT with the corrected Diffie-Hellman group. 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.

Identifier: RQ_002_6093
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:
The security keys that are used in IKE, ESP and AH Security Associations SHOULD be used only for a limited period of time.

RFC Text:
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. When the lifetime of a security association expires, the security association MUST NOT be used. 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".

Identifier: RQ_002_6094
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:
The security keys that are used in IKE, ESP and AH Security Associations SHOULD be used to protect a limited quantity of data.

RFC Text:
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. When the lifetime of a security association expires, the security association MUST NOT be used. 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".

Identifier: RQ_002_6095
RFC Clause: 2.8.
Type: Mandatory
Applies to: Host

Requirement:
When the lifetime of a security association expires, the security association MUST NOT be used further

RFC Text:
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. **When the lifetime of a security association expires, the security association MUST NOT be used**. 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".

Identifier: RQ_002_6096
RFC Clause: 2.8.
Type: Optional
Applies to: Host

Requirement:
When the lifetime of a security association expires and if there is demand, a new security association MAY be established to replace the expired one

RFC Text:
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. When the lifetime of a security association expires, the security association MUST NOT be used. **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".

Identifier: RQ_002_6097
RFC Clause: 2.8.
Type: Optional
Applies to: Host

Requirement:
An IKE endpoint MAY rekey a CHILD_SA without restarting the entire IKE_SA

RFC Text:

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. 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. 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.

Identifier: RQ_002_6098
RFC Clause: 2.8.
Type: Optional
Applies to: Host

Requirement:

An implementation MAY refuse all CREATE_CHILD_SA requests within an IKE_SA

RFC Text:

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**. 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. 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.

Identifier: RQ_002_6099
RFC Clause: 2.8.
Type: Mandatory
Applies to: Host

Requirement:

If an IKE Security Association has expired or is about to expire and rekeying attempts fail, an implementation MUST close the IKE_SA and any associated CHILD_Sas

RFC Text:

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. **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. 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.

Identifier: RQ_002_6100
RFC Clause: 2.8.
Type: Optional
Applies to: Host

Requirement:

If an implementation has closed an IKE_SA because rekeying attempts have failed, it MAY then start new ones

RFC Text:

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. 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**. 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.

Identifier: RQ_002_6101
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:
IKE implementations SHOULD support the ability to rekey a CHILD_SA without restarting the entire IKE_SA

RFC Text:
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. 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. **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**.

Identifier: RQ_002_6102
RFC Clause: 2.8.
Type: Mandatory
Applies to: Host

Requirement:
In order to rekey a CHILD_SA within an existing IKE_SA, an IKE endpoint MUST create a new, equivalent CHILD_SA and, when the new one is established, delete the old one

RFC Text:
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. 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. 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

Identifier: RQ_002_6103
RFC Clause: 2.8.
Type: Mandatory
Applies to: Host

Requirement:
In order to rekey an IKE_SA, an IKE endpoint MUST establish a new equivalent IKE_SA with the peer with which the old IKE_SA is shared using a CREATE_CHILD_SA within the existing IKE_SA

RFC Text:
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. **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. 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

Identifier: RQ_002_6104
RFC Clause: 2.8.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE endpoint has rekeyed an IKE_SA, it MUST 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

RFC Text:

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. 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. **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 payload to delete itself MUST be the last request sent over an IKE_SA

Identifier: RQ_002_6105
RFC Clause: 2.8.
Type: Mandatory
Applies to: Host

Requirement:

When an IKE endpoint has rekeyed an IKE_SA, the message containing the payload to delete the old IKE_SA MUST be the last request sent over the old IKE_SA

RFC Text:

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. 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. 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 payload to delete itself MUST be the last request sent over an IKE_SA**

Identifier: RQ_002_6106
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:

An IKE Security Association SHOULD be rekeyed before the existing one expires and becomes unusable

RFC Text:

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

Identifier: RQ_002_6107
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:

When rekeying an IKE Security Association, an IKE endpoint SHOULD NOT finally delete the old IKE_SA until all of its current traffic has been switched over to the new IKE_SA.

RFC Text:

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**

Identifier: RQ_002_6108
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:
When the lifetime of an IKE_SA expires, an IKE endpoint SHOULD close it if there has been no traffic on it since the last time the IKE_SA was rekeyed

RFC Text:
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. 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. **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**

Identifier: RQ_002_6109
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:
The timing of rekeying requests SHOULD be delayed by a random amount of time after the need for rekeying is detected

RFC Text:
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)**.

This form of rekeying may temporarily result in multiple similar SAs between the same pairs of nodes. When there are two SAs eligible to receive packets, a node MUST accept incoming packets through either SA. 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.

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.

The node that initiated the surviving rekeyed SA SHOULD delete the replaced SA after the new one is established.

Identifier: RQ_002_6110
RFC Clause: 2.8.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE endpoint has two IKE Security Associations eligible to receive packets, it MUST accept incoming packets through either Security Association

RFC Text:
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).

This form of rekeying may temporarily result in multiple similar SAs between the same pairs of nodes. **When there are two SAs eligible to receive packets, a node MUST accept incoming packets through either SA**. 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.

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.

The node that initiated the surviving rekeyed SA SHOULD delete the replaced SA after the new one is established.

Identifier: RQ_002_6111
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:
If redundant IKE Security Associations are created as a result of both endpoints rekeying the same Security Association at the same time, the Security Association created with the lowest of the four nonces used in the two exchanges SHOULD be closed by the endpoint that created it

RFC Text:
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).

This form of rekeying may temporarily result in multiple similar SAs between the same pairs of nodes. When there are two SAs eligible to receive packets, a node MUST accept incoming packets through either SA. **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 i**t.

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.

The node that initiated the surviving rekeyed SA SHOULD delete the replaced SA after the new one is established.

Identifier: RQ_002_6112
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:
If a redundant IKE Security Association has been closed after a simultaneous rekeying, the IKE endpoint that initiated the surviving rekeyed Security Association SHOULD delete the replaced Security Association after the new one is established

RFC Text:
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).

This form of rekeying may temporarily result in multiple similar SAs between the same pairs of nodes. **When there are two SAs eligible to receive packets, a node MUST accept incoming packets through either SA**. 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.

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.

The node that initiated the surviving rekeyed SA SHOULD delete the replaced SA after the new one is established.

Identifier: RQ_002_6113
RFC Clause: 2.8.
Type: Mandatory
Applies to: Host

Requirement:
The responder to a CREATE_CHILD_SA MUST be prepared to accept messages on the Security Association before sending its response to the creation request

RFC Text:
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?

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.

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. 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.

Identifier: RQ_002_6114
RFC Clause: 2.8.
Type: Optional
Applies to: Host

Requirement:
The initiator of a CREATE_CHILD_SA MAY begin sending on the Security Association as soon as it processes the response

RFC Text:
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?

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.

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. 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.

Identifier: RQ_002_6115
RFC Clause: 2.8.
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation receives a CREAT_CHILD_SA request, it MAY begin sending on the Security Association as soon as it has sent its response to the request

RFC Text:
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?

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.

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. 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.

Identifier: RQ_002_6116
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:
When a Security Association is in the process of being rekeyed, the responding IKE endpoint SHOULD continue to send messages on the old Security Association until either (1) it has received a cryptographically valid message on the new Security Association, or (2) the new Security Association rekeys an existing Security Association and it receives an IKE request to close the replaced Security Association

RFC Text:
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?

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.

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. 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.

Identifier: RQ_002_6117
RFC Clause: 2.8.
Type: Optional
Applies to: Host

Requirement:
When a new Security Association is in the process of being established, the responding IKE endpoint MAY defer sending messages on a new SA until either it receives one itself or a timeout has occurred

RFC Text:
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?

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.

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**. 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.

Identifier: RQ_002_6118
RFC Clause: 2.8.
Type: Recommended
Applies to: Host

Requirement:
If an IKE endpoint has sent a CREATE_CHILD_SA request and receives a message on the CHILD_SA before it has received a response to its request, it SHOULD retransmit the CREATE_CHILD_SA request.

RFC Text:
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?

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.

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. **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.

Identifier: RQ_002_6119
RFC Clause: 2.8.
Type: Optional
Applies to: Host

Requirement:
An IKE endpoint MAY send a dummy message on a Security Association that it has recently created if there no incoming messages queued for that Security Association and none have been previously received

RFC Text:
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?

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.

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. 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**.

Identifier: RQ_002_6120
RFC Clause: 2.9.
Type: Mandatory
Applies to: Host

Requirement:
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

RFC Text:
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).

Identifier: RQ_002_6121
RFC Clause: 2.9.
Type: Mandatory
Applies to: Host

Requirement:
When an IP packet is received by an RFC4301-compliant IPsec subsystem and matches a "protect" selector in its Security Policy Database (SPD) and no appropriate Security Association exists yet, the IKE implementation MUST create it

RFC Text:
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).

Identifier: RQ_002_6122
RFC Clause: 2.9.
Type: Recommended
Applies to: Host

Requirement:
If an IKE implementation initiates a request for a CHILD_SA to enable it to support the particular security requirements of an incoming data packet, the first traffic selector specified in both the TSi and the TSr payloads of the request SHOULD identify the addresses in the packet triggering the request

RFC Text:
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.

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 TSi and TSr a very specific traffic selector including the addresses in the packet triggering the request**. In the example, the initiator would include in TSi 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 TSr.

Identifier: RQ_002_6123
RFC Clause: 2.9.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint receives a CHILD_SA request and is unable to accept the entire set of traffic selectors in the request but is able to accept the first selector of TSi and TSr, then it MUST send a response which identifies the initiator's first choices as the Traffic Selectors it is able to support

RFC Text:
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 TSi and TSr a very specific traffic selector including the addresses in the packet triggering the request. In the example, the initiator would include in TSi 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 TSr.

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 TSi and TSr, then the responder MUST narrow the traffic selectors to a subset that includes the initiator's first choices.

Identifier: RQ_002_6124
RFC Clause: 2.9.
Type: Optional
Applies to: Host

Requirement:
If an IKE implementation initiates a request for a CHILD_SA which is not in response to an incoming packet, the first Traffic Selectors in the TSi and TSr payloads MAY specify ranges rather than specific values

RFC Text:
If the initiator creates the CHILD_SA pair not in response to an arriving packet, but rather, say, upon startup, then there may be no specific addresses the initiator prefers for the initial tunnel over any other. **In that case, the first values in TSi and TSr MAY be ranges rather than specific values**, and the responder chooses a subset of the initiator's TSi and TSr that are acceptable. 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. 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.

Identifier: RQ_002_6125
RFC Clause: 2.9.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives a request for a CHILD_SA in which the first Traffic Selectors in the TSi and TSr payloads specify ranges rather than specific values, it MUST respond with TSi and TSr payloads indicating the subset of Traffic Selector values it is able to support

RFC Text:
If the initiator creates the CHILD_SA pair not in response to an arriving packet, but rather, say, upon startup, then there may be no specific addresses the initiator prefers for the initial tunnel over any other. In that case, the first values in TSi and TSr MAY be ranges rather than specific values, **and the responder chooses a subset of the initiator's TSi and TSr that are acceptable**. 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. 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.

Identifier: RQ_002_6126
RFC Clause: 2.9.
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation responds to a request for a CHILD_SA and includes in the response TSi and TSr payloads indicating the subset of Traffic Selector values it is able to support, it MAY also include a Notify payload of type ADDITIONAL_TS_POSSIBLE

RFC Text:
If the initiator creates the CHILD_SA pair not in response to an arriving packet, but rather, say, upon startup, then there may be no specific addresses the initiator prefers for the initial tunnel over any other. In that case, the first values in TSi and TSr MAY be ranges rather than specific values, and the responder chooses a subset of the initiator's TSi and TSr that are acceptable. 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. 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.

Identifier: RQ_002_6127
RFC Clause: 2.10.
Type: Mandatory
Applies to: Host

Requirement:
The nonces included in both IKE_INIT_SA requests and CREATE_CHILD_SA requests MUST be randomly chosen

RFC Text:
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. **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.) 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

----

Identifier: RQ_002_6128
RFC Clause: 2.10.
Type: Mandatory
Applies to: Host

Requirement:
The nonces included in both IKE_INIT_SA requests and CREATE_CHILD_SA requests MUST be at least 128 bits in length

RFC Text:
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. **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.) 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

----

Identifier: RQ_002_6129
RFC Clause: 2.10.
Type: Mandatory
Applies to: Host

Requirement:
The nonces included in both IKE_INIT_SA requests and CREATE_CHILD_SA requests MUST be at least half the key length of the pseudo-random function (prf) negotiated in the IKE exchange.

RFC Text:
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. **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.)** 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

----

Identifier: RQ_002_6130
RFC Clause: 2.11.
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST accept incoming IKE requests by responding to the address and port from which the request was received even if the source port is not 500 or 4500

RFC Text:
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. **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. IKE functions identically over IPv4 or IPv6

Identifier: RQ_002_6131
RFC Clause: 2.11.
Type: Mandatory
Applies to: Host

Requirement:
When responding to an incoming IKE request, an IKE implementation MUST specify the address and port at which the request was received as the source address and port in the response

RFC Text:
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. 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**. IKE functions identically over IPv4 or IPv6

Identifier: RQ_002_6132
RFC Clause: 2.12.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE Security Association is closed, each endpoint MUST NOT reuse either the keys used by the Security Association or any information that could be used to recompute those keys (including the secrets used in the Diffie-Hellman calculation and any data that may persist in a pseudo-random number generator that could be used to recompute the Diffie-Hellman secrets)

RFC Text:
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.

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 key}. 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

----

Identifier: RQ_002_6133
RFC Clause: 2.12.
Type: Optional
Applies to: Host

Requirement:
An IKE endpoint MAY reuse Diffie-Hellman exponentials for multiple Security Association setups

RFC Text:
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.

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.

Identifier: RQ_002_6134
RFC Clause: 2.13.
Type: Mandatory
Applies to: Host

Requirement:
When establishing an IKE Security Association, an IKE implementation MUST specify a fixed key size in the Transforms substructure of the Proposals payload even for algorithms that accept a variable length key

RFC Text:
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.

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. **For algorithms that accept a variable length key, a fixed key size MUST be specified as part of the cryptographic transform negotiated**. 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. 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.

Identifier: RQ_002_6135
RFC Clause: 2.13.
Type: Mandatory
Applies to: Host

Requirement:
When establishing an IKE Security Association, an IKE implementation MUST specify in the Transforms substructure of the Proposals payload, the algorithm by which keys are derived for encryption and integrity algorithms in which not all values are valid keys

RFC Text:
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.

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. For algorithms that accept a variable length key, a fixed key size MUST be specified as part of the cryptographic transform negotiated. **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**. 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.

Identifier: RQ_002_6136
RFC Clause: 2.14.
Type: Mandatory
Applies to: Host

Requirement:
When generating the shared security keys required for (a) establishing CHILD_SAs, (b) authentication and (c) encryption, the quantity, SKEYSEED, MUST be calculated from the nonces (Ni and Nr) exchanged in the Nonce payloads of the IKE_SA_INIT exchange and the Diffie-Hellman shared secrets (g^ir) established during that same exchange, according to the formula:

```
  SKEYSEED = prf(Ni | Nr, g^ir)
```

RFC Text:
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**. 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.

SKEYSEED and its derivatives are computed as follows:

SKEYSEED = prf(Ni | Nr, g^ir)

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

(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.

Identifier: RQ_002_6137
RFC Clause: 2.14.
Type: Mandatory
Applies to: Host

Requirement:
When generating the shared security keys required for (a) establishing CHILD_SAs, (b) authentication and (c) encryption, the derivative keys of the quantity, SKEYSEED, MUST be calculated from SKEYSEED itself, the nonces (Ni and Nr) exchanged in the Nonce payloads of the IKE_SA_INIT exchange and the Security Parameter Indexes (SPIi and SPIr) specified in the IKE Header of the IKE_SA_INIT exchange using the formula:

```
  {SK_d | SK_ai | SK_ar | SK_ei | SK_er | SK_pi | SK_pr } = prf+
            (SKEYSEED, Ni | Nr | SPIi | SPIr )
where:
  SK_d is used for deriving new keys for the CHILD_SAs established with this IKE_SA
  SK_ai and SK_ar are used as keys to the integrity protection algorithm for authenticating
        the component messages of subsequent exchanges
  SK_ei and SK_er are used for encrypting and decrypting all subsequent exchanges
  SK_pi and SK_pr are used when generating an AUTH payload.
```

RFC Text:
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. **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**.

SKEYSEED and its derivatives are computed as follows:

```
  SKEYSEED = prf(Ni | Nr, g^ir)

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

[2.14.](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.

Identifier: RQ_002_6138
RFC Clause: 2.13.
Type: Mandatory
Applies to: Host

Requirement:
When computing security keying material, an IKE implementation MUST use the Pseudo-Random Function (prf) exchanged in the Transform substructure of the Proposal payload of the relevant IKE_SA_INIT exchange

RFC Text:
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)

```
  prf+ (K,S) = T1 | T2 | T3 | T4 | ...

where:
 T1 = prf (K, S | 0x01)
 T2 = prf (K, T1 | S | 0x02)
 T3 = prf (K, T2 | S | 0x03)
 T4 = prf (K, T3 | S | 0x04)
```

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)

Identifier: RQ_002_6139
RFC Clause: 2.14.
Type: Mandatory
Applies to: Host

Requirement:
When computing the security keys for use in the establishment of CHILD_SAs, if the negotiated pseudo-random function (prf) takes a fixed-length key which is greater than the lengths of Ni and Nr, an IKE endpoint MUST construct the prf key with half the bits coming from the least significant bits of Ni and half from the least significant bits of Nr

RFC Text:
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. 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.

SKEYSEED and its derivatives are computed as follows:

```
  SKEYSEED = prf(Ni | Nr, g^ir)

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

(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**.

Identifier: RQ_002_6140
RFC Clause: 2.15.
Type: Mandatory
Applies to: Host

Requirement:
When not using extensible authentication, an IKE endpoint responding to an IKE_SA_INIT request MUST authenticate the other endpoint by:

(1) using the authentication algorithm established in the initial IKE_SA_INIT exchange to compute the security signature of the block of data which starts with the first octet of the first SPI in the header of the second message and ends with the last octet of the last payload in the second message with the initiator's nonce value, Ni , and the value prf(SK_pr,IDr') appended to it (where IDr is the responding endpoint's own ID payload excluding the fixed header); and

(2) send the computed signature to the IKE_SA_INIT initiator's endpoint in the Certificate payload of its response.

RFC Text:
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. 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. 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'). 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.

Identifier: RQ_002_6141
RFC Clause: 2.15.
Type: Mandatory
Applies to: Host

Requirement:
When not using extensible authentication, an IKE endpoint initiating an IKE_SA_INIT request MUST authenticate the other endpoint by:

(1) using the selected authentication algorithm to compute the security signature of the block of data which starts with the first octet of the first SPI in the header of the first message and ends with the last octet of the last payload in the first message with the nonce value, Ni , and the value prf(SK_pr,IDr') appended to it (where IDr is the responding endpoint's own ID payload excluding the fixed header); and

(2) send the computed signature to the other endpoint in the Certificate payload of its response.

RFC Text:
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. 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. **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'). 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.

Identifier: RQ_002_6142
RFC Clause: 2.15.
Type: Mandatory
Applies to: Host

Requirement:
When not using extensible authentication, if the first message of an INIT_IKE_SA exchange is sent twice (the second time with a responder cookie and/or a different Diffie-Hellman group), the second version of the message MUST be signed

RFC Text:
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**.

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.

Identifier: RQ_002_6143
RFC Clause: 2.15.
Type: Optional
Applies to: Host

Requirement:
Messages 3 and 4 in an IKE_SA_INIT exchange 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.

RFC Text:
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.

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.

Identifier: RQ_002_6144
RFC Clause: 2.15.
Type: Mandatory
Applies to: Host

Requirement:
If messages 3 and 4 in an IKE_SA_INIT exchange 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 MUST be computed using algorithms dictated by the type of key used by the signer and the Auth Method field in the Authentication payload

RFC Text:
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.

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.

Identifier: RQ_002_6145
RFC Clause: 2.15.
Type: Recommended
Applies to: Host

Requirement:
When not using extensible authentication, the pre-shared key used in the IKE_SA_INIT exchanges SHOULD contain as much unpredictability as the strongest key being negotiated.

RFC Text:
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**. 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. The management interface by which the Shared Secret is provided MUST accept ASCII strings of at least 64 octets and MUST NOT add a null terminator before using them as shared secrets. It MUST also accept a HEX encoding of the Shared Secret. The management interface MAY accept other encodings if the algorithm for translating the encoding to a binary string is specified. If the negotiated prf takes a fixed-size key, the shared secret MUST be of that fixed size.

Identifier: RQ_002_6146
RFC Clause: 2.15.
Type: Mandatory
Applies to: Host

Requirement:
When not using extensible authentication, if a pre-shared key is used in the IKE_SA_INIT exchanges, the associated AUTH value MUST be 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

RFC Text:
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. **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. The management interface by which the Shared Secret is provided MUST accept ASCII strings of at least 64 octets and MUST NOT add a null terminator before using them as shared secrets. It MUST also accept a HEX encoding of the Shared Secret. The management interface MAY accept other encodings if the algorithm for translating the encoding to a binary string is specified. If the negotiated prf takes a fixed-size key, the shared secret MUST be of that fixed size.

Identifier: RQ_002_6147
RFC Clause: 2.15.
Type: Optional
Applies to: Host

Requirement:
When not using extensible authentication, a pre-shared key used in the computation of the AUTH value for an IKE_SA_INIT exchange MAY be either fixed length or variable length.

RFC Text:
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. 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. The management interface by which the Shared Secret is provided MUST accept ASCII strings of at least 64 octets and MUST NOT add a null terminator before using them as shared secrets. It MUST also accept a HEX encoding of the Shared Secret. The management interface MAY accept other encodings if the algorithm for translating the encoding to a binary string is specified. If the negotiated prf takes a fixed-size key, the shared secret MUST be of that fixed size.

Identifier: RQ_002_6148
RFC Clause: 2.15.
Type: Mandatory
Applies to: Host

Requirement:
When not using extensible authentication , if the negotiated prf in an IKE_SA_INIT exchange takes a fixed-size key, the shared secret MUST be of that fixed sizes fixed length .

RFC Text:
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. 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. The management interface by which the Shared Secret is provided MUST accept ASCII strings of at least 64 octets and MUST NOT add a null terminator before using them as shared secrets. It MUST also accept a HEX encoding of the Shared Secret. The management interface MAY accept other encodings if the algorithm for translating the encoding to a binary string is specified. **If the negotiated prf takes a fixed-size key, the shared secret MUST be of that fixed size**.

Identifier: RQ_002_6149
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
The authentication protocols defined in RFC3748 MUST be used in conjunction with a public key signature based method in the authentication of the IKE_SA_INIT responder back to the initiator

RFC Text:
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. **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**. These methods are often associated with mechanisms referred to as "Legacy Authentication" mechanisms.

Identifier: RQ_002_6150
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
The additional IKE_AUTH exchanges associated with Extensible Authentication MUST be completed in order to initialize an IKE_SA

RFC Text:
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. **Extensible Authentication is implemented in IKE as additional IKE_AUTH exchanges that MUST be completed in order to initialize the IKE_SA**.

Identifier: RQ_002_6151
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If the initiator of an IKE_SA_INIT exchange is configured to use Extensible Authentication, It MUST omit the AUTH payload from the first message.

RFC Text:
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:

```
  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 }
```

Identifier: RQ_002_6152
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives the first message in an IKE_SA_INIT exchange with the AUTH payload omitted and it is capable of handling Extensible Authentication as defined in RFC3748, it MUST place an Extensible Authentication Protocol payload in message 4 of the exchange.

RFC Text:
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:

```
  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 }
```

Identifier: RQ_002_6153
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives the first message in an IKE_SA_INIT exchange with the AUTH payload omitted and it is capable of handling Extensible Authentication as defined in RFC3748, it MUST defer sending SAr2, TSi, and TSr until initiator authentication is complete in a subsequent IKE_AUTH exchange

RFC Text:
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:

```
  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 }
```

Identifier: RQ_002_6154
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation uses an Extensible Authentication Protocol method that creates a shared key as a side effect of authentication, that shared key MUST be used by the implementation to generate AUTH payloads in message 7 of the IKE_SA_INIT exchange using the syntax for shared secrets specified in section 2.15 of RFC4306

RFC Text:
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

----

Identifier: RQ_002_6155
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation uses an Extensible Authentication Protocol method that creates a shared key as a side effect of authentication, that shared key MUST be used by the implementation to generate AUTH payloads in message 8 of the IKE_SA_INIT exchange using the syntax for shared secrets specified in section 2.15 of RFC4306

RFC Text:
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

----

Identifier: RQ_002_6156
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation uses an Extensible Authentication Protocol method that creates a shared key as a side effect of authentication, that shared key MUSTNOT be used for any purpose other than the generation of AUTH payloads in messages 7 and 8 in an IKE_SA_INIT

RFC Text:
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**

Identifier: RQ_002_6157
RFC Clause: 2.16.
Type: Recommended
Applies to: Host

Requirement:
Extensible Authentication Protocol methods that do not establish a shared key SHOULD NOT be used to authenticate IKE_SA endpoints

RFC Text:
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. 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.

Identifier: RQ_002_6158
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If Extensible Authentication Protocol methods that do not generate a shared key are used in the authentication of IKE endpoints, the AUTH payload in message 7 of the IK_SA_INIT exchange MUST be generated using SK_pi

RFC Text:
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.

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.

Identifier: RQ_002_6159
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If Extensible Authentication Protocol methods that do not generate a shared key are used in the authentication of IKE endpoints, the AUTH payload in message 8 of the IK_SA_INIT exchange MUST be generated using SK_pr

RFC Text:
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. **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**.

Identifier: RQ_002_6160
RFC Clause: 2.16.
Type: Recommended
Applies to: Host

Requirement:
The initiator of an IKE_SA using Extensible Authentication Protocol SHOULD be capable of extending the initial protocol exchange to at least ten IKE_AUTH exchanges if the responder sends notification messages and/or retries the authentication prompt

RFC Text:
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. 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. Similarly, if the authentication method has failed, the responder MUST send an EAP payload containing the Failure message. The responder MAY at any time terminate the IKE exchange by sending an EAP payload containing the Failure message.

Identifier: RQ_002_6161
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation uses an Extensible Authentication Protocol method to authenticate the other endpoint in an IKE_SA, the responder MUST send an EAP payload containing the Success message when the authentication method has successfully terminated

RFC Text:
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. **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**. Similarly, if the authentication method has failed, the responder MUST send an EAP payload containing the Failure message. The responder MAY at any time terminate the IKE exchange by sending an EAP payload containing the Failure message.

Identifier: RQ_002_6162
RFC Clause: 2.16.
Type: Optional
Applies to: Host

Requirement:
If an IKE implementation uses an Extensible Authentication Protocol method to authenticate the other endpoint in an IKE_SA, the responder MUST send a NOTIFY message containing an AUTHENTICATION_FAILED error type in the event that the authentication method does not terminate successfully

RFC Text:
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. 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. **Similarly, if the authentication method has failed, the responder MUST send an EAP payload containing the Failure message.** The responder MAY at any time terminate the IKE exchange by sending an EAP payload containing the Failure message.

Identifier: RQ_002_6163
RFC Clause: 2.16.
Type: Optional
Applies to: Host

Requirement:
If an IKE implementation uses an Extensible Authentication Protocol method to authenticate the other endpoint in an IKE_SA, the responder MAY send a NOTIFY message containing an AUTHENTICATION_FAILED error type at any time to terminate the IKE exchange

RFC Text:
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. 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. Similarly, if the authentication method has failed, the responder MUST send an EAP payload containing the Failure message. The **responder MAY at any time terminate the IKE exchange by sending an EAP payload containing the Failure message**. Following such an extended exchange, the EAP AUTH payloads MUST be included in the two messages following the one containing the EAP Success message.

Identifier: RQ_002_6164
RFC Clause: 2.16.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation uses an Extensible Authentication Protocol method to authenticate the other endpoint in an IKE_SA and the authentication is successful, the EAP AUTH payloads MUST be included in the two messages following the one containing the EAP Success message

RFC Text:
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. 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. Similarly, if the authentication method has failed, the responder MUST send an EAP payload containing the Failure message. The responder MAY at any time terminate the IKE exchange by sending an EAP payload containing the Failure message. **Following such an extended exchange, the EAP AUTH payloads MUST be included in the two messages following the one containing the EAP Success message**.

Identifier: RQ_002_6165
RFC Clause: 2.17.
Type: Mandatory
Applies to: Host

Requirement:

When a first CHILD_SA is created by an IKE endpoint as a result of an IKE_AUTH exchange, the endpoint MUST generate the associated keying material using the algorithm:

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

Where Ni and Nr are the nonces from the IKE_SA_INIT exchange

RFC Text:

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:

KEYMAT = prf+(SK_d, Ni | Nr)

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.

Identifier: RQ_002_6166
RFC Clause: 2.17.
Type: Mandatory
Applies to: Host

Requirement:

When an additional CHILD_SA is created by an IKE endpoint using a CREATE_CHILD_SA exchange, the endpoint MUST generate the associated keying material using the algorithm:

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

Where Ni and Nr are the nonces from the CREATE_CHILD_SA exchange

RFC Text:

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:

KEYMAT = prf+(SK_d, Ni | Nr)

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.

Identifier: RQ_002_6167
RFC Clause: 2.17.
Type: Mandatory
Applies to: Host

Requirement:

When an IKE endpoint initiates a CREATE_CHILD_SA exchange which includes a Diffie-Hellman exchange, it MUST generate the necessary keying material using the following algorithm:

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

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).

RFC Text:

For CREATE_CHILD_SA exchanges including an optional Diffie-Hellman exchange, the keying material is defined as:

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

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).

Identifier: RQ_002_6168
RFC Clause: 2.17.
Type: Mandatory
Applies to: Host

Requirement:

If multiple IPsec protocols have been negotiated in the establishment of an IKE Security Association then the key sets for each protocol MUST be extracted from the keying material (KEYMAT), generated from pre-set and transmitted parameters, in the order (big-endian) in which the protocol headers will appear in the encapsulated IKE_SA_INIT packet

[See also RQ_SEC_6169, RQ_SEC_6170 and RQ_SEC_6171]

RFC Text:

Keying material MUST be taken from the expanded KEYMAT in the following order:

All keys for SAs carrying data from the initiator to the responder are taken before SAs going in the reverse direction.

If multiple IPsec protocols are negotiated, keying material is taken in the order in which the protocol headers will appear in the encapsulated packet.

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.

Identifier: RQ_002_6169
RFC Clause: 2.17.
Type: Mandatory
Applies to: Host

Requirement:
For each IPsec protocol negotiated in the establishment of an IKE Security Association, the key sets for each SA carrying data from the initiator to the responder MUST be extracted from the keying material (KEYMAT), generated from pre-set and transmitted parameters, before (in big-endian order) the key sets for the SAs carrying data from the responder to the initiator

[See also RQ_SEC_6168, RQ_SEC_6170 and RQ_SEC_6171]

RFC Text:
Keying material MUST be taken from the expanded KEYMAT in the following order:

All keys for SAs carrying data from the initiator to the responder are taken before SAs going in the reverse direction.

If multiple IPsec protocols are negotiated, keying material is taken in the order in which the protocol headers will appear in the encapsulated packet.

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.

Identifier: RQ_002_6170
RFC Clause: 2.17.
Type: Mandatory
Applies to: Host

Requirement:
For each established IKE Security Association, if the security protocol requires both encryption and authentication keys, the encryption key MUST be extracted from the first octets (big-endian) of the keying material (KEYMAT), generated from pre-set and transmitted parameters.

[See also RQ_SEC_6168, RQ_SEC_6169 and RQ_SEC_6171]

RFC Text:
Keying material MUST be taken from the expanded KEYMAT in the following order:

All keys for SAs carrying data from the initiator to the responder are taken before SAs going in the reverse direction.

If multiple IPsec protocols are negotiated, keying material is taken in the order in which the protocol headers will appear in the encapsulated packet.

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.

Identifier: RQ_002_6171
RFC Clause: 2.17.
Type: Mandatory
Applies to: Host

Requirement:
For each established IKE Security Association, if the security protocol requires both encryption and authentication keys, the authentication key MUST be extracted from the octets following (big-endian) the encryption key in the keying material (KEYMAT), generated from pre-set and transmitted parameters.

[See also RQ_SEC_6168, RQ_SEC_6169 and RQ_SEC_6170]

RFC Text:
Keying material MUST be taken from the expanded KEYMAT in the following order:

All keys for SAs carrying data from the initiator to the responder are taken before SAs going in the reverse direction.

If multiple IPsec protocols are negotiated, keying material is taken in the order in which the protocol headers will appear in the encapsulated packet.

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.

Identifier: RQ_002_6172
RFC Clause: 2.18.
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY use a CREATE_CHILD_SA exchange to rekey an existing IKE Security Association

RFC Text:
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:

SKEYSEED = prf(SK_d (old), [g^ir (new)] | Ni | Nr)

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.

Identifier: RQ_002_6173
RFC Clause: 2.18.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation uses a CREATE_CHILD_SA exchange to rekey an existing IKE Security Association, it MUST omit the Traffic Selector payloads from the exchange messages

RFC Text:
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:

```
  SKEYSEED = prf(SK_d (old), [g^ir (new)] | Ni | Nr)
```

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.

Identifier: RQ_002_6174
RFC Clause: 2.18.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation uses a CREATE_CHILD_SA exchange to rekey an existing IKE Security Association, it MUST compute SKEYSEED for the new IKE_SA using SK_d from the existing IKE_SA as follows:

```
    SKEYSEED = prf(SK_d (old), [g^ir (new)] | Ni | Nr)
```

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.

RFC Text:
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:**

```
  SKEYSEED = prf(SK_d (old), [g^ir (new)] | Ni | Nr)
```

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.

Identifier: RQ_002_6175
RFC Clause: 2.18.
Type: Mandatory
Applies to: Host

Requirement:
When and IKE implementation uses the CREATE_CHILD_SA exchange to rekey an existing IKE Security Association. it MUST reset the message counters on the rekeyed IKE_SA to zero (0)

RFC Text:
The new IKE_SA MUST reset its message counters to 0.

Identifier: RQ_002_6176
RFC Clause: 2.19.
Type: Mandatory
Applies to: Host

Requirement:
In order to request a temporary IP address in a network protected by a security gateway, an IKE endpoint MAY request the creation of a CHILD_SA to the gateway and include in this request a Configuration Payload (CP) with the CFG Type set to CFG_REQUEST and the Attribute Type set to INTERNAL_IP6_ADDRESS

RFC Text:
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.

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). 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. 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.

Identifier: RQ_002_6177
RFC Clause: 2.19.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE security gateway receives an IK_AUTH request containing a Configuration Payload (CP) with the CFG Type set to CFG_REQUEST and the Attribute Type set to INTERNAL_IP6_ADDRESS from an IKE endpoint, it MUST include in the IK_AUTH response a Configuration Payload (CP) with the CFG Type set to CFG_REPLY, the Attribute Type set to INTERNAL_IP6_ADDRESS and the Attribute Value containing the temporary IP address to be used by the requesting endpoint

RFC Text:
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.

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). 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. 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.

Identifier: RQ_002_6178
RFC Clause: 2.19.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends either an IKE_AUTH request or an IKE_AUTH response containing a Configuration Payload (CP), it MUST insert the CP payload before the SA payload.

RFC Text:
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). 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. 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.

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.

CP(CFG_REQUEST) MUST contain at least an INTERNAL_ADDRESS attribute (either IPv4 or IPv6) but MAY contain any number of additional attributes the initiator wants returned in the response.

Identifier: RQ_002_6179
RFC Clause: 2.19.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation is required to send multiple IKE_AUTH messages (requests or responses) to establish a CHILD_SA between an endpoint and a security gateway, the Configuration Payloads (CP) MUST be included in each message containing the SA payload.

RFC Text:
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). 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. 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.

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**.

CP(CFG_REQUEST) MUST contain at least an INTERNAL_ADDRESS attribute (either IPv4 or IPv6) but MAY contain any number of additional attributes the initiator wants returned in the response.

Identifier: RQ_002_6180
RFC Clause: 2.19.
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation is required to send an IKE_AUTH messages containing a Configuration Payload (CP) with the CFG Type set to CFG_REQUEST, it MAY contain any number of attributes the initiator wants returned in the response in addition to the mandatory INTERNAL_IP6_ADDRESS attribute type

RFC Text:
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). 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. 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.

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.

CP(CFG_REQUEST) MUST contain at least an INTERNAL_ADDRESS attribute (either IPv4 or IPv6) but MAY contain any number of additional attributes the initiator wants returned in the response.

Identifier: RQ_002_6181
RFC Clause: 2.19.
Type: Optional
Applies to: Host

Requirement:
When an IKE security gateway receives an IKE_AUTH request containing a Configuration Payload (CP) with the CFG Type set to CFG_REQUEST, it MAY return additional configuration attributes that were not included in the original request

RFC Text:
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.

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. 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.

Identifier: RQ_002_6182
RFC Clause: 2.19.
Type: Optional
Applies to: Host

Requirement:
When an IKE security gateway receives an IKE_AUTH request containing a Configuration Payload (CP) with the CFG Type set to CFG_REQUEST, it MAY ignore any requested non-mandatory attributes that it does not support

RFC Text:
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**.

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. 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.

Identifier: RQ_002_6183
RFC Clause: 2.19.
Type: Mandatory
Applies to: Host

Requirement:
An IKE security gateway MUST NOT send a Configuration Payload (CP) with the CFG Type set to CFG_REPLY without having first received a CP with the CFG Type set to CFG_REQUEST from the initiating endpoint

RFC Text:
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.

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. 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.

Identifier: RQ_002_6184
RFC Clause: 2.19.
Type: Mandatory
Applies to: Host

Requirement:
If a security gateway is configured to expect an IKE_AUTH request from a particular endpoint to include a Configuration Payload with the CFG Type set to CFG_REQUEST, it MUST terminate any IKE_AUTH request from this endpoint if the CFG_REQUEST is not included and send a Notify payload to the endpoint with an Error Type set to FAILED_CP_REQUIRED.

RFC Text:
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.

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. **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**.

Identifier: RQ_002_6185
RFC Clause: 2.20.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint receives an INFORMATIONAL exchange message after the IKE_SA and first CHILD_SA have been established and it contains a Configuration (CP) Payload with the CFG Type set to CFG_REQUEST and the Attribute Type set to APPLICATION_VERSION, it MUST send one of the following to the peer IKE endpoint:

(i) an INFORMATIONAL exchange message containing a CP payload with the CFG Type set to CFG_REPLY, the Attribute Type set to APPLICATION_VERSION and the Attribute Value set to a string containing information regarding the endpoint's software version;
(ii) an INFORMATIONAL exchange message containing a CP payload with the CFG Type set to CFG_REPLY, the Attribute Type set to APPLICATION_VERSION and the Attribute Value set to an empty string; or
(iii) an INFORMATIONAL exchange message containing no CP payload if CP is not supported.

RFC Text:
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.

```
 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.")
```

Identifier: RQ_002_6186
RFC Clause: 2.21.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE request that is badly formatted or unacceptable for reasons of policy, its response MUST contain a Notify payload indicating the error

RFC Text:
There are many kinds of errors that can occur during IKE processing. **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**. If an error occurs outside the context of an IKE request (e.g., the node is getting ESP messages on a nonexistent SPI), the node SHOULD initiate an INFORMATIONAL exchange with a Notify payload describing the problem

----

Identifier: RQ_002_6187
RFC Clause: 2.21.
Type: Recommended
Applies to: Host

Requirement:
If an IKE implementation detects an error outside the context of an IKE request (e.g., the node is getting ESP messages on a nonexistent SPI), it SHOULD initiate an INFORMATIONAL exchange with a Notify payload describing the problem

RFC Text:
There are many kinds of errors that can occur during IKE processing. 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. **If an error occurs outside the context of an IKE request (e.g., the node is getting ESP messages on a nonexistent SPI), the node SHOULD initiate an INFORMATIONAL exchange with a Notify payload describing the problem**

Identifier: RQ_002_6188
RFC Clause: 2.21.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint receives an IKE exchange response on UDP port 500 or 4500 outside the context of an IKE_SA known to it, it MUST NOT send any response to the message.

RFC Text:
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. **If the message is marked as a response, the node MAY audit the suspicious event but MUST NOT respond.** If the message is marked as a request, the node MAY audit the suspicious event and MAY send a response. 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.

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. 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.

Identifier: RQ_002_6189
RFC Clause: 2.21.
Type: Optional
Applies to: Host

Requirement:
If an IKE endpoint receives an IKE exchange request on UDP port 500 or 4500 outside the context of an IKE_SA known to it, it MAY send a response to the message.

RFC Text:
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. If the message is marked as a response, the node MAY audit the suspicious event but MUST NOT respond. **If the message is marked as a request, the node MAY audit the suspicious event and MAY send a response**. 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.

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. 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.

Identifier: RQ_002_6190
RFC Clause: 2.21.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint responds to IKE exchange request received on UDP port 500 or 4500 but outside the context of an IKE_SA known to it, it MUST send the response to the IP address and port from whence it came with the same IKE SPIs and the Message ID copied.

RFC Text:
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. If the message is marked as a response, the node MAY audit the suspicious event but MUST NOT respond. If the message is marked as a request, the node MAY audit the suspicious event and MAY send a response. **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.

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. 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.

Identifier: RQ_002_6191
RFC Clause: 2.21.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint responds to IKE exchange request received on UDP port 500 or 4500 but outside the context of an IKE_SA known to it, it MUST NOT cryptographically protect the response

RFC Text:
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. If the message is marked as a response, the node MAY audit the suspicious event but MUST NOT respond. If the message is marked as a request, the node MAY audit the suspicious event and MAY send a response. 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.

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. 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.

Identifier: RQ_002_6192
RFC Clause: 2.21.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint responds to IKE exchange request received on UDP port 500 or 4500 but outside the context of an IKE_SA known to it, it MUST include a Notify payload indicating INVALID_IKE_SPI in the response

RFC Text:
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. If the message is marked as a response, the node MAY audit the suspicious event but MUST NOT respond. If the message is marked as a request, the node MAY audit the suspicious event and MAY send a response. 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**.

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. 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.

Identifier: RQ_002_6193
RFC Clause: 2.21.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint receives a cryptographically unprotected IKE response containing a Notify payload indicating INVALID_IKE_SPI, it MUST NOT respond in any way to this message

RFC Text:
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. If the message is marked as a response, the node MAY audit the suspicious event but MUST NOT respond. If the message is marked as a request, the node MAY audit the suspicious event and MAY send a response. 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.

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. 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.

Identifier: RQ_002_6194
RFC Clause: 2.21.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint receives a cryptographically unprotected IKE response containing a Notify payload indicating INVALID_IKE_SPI, it MUST NOT change the state of any existing Security Associations

RFC Text:
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. If the message is marked as a response, the node MAY audit the suspicious event but MUST NOT respond. If the message is marked as a request, the node MAY audit the suspicious event and MAY send a response. 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.

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. 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.

Identifier: RQ_002_6195
RFC Clause: 2.21.
Type: Recommended
Applies to: Host

Requirement:
If an IKE endpoint receives a cryptographically unprotected IKE response containing a Notify payload indicating INVALID_IKE_SPI, it SHOULD initiate a liveness test for the IKE_SA on which the unprotected response was received

RFC Text:
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. If the message is marked as a response, the node MAY audit the suspicious event but MUST NOT respond. If the message is marked as a request, the node MAY audit the suspicious event and MAY send a response. 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.

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. **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.

Identifier: RQ_002_6196
RFC Clause: 2.21.
Type: Optional
Applies to: Host

Requirement:
If an IKE endpoint receives a cryptographically unprotected, unsolicited or otherwise unexpected message from the other endpoint in an established IKE Security Association, it MAY send an IKE Notify payload in an IKE INFORMATIONAL exchange over that SA

RFC Text:
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. The recipient MUST NOT change the state of any SA's as a result but SHOULD audit the event to aid in diagnosing malfunctions. A node MUST limit the rate at which it will send messages in response to unprotected messages.

Identifier: RQ_002_6197
RFC Clause: 2.21.
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint receives a cryptographically unprotected, unsolicited or otherwise unexpected message from the other endpoint in an established IKE Security Association, it MUST NOT change the state of any SA's as a result

RFC Text:
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. **The recipient MUST NOT change the state of any SA's as a result** but SHOULD audit the event to aid in diagnosing malfunctions. A node MUST limit the rate at which it will send messages in response to unprotected messages.

Identifier: RQ_002_6198
RFC Clause: 2.21.
Type: Recommended
Applies to: Host

Requirement:
If an IKE endpoint receives a cryptographically unprotected, unsolicited or otherwise unexpected message from the other endpoint in an established IKE Security Association, it SHOULD record the event to aid in diagnosing malfunctions

RFC Text:
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. The recipient MUST NOT change the state of any SA's as a result **but SHOULD audit the event to aid in diagnosing malfunctions**. A node MUST limit the rate at which it will send messages in response to unprotected messages.

Identifier: RQ_002_6199
RFC Clause: 2.22.
Type: Optional
Applies to: Host

Requirement:
An IKE endpoint requesting a CHILD_SA MAY advertise its support for one or more compression algorithms through one or more Notify payloads of type IPCOMP_SUPPORTED

RFC Text:
Negotiation of IP compression is separate from the negotiation of cryptographic parameters associated with a CHILD_SA. **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**. The response MAY indicate acceptance of a single compression algorithm with a Notify payload of type IPCOMP_SUPPORTED. These payloads MUST NOT occur in messages that do not contain SA payloads.

Identifier: RQ_002_6200
RFC Clause: 2.22.
Type: Mandatory
Applies to: Host

Requirement:
An IKE endpoint receiving a CHILD_SA request advertising support for one or more compression algorithms through one or more Notify payloads of type IPCOMP_SUPPORTED MUST NOT indicate acceptance of more than a single compression algorithm with a Notify payload of type IPCOMP_SUPPORTED

RFC Text:
Negotiation of IP compression is separate from the negotiation of cryptographic parameters associated with a CHILD_SA. 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. **The response MAY indicate acceptance of a single compression algorithm with a Notify payload of type IPCOMP_SUPPORTED**. These payloads MUST NOT occur in messages that do not contain SA payloads.

Identifier: RQ_002_6201
RFC Clause: 2.22.
Type: Mandatory
Applies to: Host

Requirement:
An IKE endpoint MUST NOT include Notify payloads of type IPCOMP_SUPPORTED in messages that do not contain SA payloads

RFC Text:
Negotiation of IP compression is separate from the negotiation of cryptographic parameters associated with a CHILD_SA. 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. The response MAY indicate acceptance of a single compression algorithm with a Notify payload of type IPCOMP_SUPPORTED. **These payloads MUST NOT occur in messages that do not contain SA payloads**.

Identifier: RQ_002_6202
RFC Clause: 2.22.
Type: Mandatory
Applies to: Host

Requirement:
An endpoint in an IKE Security Association MUST NOT accept an IP compression algorithm that was not included in the set of available algorithms proposed to the other endpoint during a CREATE_CHILD_SA exchange

RFC Text:
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, **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.

Identifier: RQ_002_6203
RFC Clause: 2.22.
Type: Mandatory
Applies to: Host

Requirement:
An endpoint in an IKE Security Association MUST NOT accept more than one IP compression algorithm from the set of available algorithms proposed by the other endpoint during a CREATE_CHILD_SA exchange

RFC Text:
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, 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.

Identifier: RQ_002_6204
RFC Clause: 2.22.
Type: Mandatory
Applies to: Host

Requirement:
An endpoint in an IKE Security Association MUST use only the IP compression algorithm proposed and accepted during a CREATE_CHILD_SA exchange

RFC Text:
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, 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**.

Identifier: RQ_002_6205
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE request, it MUST set both the Source Port and the Destination Port in the UDP Header to 500

RFC Text:
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.

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).

Identifier: RQ_002_6206
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST ACCEPT IKE messages with any source port number set in the UDP Header

RFC Text:
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.

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).

Identifier: RQ_002_6207
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation supports NAT Traversal, it must set the Destination Port number in the UDP Header of an IKE response message to the Source Port number set in the UDP Header of the associated received message

RFC Text:
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.

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).

Identifier: RQ_002_6208
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:
If an IKE endpoint determines that a Network Address Translation (NAT) gateway exists between itself and the other endpoint in an IKE Security Association, it MUST set the Source and the Destination Port number to 4500 in the UDP Header of all subsequent messages to the other endpoint

RFC Text:
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.

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).

Identifier: RQ_002_6209
RFC Clause: 2.23
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY support NAT traversal

RFC Text:
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.

* IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived.

* 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).

* 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.

* 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].

* 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. 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.

* 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.

* 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.

Identifier: RQ_002_6210
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation supports NAT Traversal, an IKE implementation MUST respond to IKE messages received on UDP port 500

RFC Text:
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.

* **IKE MUST listen on port 4500 as well as port 500**. IKE MUST respond to the IP address and port from which packets arrived.

* 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).

* 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.

* 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].

* 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. 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.

* 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.

* 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.

Identifier: RQ_002_6211
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation supports NAT Traversal , it MUST respond to IKE messages received on UDP port 4500

RFC Text:
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.

```
 * IKE MUST listen on port 4500 as well as port 500. IKE MUST
   respond to the IP address and port from which packets arrived.

 * 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).

 * 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.

 * 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].

 * 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. 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.

 * 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.

 * 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.
```

Identifier: RQ_002_6212
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation supports NAT Traversal, it must set the Destination Port number in the UDP Header of an IKE response message to the Source Port number set in the UDP Header of the associated received message

RFC Text:

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.

* IKE MUST listen on port 4500 as well as port 500. **IKE MUST respond to the IP address and port from which packets arrived.**

* 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).

* 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.

* 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].

* 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. 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.

* 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.

* 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.

Identifier: RQ_002_6213
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation supports NAT Traversal, it must set the Destination IP Address in the IPv6 Header of an IKE response message to the Source IP Address set in the IPv6 Header of the associated received message

RFC Text:
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.

* IKE MUST listen on port 4500 as well as port 500. **IKE MUST respond to the IP address and port from which packets arrived.**

* 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).

* 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.

* 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].

* 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. 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.

* 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.

* 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.

Identifier: RQ_002_6214
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation supports NAT Traversal , it MUST include Notify payloads of type NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP in any IKE_SA_INIT request

RFC Text:

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.

* IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived.

* **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).

* 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.

* 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].

* 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. 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.

* 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.

* 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.

Identifier: RQ_002_6215
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation supports NAT Traversal , it MUST include Notify payloads of type NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP in any IKE_SA_INIT response

RFC Text:

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.

* IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived.

* **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).

* 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.

* 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].

* 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. 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.

* 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.

* 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.

Identifier: RQ_002_6216
RFC Clause: 2.23
Type: Recommended
Applies to: Host

Requirement:
If an IKE implementation supports NAT Traversal and it receives a NAT_DETECTION_DESTINATION_IP payload that does not match the SHA-1 hash of the destination IP address in the IPv6 Header and the port number found in the UDP Header of the packet containing the payload, it SHOULD start sending keepalive packets as defined in RFC3948

RFC Text:
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.

* IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived.

* 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).

* 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.

* **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].**

* 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. 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.

* 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.

* 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.

Identifier: RQ_002_6217
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation supports NAT Traversal and it receives a NAT_DETECTION_DESTINATION_IP payload that does not match the SHA-1 hash of the destination IP address in the IPv6 Header and the port number found in the UDP Header of the packet containing the payload, all subsequent IKE packets associated with this Security Association MUST be inserted immediately after the UDP Header but preceded by four bytes of zero and sent from UDP port 4500

RFC Text:

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.

* IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived.

* 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).

* 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.

* 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].

* **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. 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.

* 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.

* 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.

Identifier: RQ_002_6218
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation supports NAT Traversal and it receives a NAT_DETECTION_SOURCE_IP payload that does not match the SHA-1 hash of the source IP address in the IPv6 Header and the port number found in the UDP Header of the packet containing the payload, all subsequent IKE packets associated with this Security Association MUST be inserted immediately after the UDP Header but preceded by four bytes of zero and sent from UDP port 4500

RFC Text:

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.

* IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived.

* 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).

* 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.

* 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].

* **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. 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.**

* 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.

* 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.

Identifier: RQ_002_6219
RFC Clause: 2.23
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation supports NAT Traversal, any Traffic Selector MUST contain exactly one IP address (i.e. Starting Address and Ending Address must be set to the same value)

RFC Text:

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.

* IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived.

* 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).

* 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.

* 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].

* 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. 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.

* 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.**

* 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.

Identifier: RQ_002_6220
RFC Clause: 2.23
Type: Recommended
Applies to: Security Association

Requirement:

If an IKE implementation supports NAT Traversal and it has not detected the presence of a NAT between itself and the other endpoint in an IKE Security Association, it SHOULD send all packets (including retransmission packets) to the IP address and port from the last valid authenticated packet from the other endpoint

RFC Text:

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.

* IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived.

* 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).

* 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.

* 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].

* 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. 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.

* 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.

* 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.

Identifier: RQ_002_6221
RFC Clause: 2.23
Type: Recommended
Applies to: Security Association

Requirement:

If an IKE implementation supports NAT Traversal and it has detected the presence of a NAT gateway between itself and the other endpoint in an IKE Security Association, it SHOULD NOT send any packets (including retransmission packets) to the IP address and port from the last valid authenticated packet from the other endpoint

RFC Text:

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.

* IKE MUST listen on port 4500 as well as port 500. IKE MUST respond to the IP address and port from which packets arrived.

* 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).

* 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.

* 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].

* 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. 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.

* 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.

* 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.

Identifier: RQ_002_6222
RFC Clause: 2.24
Type: Mandatory
Applies to: Host

Requirement:
When encapsulating or decapsulating packets for all tunnel-mode Security Associations created by IKE, an endpoint MUST support the Explicit Congestion Notification (ECN) full-functionality for tunnels specified in RFC3168.

RFC Text:
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]). IKEv2 simplifies this situation by requiring that ECN be usable in the outer IP headers of all tunnel-mode IPsec SAs created by IKEv2. 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.

Identifier: RQ_002_6223
RFC Clause: 2.24
Type: Mandatory
Applies to: Host

Requirement:
When encapsulating or decapsulating packets for all tunnel-mode Security Associations created by IKE, an endpoint MUST implement the tunnel encapsulation and decapsulation processing specified in RFC4301

RFC Text:
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]). IKEv2 simplifies this situation by requiring that ECN be usable in the outer IP headers of all tunnel-mode IPsec SAs created by IKEv2. 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.

Identifier: RQ_002_6224
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message on UDP port 500, it MUST insert that message immediately following the UDP Header in the packet

RFC Text:
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. **When sent on UDP port 500, IKE messages begin immediately following the UDP header**. When sent on UDP port 4500, IKE messages have prepended four octets of zero. 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. If a payload of type "Encrypted" is found, that payload is decrypted and its contents parsed as additional payloads. An Encrypted payload MUST be the last payload in a packet and an Encrypted payload MUST NOT contain another Encrypted payload

----

Identifier: RQ_002_6225
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message on UDP port 4500, it MUST insert that message immediately following the UDP Header in the packet but preceded by four octets of zero, i.e.:

```
+---------------------+----+---+---+---+--------------------- - - -
|  UDP Header         |  0 : 0 : 0 : 0 |  IKE Message
+---------------------+----+---+---+---+--------------------- - - -
```

RFC Text:
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. When sent on UDP port 500, IKE messages begin immediately following the UDP header. **When sent on UDP port 4500, IKE messages have prepended four octets of zero.** 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. If a payload of type "Encrypted" is found, that payload is decrypted and its contents parsed as additional payloads. An Encrypted payload MUST be the last payload in a packet and an Encrypted payload MUST NOT contain another Encrypted payload

----

Identifier: RQ_002_6226
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
Each IKE message MUST consist of one IKE header followed by one or more IKE payloads

RFC Text:
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. When sent on UDP port 500, IKE messages begin immediately following the UDP header. When sent on UDP port 4500, IKE messages have prepended four octets of zero. 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. If a payload of type "Encrypted" is found, that payload is decrypted and its contents parsed as additional payloads. An Encrypted payload MUST be the last payload in a packet and an Encrypted payload MUST NOT contain another Encrypted payload

----

Identifier: RQ_002_6227
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE message containing a payload of type "Encrypted", it MUST decrypt that payload and parse the contents as additional payloads

RFC Text:
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. When sent on UDP port 500, IKE messages begin immediately following the UDP header. When sent on UDP port 4500, IKE messages have prepended four octets of zero. 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. **If a payload of type "Encrypted" is found, that payload is decrypted and its contents parsed as additional payloads**. An Encrypted payload MUST be the last payload in a packet and an Encrypted payload MUST NOT contain another Encrypted payload

----

Identifier: RQ_002_6228
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When constructing an IKE packet which is to contain an encrypted payload, an IKE implementation MUST place the encrypted payload as the last payload in the packet

RFC Text:
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. When sent on UDP port 500, IKE messages begin immediately following the UDP header. When sent on UDP port 4500, IKE messages have prepended four octets of zero. 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. If a payload of type "Encrypted" is found, that payload is decrypted and its contents parsed as additional payloads. **An Encrypted payload MUST be the last payload in a packet** and an Encrypted payload MUST NOT contain another Encrypted payload

----

Identifier: RQ_002_6229
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When constructing an IKE packet, an IKE implementation MUST NOT include more than one encrypted payload in the packet

RFC Text:
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. When sent on UDP port 500, IKE messages begin immediately following the UDP header. When sent on UDP port 4500, IKE messages have prepended four octets of zero. 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. If a payload of type "Encrypted" is found, that payload is decrypted and its contents parsed as additional payloads. An Encrypted payload MUST be the last payload in a packet and **an Encrypted payload MUST NOT contain another Encrypted payload**

Identifier: RQ_002_6230
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
All multi-octet fields representing integers in an IKE header MUST be encoded with the most significant byte first (i.e. network byte or big-endian order)

RFC Text:
All multi-octet fields representing integers are laid out in big endian order (aka most significant byte first, or network byte order)

Identifier: RQ_002_6231
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:

```
The header in an IKE packet MUST be in the following format:

Octets               Field
--------------------------
1  - 8               IKE_SA Initiator's SPI
9  - 16              IKE_SA Responder's SPI
17                   Next Payload indicator
18 (bits 0 - 3)      Major Version number
18 (bits 4 - 7)      Minor Version number
19                   Exchange Type
20                   Flags
21 - 24              Message Identifier
25 - 28              Length
```

RFC Text:

The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

```
 o  Initiator's SPI (8 octets) - A value chosen by the
    initiator to identify a unique IKE security association. This
    value MUST NOT be zero.

 o  Responder's SPI (8 octets) - A value chosen by the
    responder to identify a unique IKE security association. This
    value MUST be zero in the first message of an IKE Initial
    Exchange (including repeats of that message including a
    cookie) and MUST NOT be zero in any other message.

 o  Next Payload (1 octet) - Indicates the type of payload that
    immediately follows the header. The format and value of each
    payload are defined below.

 o  Major Version (4 bits) - Indicates the major version of the IKE
    protocol in use. Implementations based on this version of IKE
    MUST set the Major Version to 2. Implementations based on
    previous versions of IKE and ISAKMP MUST set the Major Version
    to 1. Implementations based on this version of IKE MUST reject
    or ignore messages containing a version number greater than
    2.

 o  Minor Version (4 bits) - Indicates the minor version of the
    IKE protocol in use. Implementations based on this version of
    IKE MUST set the Minor Version to 0. They MUST ignore the
    minor version number of received messages.

 o  Exchange Type (1 octet) - Indicates the type of exchange being
    used. This constrains the payloads sent in each message and
    orderings of messages in an exchange.

                  Exchange Type             Value

                  RESERVED                  0-33
                  IKE_SA_INIT               34
                  IKE_AUTH                  35
```

```
                 CREATE_CHILD_SA          36
                 INFORMATIONAL            37
                 RESERVED TO IANA         38-239
                 Reserved for private use 240-255

   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'.

    --  X(reserved) (bits 0-2) - These bits MUST be cleared
        when sending and MUST be ignored on receipt.

    --  I(nitiator) (bit 3 of Flags) - This bit MUST be set in
        messages sent by the original initiator of the IKE_SA
        and MUST be cleared in messages sent by the original
        responder. It is used by the recipient to determine
     which eight octets of the SPI were generated by the
     recipient.

   -- 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.

   -- 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 and MUST be set in all responses.
     An IKE endpoint MUST NOT generate a response to a
     message that is marked as being a response.

   -- X(reserved) (bits 6-7 of Flags) - These bits MUST be
     cleared when sending and MUST be ignored on receipt.

   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.

   o Length (4 octets) - Length of total message (header + payloads)
   in octets.

  --------------------
```

Identifier: RQ_002_6232
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message on an established IKE Security Association, it MUST insert the non-zero Security Parameter Index (SPI) value (set by the initiator in the original IKE_SA_INIT request) into the IKE_SA Initiator's SPI field of the IKE Header

RFC Text:
The format of the IKE header is shown in Figure 4.

```
          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               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

 -- X(reserved) (bits 0-2) - These bits MUST be cleared
   when sending and MUST be ignored on receipt.

 -- I(nitiator) (bit 3 of Flags) - This bit MUST be set in
   messages sent by the original initiator of the IKE_SA
   and MUST be cleared in messages sent by the original
   responder. It is used by the recipient to determine
   which eight octets of the SPI were generated by the
   recipient.

 -- 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.

 -- 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 and MUST be set in all responses.
   An IKE endpoint MUST NOT generate a response to a
   message that is marked as being a response.

 -- X(reserved) (bits 6-7 of Flags) - These bits MUST be
   cleared when sending and MUST be ignored on receipt.

 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.

 o Length (4 octets) - Length of total message (header + payloads)
 in octets.

--------------------
```

Identifier: RQ_002_6233
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message on an established IKE Security Association, it MUST insert the non-zero Security Parameter Index (SPI) value (set by the responder in the original IKE_SA_INIT exchange) into the IKE_SA Responder's SPI field of the IKE Header

RFC Text:

The format of the IKE header is shown in Figure 4.

```
           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               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

 -- X(reserved) (bits 0-2) - These bits MUST be cleared
   when sending and MUST be ignored on receipt.

 -- I(nitiator) (bit 3 of Flags) - This bit MUST be set in
   messages sent by the original initiator of the IKE_SA
   and MUST be cleared in messages sent by the original
   responder. It is used by the recipient to determine
   which eight octets of the SPI were generated by the
   recipient.

 -- 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.

 -- 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 and MUST be set in all responses.
   An IKE endpoint MUST NOT generate a response to a
   message that is marked as being a response.

 -- X(reserved) (bits 6-7 of Flags) - These bits MUST be
   cleared when sending and MUST be ignored on receipt.

 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.

 o Length (4 octets) - Length of total message (header + payloads)
 in octets.

--------------------
```

Identifier: RQ_002_6234
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE_SA_INIT request, it MUST insert a value of zero into the IKE_SA Responder's SPI field of the IKE Header

RFC Text:
The format of the IKE header is shown in Figure 4.

```
          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              !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared
  when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in
  messages sent by the original initiator of the IKE_SA
  and MUST be cleared in messages sent by the original
  responder. It is used by the recipient to determine
  which eight octets of the SPI were generated by the
  recipient.

-- 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.

-- 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 and MUST be set in all responses.
  An IKE endpoint MUST NOT generate a response to a
  message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be
  cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads)
in octets.

--------------------
```

Identifier: RQ_002_6235
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message, it MUST insert a value into the Next Payload field of the IKE Header according to the following list of permitted values:

```
   Next Payload Type        Notation  Value
-----------------------------------------------------
   No Next Payload              0

 RESERVED                  1-32
   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
   RESERVED TO IANA             49-127
   PRIVATE USE               128-255
```

RFC Text:

The format of the IKE header is shown in Figure 4.

```
          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               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        Figure 4: IKE Header Format
```

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than

```
  2.

 o Minor Version (4 bits) - Indicates the minor version of the
  IKE protocol in use. Implementations based on this version of
  IKE MUST set the Minor Version to 0. They MUST ignore the
  minor version number of received messages.

 o Exchange Type (1 octet) - Indicates the type of exchange being
  used. This constrains the payloads sent in each message and
  orderings of messages in an exchange.

         Exchange Type      Value

         RESERVED         0-33
         IKE_SA_INIT       34
         IKE_AUTH         35
         CREATE_CHILD_SA     36
         INFORMATIONAL      37
         RESERVED TO IANA     38-239
         Reserved for private use 240-255

 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'.

 -- X(reserved) (bits 0-2) - These bits MUST be cleared
   when sending and MUST be ignored on receipt.

 -- I(nitiator) (bit 3 of Flags) - This bit MUST be set in
   messages sent by the original initiator of the IKE_SA
   and MUST be cleared in messages sent by the original
   responder. It is used by the recipient to determine
   which eight octets of the SPI were generated by the
   recipient.

 -- 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.

 -- 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 and MUST be set in all responses.
   An IKE endpoint MUST NOT generate a response to a
   message that is marked as being a response.

 -- X(reserved) (bits 6-7 of Flags) - These bits MUST be
   cleared when sending and MUST be ignored on receipt.

 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.

 o Length (4 octets) - Length of total message (header + payloads)
 in octets.

--------------------
```

Identifier: RQ_002_6236
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an implementation of the IKE protocol based upon RFC4306 sends an IKE message, it MUST insert a value of 2 into the Major Version field in the IKE Header

RFC Text:
The format of the IKE header is shown in Figure 4.

```
          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               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        Figure 4: IKE Header Format
```

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

 -- X(reserved) (bits 0-2) - These bits MUST be cleared
   when sending and MUST be ignored on receipt.

 -- I(nitiator) (bit 3 of Flags) - This bit MUST be set in
   messages sent by the original initiator of the IKE_SA
   and MUST be cleared in messages sent by the original
   responder. It is used by the recipient to determine
   which eight octets of the SPI were generated by the
   recipient.

 -- 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.

 -- 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 and MUST be set in all responses.
   An IKE endpoint MUST NOT generate a response to a
   message that is marked as being a response.

 -- X(reserved) (bits 6-7 of Flags) - These bits MUST be
   cleared when sending and MUST be ignored on receipt.

 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.

 o Length (4 octets) - Length of total message (header + payloads)
 in octets.

--------------------
```

Identifier: RQ_002_6237
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:

If an implementation of the IKE protocol based upon RFC4306 receives an IKE message with a value greater than 2 in the Major Version field of the IKE Header, it must either ignore it or reject it by sending an IKE Notify payload in its response with the Error Type set to INVALID_MAJOR_VERSION

RFC Text:

The format of the IKE header is shown in Figure 4.

```
          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               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. **Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.**

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

```
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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared
   when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in
   messages sent by the original initiator of the IKE_SA
   and MUST be cleared in messages sent by the original
   responder. It is used by the recipient to determine
   which eight octets of the SPI were generated by the
   recipient.

-- 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.

-- 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 and MUST be set in all responses.
   An IKE endpoint MUST NOT generate a response to a
   message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be
   cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads)
in octets.

--------------------
```

Identifier: RQ_002_6238
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an implementation of the IKE protocol based upon RFC4306 sends an IKE message, it MUST insert a value of 0 into the Minor Version field in the IKE Header

RFC Text:
The format of the IKE header is shown in Figure 4.

```
          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               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

```
 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'.

 -- X(reserved) (bits 0-2) - These bits MUST be cleared
   when sending and MUST be ignored on receipt.

 -- I(nitiator) (bit 3 of Flags) - This bit MUST be set in
   messages sent by the original initiator of the IKE_SA
   and MUST be cleared in messages sent by the original
   responder. It is used by the recipient to determine
   which eight octets of the SPI were generated by the
   recipient.

 -- 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.

 -- 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 and MUST be set in all responses.
   An IKE endpoint MUST NOT generate a response to a
   message that is marked as being a response.

 -- X(reserved) (bits 6-7 of Flags) - These bits MUST be
   cleared when sending and MUST be ignored on receipt.

 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.

 o Length (4 octets) - Length of total message (header + payloads)
 in octets.

--------------------
```

Identifier: RQ_002_6239
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:

When an implementation of the IKE protocol based upon RFC4306 receives an IKE message, it MUST ignore any value set in the Minor Version field of the IKE Header

RFC Text:

The format of the IKE header is shown in Figure 4.

```
          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               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        Figure 4: IKE Header Format
```

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. **They MUST ignore the minor version number of received messages**.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

```
        Exchange Type      Value

        RESERVED         0-33
        IKE_SA_INIT       34
        IKE_AUTH         35
        CREATE_CHILD_SA     36
        INFORMATIONAL      37
        RESERVED TO IANA     38-239
        Reserved for private use 240-255
```

```
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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared
   when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in
   messages sent by the original initiator of the IKE_SA
   and MUST be cleared in messages sent by the original
   responder. It is used by the recipient to determine
   which eight octets of the SPI were generated by the
   recipient.

-- 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.

-- 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 and MUST be set in all responses.
   An IKE endpoint MUST NOT generate a response to a
   message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be
   cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads)
in octets.

--------------------
```

Identifier: RQ_002_6240
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message, it MUST insert one of the following values into the Exchange Type field in the IKE Header to indicate the type of message exchange to which the message belongs:

```
Exchange Type       Value
------------------------------
RESERVED         0-33
IKE_SA_INIT        34
IKE_AUTH         35
CREATE_CHILD_SA      36
INFORMATIONAL       37
RESERVED TO IANA     38-239
Reserved for private use 240-255
```

RFC Text:

The format of the IKE header is shown in Figure 4.

```
          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               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type Value

```
        RESERVED         0-33
        IKE_SA_INIT       34
        IKE_AUTH         35
        CREATE_CHILD_SA     36
        INFORMATIONAL      37
        RESERVED TO IANA     38-239
        Reserved for private use 240-255
```

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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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.

-- 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 and MUST be set in all responses. An IKE endpoint MUST NOT generate a response to a message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6241
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message, it MUST clear bits 0, 1, 2, 6 and 7 of the Flags field in the IKE Header to zero (bit 0 is the least significant bit of the octet) as they are reserved for future use

RFC Text:

The format of the IKE header is shown in Figure 4.

```
          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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

```
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'.
```

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

```
 --  I(nitiator) (bit 3 of Flags) - This bit MUST be set in
     messages sent by the original initiator of the IKE_SA
     and MUST be cleared in messages sent by the original
     responder. It is used by the recipient to determine
     which eight octets of the SPI were generated by the
     recipient.

 --  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.

 --  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 and MUST be set in all responses.
     An IKE endpoint MUST NOT generate a response to a
     message that is marked as being a response.
```

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

```
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.

o Length (4 octets) - Length of total message (header + payloads)
in octets.
```

Identifier: RQ_002_6242
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message, it MUST ignore bits 0, 1, 2, 6 and 7 of the Flags field in the IKE Header (bit 0 is the least significant bit of the octet) as they are reserved for future use

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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.

-- 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 and MUST be set in all responses. An IKE endpoint MUST NOT generate a response to a message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6243
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message, it MUST clear bits 0, 1, 2, 6 and 7 of the Flags field in the IKE Header to zero (bit 0 is the least significant bit of the octet) as they are reserved for future use

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                 Figure 4:  IKE Header Format
```

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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.

-- 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 and MUST be set in all responses. An IKE endpoint MUST NOT generate a response to a message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6244
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message, it MUST set bit 3 in the Flags field in the IKE Header to 1 (bit 0 is the least significant bit of the octet) if the implementation initiated the establishment of the IKE_SA being used

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

- 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'.


 --  X(reserved) (bits 0-2) - These bits MUST be cleared
     when sending and MUST be ignored on receipt.


 --  I(nitiator) (bit 3 of Flags) - This bit MUST be set in
     messages sent by the original initiator of the IKE_SA
     and MUST be cleared in messages sent by the original
     responder. It is used by the recipient to determine
     which eight octets of the SPI were generated by the
     recipient.

 --  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.

 --  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 and MUST be set in all responses.
     An IKE endpoint MUST NOT generate a response to a
     message that is marked as being a response.

 --  X(reserved) (bits 6-7 of Flags) - These bits MUST be
     cleared when sending and MUST be ignored on receipt.


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.

o Length (4 octets) - Length of total message (header + payloads)
in octets.

--------------------
```

Identifier: RQ_002_6245
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message, it MUST clear bit 3 in the Flags field in the IKE Header to 0 (bit 0 is the least significant bit of the octet) if the implementation was the responder to the original IKE_SA establishment request

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA **and MUST be cleared in messages sent by the original responde**r. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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.

-- 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 and MUST be set in all responses. An IKE endpoint MUST NOT generate a response to a message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6246
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an implementation of the IKE protocol based upon RFC4306 sends an IKE message, it MUST clear bit 4 in the Flags field in the IKE Header to 0 (bit 0 is the least significant bit of the octet)

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

```
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'.

 --  X(reserved) (bits 0-2) - These bits MUST be cleared
     when sending and MUST be ignored on receipt.

 --  I(nitiator) (bit 3 of Flags) - This bit MUST be set in
     messages sent by the original initiator of the IKE_SA
     and MUST be cleared in messages sent by the original
     responder.  It is used by the recipient to determine
     which eight octets of the SPI were generated by the
     recipient.

 --  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.

 --  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 and MUST be set in all responses.
     An IKE endpoint MUST NOT generate a response to a
     message that is marked as being a response.

 --  X(reserved) (bits 6-7 of Flags) - These bits MUST be
     cleared when sending and MUST be ignored on receipt.

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.

o  Length (4 octets) - Length of total message (header + payloads)
in octets.

--------------------
```

Identifier: RQ_002_6247
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an implementation of the IKE protocol based upon RFC4306 receives an IKE message, it MUST ignore the state of bit 4 in the Flags field in the IKE Header

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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**.

-- 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 and MUST be set in all responses. An IKE endpoint MUST NOT generate a response to a message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6248
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE request message, it MUST clear bit 5 in the Flags field in the IKE Header to zero (bit 0 is the least significant bit of the octet)

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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.

-- 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** and MUST be set in all responses. An IKE endpoint MUST NOT generate a response to a message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6249
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE response message, it MUST set bit 5 in the Flags field in the IKE Header to one (bit 0 is the least significant bit of the octet)

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                 Figure 4:  IKE Header Format
```

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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.

-- 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 message**s and MUST be set in all responses**. An IKE endpoint MUST NOT generate a response to a message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6250
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
An IKE endpoint MUST NOT generate a response to a message that is marked as being a response (bit 5 set to one in the IKE Header of the received message).

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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.

-- 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 and MUST be set in all responses. **An IKE endpoint MUST NOT generate a response to a message that is marked as being a response**.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6251
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message, it MUST insert a number into the Message ID field of the IKE Header to uniquely identify the message

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

- 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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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.

-- 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 and MUST be set in all responses. An IKE endpoint MUST NOT generate a response to a message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6252
RFC Clause: 3.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message, it MUST insert the length of the message (header plus payloads) in octets into the Length field of the IKE Header

RFC Text:
The format of the IKE header is shown in Figure 4.

```
                     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                             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: IKE Header Format

- o Initiator's SPI (8 octets) - A value chosen by the initiator to identify a unique IKE security association. This value MUST NOT be zero.

- o Responder's SPI (8 octets) - A value chosen by the responder to identify a unique IKE security association. This value MUST be zero in the first message of an IKE Initial Exchange (including repeats of that message including a cookie) and MUST NOT be zero in any other message.

- o Next Payload (1 octet) - Indicates the type of payload that immediately follows the header. The format and value of each payload are defined below.

- o Major Version (4 bits) - Indicates the major version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Major Version to 2. Implementations based on previous versions of IKE and ISAKMP MUST set the Major Version to 1. Implementations based on this version of IKE MUST reject or ignore messages containing a version number greater than 2.

- o Minor Version (4 bits) - Indicates the minor version of the IKE protocol in use. Implementations based on this version of IKE MUST set the Minor Version to 0. They MUST ignore the minor version number of received messages.

- o Exchange Type (1 octet) - Indicates the type of exchange being used. This constrains the payloads sent in each message and orderings of messages in an exchange.

Exchange Type	Value
RESERVED	0-33
IKE_SA_INIT	34
IKE_AUTH	35
CREATE_CHILD_SA	36
INFORMATIONAL	37
RESERVED TO IANA	38-239
Reserved for private use	240-255

- 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'.

-- X(reserved) (bits 0-2) - These bits MUST be cleared when sending and MUST be ignored on receipt.

-- I(nitiator) (bit 3 of Flags) - This bit MUST be set in messages sent by the original initiator of the IKE_SA and MUST be cleared in messages sent by the original responder. It is used by the recipient to determine which eight octets of the SPI were generated by the recipient.

-- 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.

-- 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 and MUST be set in all responses. An IKE endpoint MUST NOT generate a response to a message that is marked as being a response.

-- X(reserved) (bits 6-7 of Flags) - These bits MUST be cleared when sending and MUST be ignored on receipt.

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.

o Length (4 octets) - Length of total message (header + payloads) in octets.

Identifier: RQ_002_6253
RFC Clause: 3.2
Type: Mandatory
Applies to: Host

Requirement:

When sending an IKE message, an IKE implementation MUST include a Generic Payload Header in the following format at the start of each Payload:

```
Octet                 Field
----------------------------
1                     Next Payload
2 (bit 0)             Critical flag
2 (bit1 to bit 7)     Reserved
3 and 4               Payload Length
```

RFC Text:

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.

```
                     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        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Generic Payload Header

The Generic Payload Header fields are defined as follows:

o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0. This field provides a "chaining" capability whereby additional payloads can be added to a message by appending it to the end of the message and setting the "Next Payload" field of the preceding payload to indicate the new payload's type. An Encrypted payload, which must always be the last payload of a message, is an exception. It contains data structures in the format of additional payloads. In the header of an Encrypted payload, the Next Payload field is set to the payload type of the first contained payload (instead of 0).

Payload Type Values

Next Payload Type	Notation	Value
No Next Payload		0
RESERVED		1-32
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
RESERVED TO IANA		49-127
PRIVATE USE		128-255

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.

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

o  RESERVED (7 bits) - MUST be sent as zero; MUST be ignored on
   receipt.

o  Payload Length (2 octets) - Length in octets of the current
   payload, including the generic payload header.

--------------------
```

Identifier: RQ_002_6254
RFC Clause: 3.2
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message, it MUST insert a value into the Next Payload field of the Generic Payload Header according to the following list of permitted values:

```
     Next Payload Type               Notation   Value
----------------------------------------------------
     No Next Payload

     RESERVED                                    1-32
     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
     RESERVED TO IANA                            49-127
     PRIVATE USE                                 128-255
```

RFC Text:

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.

```
                      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        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Generic Payload Header

The Generic Payload Header fields are defined as follows:

- **Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0. This field provides a "chaining" capability whereby additional payloads can be added to a message by appending it to the end of the message and setting the "Next Payload" field of the preceding payload to indicate the new payload's type. An Encrypted payload, which must always be the last payload of a message, is an exception. It contains data structures in the format of additional payloads. In the header of an Encrypted payload, the Next Payload field is set to the payload type of the first contained payload (instead of 0).**

Payload Type Values

```
     Next Payload Type               Notation  Value

     No Next Payload                             0

     RESERVED                                   1-32
     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**
RESERVED TO IANA		**49-127**
PRIVATE USE		**128-255**

Payload type values 1-32 should not be used o 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.

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

o RESERVED (7 bits) - MUST be sent as zero; MUST be ignored on receipt.

o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

Identifier: RQ_002_6255
RFC Clause: 3.2
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message with the Critical flag set to zero in the Generic Payload Header, it MUST ignore the payload if it does not support the payload type indicated in the previous Next Payload field

RFC Text:
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.

```
                     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        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Generic Payload Header

The Generic Payload Header fields are defined as follows:

o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0. This field provides a "chaining" capability whereby additional payloads can be added to a message by appending it to the end of the message and setting the "Next Payload" field of the preceding payload to indicate the new payload's type. An Encrypted payload, which must always be the last payload of a message, is an exception. It contains data structures in the format of additional payloads. In the header of an Encrypted payload, the Next Payload field is set to the payload type of the first contained payload (instead of 0).

Payload Type Values

Next Payload Type	Notation	Value
No Next Payload		0
RESERVED		1-32
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
RESERVED TO IANA		49-127
PRIVATE USE		128-255

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.

o Critical (1 bit) **- MUST be set to zero if the sender wants the recipient to skip this payload if it does not understand the payload type code in the Next Payload field of the previous payload.** MUST be set to one if the sender wants the recipient to reject this entire message if it does not understand the payload type. MUST be ignored by the recipient if the recipient understands the payload type code. MUST be set to zero for payload types defined in this document. Note that the critical

```
   bit applies to the current payload rather than the "next" payload
   whose type code appears in the first octet. 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.

o  RESERVED (7 bits) - MUST be sent as zero; MUST be ignored on
   receipt.

o  Payload Length (2 octets) - Length in octets of the current
   payload, including the generic payload header.

--------------------
```

Identifier: RQ_002_6256
RFC Clause: 3.2
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message with the Critical flag in the Generic Payload Header set to one, it MUST reject the entire message by sending an IKE Notify payload with the Error Type set to UNSUPPORTED_CRITICAL_PAYLOAD if it does not support the payload type indicated in the previous Next Payload field

RFC Text:
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.

```
                     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        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Generic Payload Header

The Generic Payload Header fields are defined as follows:

o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0. This field provides a "chaining" capability whereby additional payloads can be added to a message by appending it to the end of the message and setting the "Next Payload" field of the preceding payload to indicate the new payload's type. An Encrypted payload, which must always be the last payload of a message, is an exception. It contains data structures in the format of additional payloads. In the header of an Encrypted payload, the Next Payload field is set to the payload type of the first contained payload (instead of 0).

Payload Type Values

Next Payload Type	Notation	Value
No Next Payload		0
RESERVED		1-32
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
RESERVED TO IANA		49-127
PRIVATE USE		128-255

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.

o Critical (1 bit) - MUST be set to zero if the sender wants the recipient to skip this payload if it does not understand the payload type code in the Next Payload field of the previous payload. **MUST be set to one if the sender wants the recipient to reject this entire message if it does not understand the payload type**. MUST be ignored by the recipient if the recipient understands the payload type code. MUST be set to zero for

```
   payload types defined in this document. Note that the critical
   bit applies to the current payload rather than the "next" payload
   whose type code appears in the first octet. 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.

o  RESERVED (7 bits) - MUST be sent as zero; MUST be ignored on
   receipt.

o  Payload Length (2 octets) - Length in octets of the current
   payload, including the generic payload header.

--------------------
```

Identifier: RQ_002_6257
RFC Clause: 3.2
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message it MUST ignore the state of the Critical flag in the Generic Payload Header if it supports the payload type indicated in the previous Next Payload field

RFC Text:
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.

```
                     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        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                  Figure 5:  Generic Payload Header
```

The Generic Payload Header fields are defined as follows:

o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0. This field provides a "chaining" capability whereby additional payloads can be added to a message by appending it to the end of the message and setting the "Next Payload" field of the preceding payload to indicate the new payload's type. An Encrypted payload, which must always be the last payload of a message, is an exception. It contains data structures in the format of additional payloads. In the header of an Encrypted payload, the Next Payload field is set to the payload type of the first contained payload (instead of 0).

Payload Type Values

Next Payload Type	Notation	Value
No Next Payload		0
RESERVED		1-32
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
RESERVED TO IANA		49-127
PRIVATE USE		128-255

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.

o Critical (1 bit) - MUST be set to zero if the sender wants the recipient to skip this payload if it does not understand the payload type code in the Next Payload field of the previous payload. MUST be set to one if the sender wants the recipient to reject this entire message if it does not understand the payload type. **MUST be ignored by the recipient if the recipient understands the payload type code**. MUST be set to zero for payload types defined in this document. Note that the critical

```
   bit applies to the current payload rather than the "next" payload
   whose type code appears in the first octet. 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.

o  RESERVED (7 bits) - MUST be sent as zero; MUST be ignored on
   receipt.

o  Payload Length (2 octets) - Length in octets of the current
   payload, including the generic payload header.

--------------------
```

Identifier: RQ_002_6258
RFC Clause: 3.2.
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message it MUST set the Critical flag in the Generic Payload Header to zero for if the payload type is one of the following:

```
Security Association
Key Exchange
Identification - Initiator
Identification - Responder
Certificate
Certificate Request
Authentication
Nonce
Notify
Delete
Vendor ID
Traffic Selector - Initiator
Traffic Selector - Responder
Encrypted
Configuration
Extensible Authentication
```

RFC Text:

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.

```
                     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        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                    Figure 5:  Generic Payload Header
```

The Generic Payload Header fields are defined as follows:

```
o  Next Payload (1 octet) - Identifier for the payload type of the
   next payload in the message. If the current payload is the last
   in the message, then this field will be 0. This field provides a
   "chaining" capability whereby additional payloads can be added to
   a message by appending it to the end of the message and setting
   the "Next Payload" field of the preceding payload to indicate the
   new payload's type. An Encrypted payload, which must always be
   the last payload of a message, is an exception. It contains data
   structures in the format of additional payloads. In the header of
   an Encrypted payload, the Next Payload field is set to the payload
   type of the first contained payload (instead of 0).

   Payload Type Values

     Next Payload Type               Notation  Value

     No Next Payload                              0

     RESERVED                                   1-32
     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
```

```
   RESERVED TO IANA                     49-127
   PRIVATE USE                         128-255
```

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.

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

o RESERVED (7 bits) - MUST be sent as zero; MUST be ignored on receipt.

o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

Identifier: RQ_002_6259
RFC Clause: 3.2.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message it MUST set the Payload Length field in the Generic Payload Header to the length in octets of the current payload, including the Generic Payload Header itself.

RFC Text:
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.

```
                     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        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Generic Payload Header

The Generic Payload Header fields are defined as follows:

o Next Payload (1 octet) - Identifier for the payload type of the next payload in the message. If the current payload is the last in the message, then this field will be 0. This field provides a "chaining" capability whereby additional payloads can be added to a message by appending it to the end of the message and setting the "Next Payload" field of the preceding payload to indicate the new payload's type. An Encrypted payload, which must always be the last payload of a message, is an exception. It contains data structures in the format of additional payloads. In the header of an Encrypted payload, the Next Payload field is set to the payload type of the first contained payload (instead of 0).

Payload Type Values

Next Payload Type	Notation	Value
No Next Payload		0
RESERVED		1-32
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
RESERVED TO IANA		49-127
PRIVATE USE		128-255

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.

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

o RESERVED (7 bits) - MUST be sent as zero; MUST be ignored on receipt.

o Payload Length (2 octets) - Length in octets of the current payload, including the generic payload header.

Identifier: RQ_002_6260
RFC Clause: 3.3.
Type: Optional
Applies to: Host

Requirement:
A Security Association payload MAY contain multiple proposals

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

Identifier: RQ_002_6261
RFC Clause: 3.3.
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends a Security Association payload containing more than one proposal, it MUST order the proposals within the payload from most preferred to least preferred.

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

Identifier: RQ_002_6262
RFC Clause: 3.3.
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST assemble Security Association Payloads with great peace of mind!

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

Identifier: RQ_002_6263
RFC Clause: 3.3.
Type: Mandatory
Applies to: Host

Requirement:
A Security Association payload in an IKE packet MUST comprise the Generic Payload Header followed by one or more Proposal substructures

RFC Text:

```
                     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>                          ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Security Association Payload

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

The payload type for the Security Association Payload is thirty three (33).

Identifier: RQ_002_6264
RFC Clause: 3.3.
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing a Security Association Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Security Association Payload) to the value thirty-three (33)

RFC Text:

```
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>                          ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 6:  Security Association Payload

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

The payload type for the Security Association Payload is thirty three (33).

Identifier: RQ_002_6265
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:
A Proposal Substructure in an IKE Security Association Payload MUST be constructed in the following format:

```
Octet                   Field
-----------------------------
1                       Continuation indicator
2                       Reserved
3 & 4                   Proposal Length
5                       Proposal number
6                       Protocol Identifier
7                       SPI Size
8                       Number of Transforms included in the proposal
9 to (SPI Size + 8)     SPI
(8 + SPI Size) to End   Transforms
```

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        Figure 7:  Proposal Substructure
```

o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

o RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). 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.

o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

Protocol	Protocol ID
RESERVED	0
IKE	1
AH	2
ESP	3
RESERVED TO IANA	4-200
PRIVATE USE	201-255

o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

```
 o  # of Transforms (1 octet) - Specifies the number of transforms
    in this proposal.

 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.

 o  Transforms (variable) - One or more transform substructures.


--------------------
```

Identifier: RQ_002_6266
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:

When sending a Security Association Payload containing more than one Proposal Substructure, an IKE implementation MUST set the Continuation Indicator (octet 1) in all but the last Proposal Substructure in the payload to the value two (2)

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Proposal Substructure

- **0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.**

- RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

- Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

- Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). 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.
- Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

Protocol	Protocol ID
RESERVED	0
IKE	1
AH	2
ESP	3
RESERVED TO IANA	4-200
PRIVATE USE	201-255

- SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

- # of Transforms (1 octet) - Specifies the number of transforms in this proposal.

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.

o Transforms (variable) - One or more transform substructures.

Identifier: RQ_002_6267
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Security Association Payload containing one or more Proposal Substructure, an IKE implementation MUST set the Continuation Indicator (octet 1) in the last Proposal Substructure in the payload to the value zero (0)

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Proposal Substructure

o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

o RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). 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.

o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

```
Protocol                Protocol ID
RESERVED                 0
IKE                      1
AH                       2
ESP                      3
RESERVED TO IANA         4-200
PRIVATE USE              201-255
```

o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

o # of Transforms (1 octet) - Specifies the number of transforms in this proposal.

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.

o Transforms (variable) - One or more transform substructures.

Identifier: RQ_002_6268
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:

When sending a Security Association Payload containing one or more Proposal Substructure, an IKE implementation MUST set the Proposal Length field in each Proposal Substructure to the length of the substructure in octets.

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Proposal Substructure

o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

o RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). 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.

o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

Protocol	Protocol ID
RESERVED	0
IKE	1
AH	2
ESP	3
RESERVED TO IANA	4-200
PRIVATE USE	201-255

o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

o # of Transforms (1 octet) - Specifies the number of transforms in this proposal.

```
 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.

 o  Transforms (variable) - One or more transform substructures.
```

Identifier: RQ_002_6269
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Security Association Payload containing one or more Proposal Substructure, an IKE implementation MUST set the Proposal Number field in the any Proposal Substructure to the same value as in the previous substructure if it is part of the same proposal.

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 7:  Proposal Substructure

 o  0 (last) or 2 (more) (1 octet) - Specifies whether this is the
    last Proposal Substructure in the SA. 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.

 o  RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on
    receipt.

 o  Proposal Length (2 octets) - Length of this proposal, including
    all transforms and attributes that follow.

 o  Proposal # (1 octet) - When a proposal is made, the first
    proposal in an SA payload MUST be #1, 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). 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.
 o  Protocol ID (1 octet) - Specifies the IPsec protocol identifier
    for the current negotiation. The defined values are:

     Protocol               Protocol ID
     RESERVED                0
     IKE                     1
     AH                      2
     ESP                     3
     RESERVED TO IANA        4-200
     PRIVATE USE             201-255

 o  SPI Size (1 octet) - For an initial IKE_SA negotiation, this
    field MUST be zero; the SPI is obtained from the outer header.
    During subsequent negotiations, it is equal to the size, in
    octets, of the SPI of the corresponding protocol (8 for IKE, 4
    for ESP and AH).
```

```
 o  # of Transforms (1 octet) - Specifies the number of transforms
    in this proposal.

 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.

 o  Transforms (variable) - One or more transform substructures.


--------------------
```

Identifier: RQ_002_6270
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Security Association Payload containing one or more Proposal Substructure, an IKE implementation MUST set the Proposal Number field in the second and subsequent Proposal Substructures to one more than the value in the previous substructure if it is not part of the same proposal.

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 7:  Proposal Substructure
```

o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

o RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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).** 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.

o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

Protocol	Protocol ID
RESERVED	0
IKE	1
AH	2
ESP	3
RESERVED TO IANA	4-200
PRIVATE USE	201-255

o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

o # of Transforms (1 octet) - Specifies the number of transforms in this proposal.

```
 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.

 o  Transforms (variable) - One or more transform substructures.


--------------------
```

Identifier: RQ_002_6271
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:

When responding to a Security Association Payload containing one or more Proposal Substructure, an IKE implementation MUST accept no more than one proposal by including in the Security Association Payload response those Proposal Substructures that form the overall accepted proposal (i.e., all those with the same Proposal Number)

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 7:  Proposal Substructure
```

o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

o RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). **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.**

o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

```
 Protocol               Protocol ID
 RESERVED                0
 IKE                     1
 AH                      2
 ESP                     3
 RESERVED TO IANA        4-200
 PRIVATE USE             201-255
```

o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

o # of Transforms (1 octet) - Specifies the number of transforms in this proposal.

```
 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.

 o  Transforms (variable) - One or more transform substructures.
```

Identifier: RQ_002_6272
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:

When sending a Security Association Payload containing one or more Proposal Substructure, an IKE implementation MUST set the Protocol ID field in each Proposal Substructure to one of the security protocols that the initiator is able to support according to the following set of defined values:

```
Protocol              Protocol ID
---------------------------------
Reserved              0
IKE                   1
AH                    2
ESP                   3
Reserved to IANA      4 to 200
Private Use           201 to 255
```

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 7:  Proposal Substructure
```

o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

o RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). 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.

o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

Protocol	**Protocol ID**
RESERVED	**0**
IKE	**1**
AH	**2**
ESP	**3**
RESERVED TO IANA	**4-200**
PRIVATE USE	**201-255**

o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

```
 o  # of Transforms (1 octet) - Specifies the number of transforms
    in this proposal.

 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.

 o  Transforms (variable) - One or more transform substructures.


--------------------
```

Identifier: RQ_002_6273
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Security Association Payload containing one or more Proposal Substructure within an initial IKE_SA, exchange, an IKE implementation MUST set the SPI Size field to zero (0) in each Proposal Substructure

RFC Text:

```
                     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>                           ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 7:  Proposal Substructure

 o  0 (last) or 2 (more) (1 octet) - Specifies whether this is the
    last Proposal Substructure in the SA. 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.

 o  RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on
    receipt.

 o  Proposal Length (2 octets) - Length of this proposal, including
    all transforms and attributes that follow.

 o  Proposal # (1 octet) - When a proposal is made, the first
    proposal in an SA payload MUST be #1, 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). 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.

 o  Protocol ID (1 octet) - Specifies the IPsec protocol identifier
    for the current negotiation.  The defined values are:

     Protocol              Protocol ID
     RESERVED               0
     IKE                    1
     AH                     2
     ESP                    3
     RESERVED TO IANA       4-200
     PRIVATE USE            201-255
```

o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header.

```
   During subsequent negotiations, it is equal to the size, in
   octets, of the SPI of the corresponding protocol (8 for IKE, 4
   for ESP and AH).

o  # of Transforms (1 octet) - Specifies the number of transforms
   in this proposal.

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.

o  Transforms (variable) - One or more transform substructures.
```

Identifier: RQ_002_6274
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:

When sending a Security Association Payload containing one or more Proposal Substructure subsequent to the initial IKE_SA exchange, an IKE implementation MUST set the SPI Size field in each Proposal Substructure to the length in octets of the SPI of the corresponding protocol

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Proposal Substructure

- 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

- RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

- Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

- Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). 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.

- Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

Protocol	Protocol ID
RESERVED	0
IKE	1
AH	2
ESP	3
RESERVED TO IANA	4-200
PRIVATE USE	201-255

- SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. **During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).**

```
 o  # of Transforms (1 octet) - Specifies the number of transforms
    in this proposal.

 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.

 o  Transforms (variable) - One or more transform substructures.


--------------------
```

Identifier: RQ_002_6275
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Security Association Payload containing one or more Proposal Substructure, an IKE implementation MUST set the Number of Transforms field in each Proposal Substructure to the number of Transform Substructures included in the Proposal Substructure

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 7:  Proposal Substructure
```

o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

o RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). 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.

o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

Protocol	Protocol ID
RESERVED	0
IKE	1
AH	2
ESP	3
RESERVED TO IANA	4-200
PRIVATE USE	201-255

o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

```
 o  # of Transforms (1 octet) - Specifies the number of transforms
    in this proposal.

 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.

 o  Transforms (variable) - One or more transform substructures.


--------------------
```

Identifier: RQ_002_6276
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Security Association Payload containing one or more Proposal Substructure, an IKE implementation MUST set the SPI field in each Proposal Substructure to it Security Parameter Index value

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Proposal Substructure

- o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

- o RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

- o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

- o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). 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.

- o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

Protocol	Protocol ID
RESERVED	0
IKE	1
AH	2
ESP	3
RESERVED TO IANA	4-200
PRIVATE USE	201-255

- o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

- o # of Transforms (1 octet) - Specifies the number of transforms in this proposal.

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.

o Transforms (variable) - One or more transform substructures.

Identifier: RQ_002_6277
RFC Clause: 3.3.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Security Association Payload containing one or more Proposal Substructure, an IKE implementation MUST include one or more Transform Substructures in each Proposal Substructure

RFC Text:

```
                     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>                           ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Proposal Substructure

o 0 (last) or 2 (more) (1 octet) - Specifies whether this is the last Proposal Substructure in the SA. 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.

o RESERVED (1 octet) - MUST be sent as zero; MUST be ignored on receipt.

o Proposal Length (2 octets) - Length of this proposal, including all transforms and attributes that follow.

o Proposal # (1 octet) - When a proposal is made, the first proposal in an SA payload MUST be #1, 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). 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.

o Protocol ID (1 octet) - Specifies the IPsec protocol identifier for the current negotiation. The defined values are:

Protocol	Protocol ID
RESERVED	0
IKE	1
AH	2
ESP	3
RESERVED TO IANA	4-200
PRIVATE USE	201-255

o SPI Size (1 octet) - For an initial IKE_SA negotiation, this field MUST be zero; the SPI is obtained from the outer header. During subsequent negotiations, it is equal to the size, in octets, of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH).

```
 o  # of Transforms (1 octet) - Specifies the number of transforms
    in this proposal.

 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.
```

o Transforms (variable) - One or more transform substructures.

Identifier: RQ_002_6278
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:
A Transform Substructure within a Proposal Substructure of an IKE Security Association Payload MUST be constructed in the following format:

```
Octet              Field
-------------------------
1                  Continuation Indicator
2                  Reserved
3 & 4              Transform Length
5                  Transform Type
6                  Reserved
7 & 8              Transform ID
9 to End           Transform Attributes
```

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 8:  Transform Substructure
```

o 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

o Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.

o Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.

o Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

```
                                Transform    Used In
                                   Type
     RESERVED                        0
     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)
     RESERVED TO IANA                6-240
     PRIVATE USE                     241-255
```

For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

```
     Name                     Number           Defined In
```

```
     RESERVED                0
     ENCR_DES_IV64           1               (RFC1827)
     ENCR_DES                2               (RFC2405), [DES]
     ENCR_3DES               3               (RFC2451)
     ENCR_RC5                4               (RFC2451)
     ENCR_IDEA               5               (RFC2451), [IDEA]
     ENCR_CAST               6               (RFC2451)
     ENCR_BLOWFISH           7               (RFC2451)
     ENCR_3IDEA              8               (RFC2451)
     ENCR_DES_IV32           9
     RESERVED               10
     ENCR_NULL              11               (RFC2410)
     ENCR_AES_CBC           12               (RFC3602)
     ENCR_AES_CTR           13               (RFC3664)

     values 14-1023 are reserved to IANA.  Values 1024-65535 are
     for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

     Name                     Number                 Defined In
     RESERVED                    0
     PRF_HMAC_MD5                1                   (RFC2104), [MD5]
     PRF_HMAC_SHA1               2                   (RFC2104), [SHA]
     PRF_HMAC_TIGER              3                   (RFC2104)
     PRF_AES128_XCBC             4                   (RFC3664)

     values 5-1023 are reserved to IANA.  Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

     Name                     Number                   Defined In
     NONE                       0
     AUTH_HMAC_MD5_96           1                       (RFC2403)
     AUTH_HMAC_SHA1_96          2                       (RFC2404)
     AUTH_DES_MAC               3
     AUTH_KPDK_MD5              4                       (RFC1826)
     AUTH_AES_XCBC_96           5                       (RFC3566)

     values 6-1023 are reserved to IANA.  Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

     Name                                Number
     NONE                               0
     Defined in Appendix B              1 - 2
     RESERVED                           3 - 4
     Defined in [ADDGROUP]              5
     RESERVED TO IANA                   6 - 13
     Defined in [ADDGROUP]              14 - 18
     RESERVED TO IANA                   19 - 1023
     PRIVATE USE                        1024-65535

For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

     Name                                Number
     No Extended Sequence Numbers       0
     Extended Sequence Numbers          1
     RESERVED                           2 - 65535


--------------------
```

Identifier: RQ_002_6279
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:

When sending a Proposal Substructure containing more than one Transform Substructure within a Security Association Payload, an IKE implementation MUST set the Continuation Indicator (octet 1) in all but the last Transform Substructure in the proposal Substructure to the value three (3)

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 8:  Transform Substructure
```

- **0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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**.

- RESERVED - MUST be sent as zero; MUST be ignored on receipt.

- Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.

- Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.

- Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

```
                               Transform    Used In
                                  Type
     RESERVED                      0
     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)
     RESERVED TO IANA                6-240
     PRIVATE USE                     241-255
```

For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

```
     Name                     Number           Defined In
     RESERVED                    0
     ENCR_DES_IV64               1              (RFC1827)
     ENCR_DES                    2              (RFC2405), [DES]
     ENCR_3DES                   3              (RFC2451)
     ENCR_RC5                    4              (RFC2451)
     ENCR_IDEA                   5              (RFC2451), [IDEA]
     ENCR_CAST                   6              (RFC2451)
     ENCR_BLOWFISH               7              (RFC2451)
     ENCR_3IDEA                  8              (RFC2451)
```

```
     ENCR_DES_IV32              9
     RESERVED                  10
     ENCR_NULL                 11              (RFC2410)
     ENCR_AES_CBC              12              (RFC3602)
     ENCR_AES_CTR              13              (RFC3664)

     values 14-1023 are reserved to IANA.  Values 1024-65535 are
     for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

     Name                     Number                 Defined In
     RESERVED                    0
     PRF_HMAC_MD5                1                   (RFC2104), [MD5]
     PRF_HMAC_SHA1               2                   (RFC2104), [SHA]
     PRF_HMAC_TIGER              3                   (RFC2104)
     PRF_AES128_XCBC             4                   (RFC3664)

     values 5-1023 are reserved to IANA.  Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

     Name                     Number                 Defined In
     NONE                       0
     AUTH_HMAC_MD5_96           1                     (RFC2403)
     AUTH_HMAC_SHA1_96          2                     (RFC2404)
     AUTH_DES_MAC               3
     AUTH_KPDK_MD5              4                     (RFC1826)
     AUTH_AES_XCBC_96           5                     (RFC3566)

     values 6-1023 are reserved to IANA.  Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

     Name                                Number
     NONE                               0
     Defined in Appendix B              1 - 2
     RESERVED                           3 - 4
     Defined in [ADDGROUP]              5
     RESERVED TO IANA                   6 - 13
     Defined in [ADDGROUP]              14 - 18
     RESERVED TO IANA                   19 - 1023
     PRIVATE USE                        1024-65535
For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

     Name                                Number
     No Extended Sequence Numbers       0
     Extended Sequence Numbers          1
     RESERVED                           2 - 65535


--------------------
```

Identifier: RQ_002_6280
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:

When sending a Proposal Substructure containing one or more Transform Substructure within a Security Association Payload, an IKE implementation MUST set the Continuation Indicator (octet 1) in the last Transform Substructure in the proposal Substructure to the value zero (0)

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 8:  Transform Substructure
```

- **0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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.**

- RESERVED - MUST be sent as zero; MUST be ignored on receipt.

- Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.

- Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.

- Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

	Transform Type	Used In
RESERVED	0	
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)
RESERVED TO IANA	6-240	
PRIVATE USE	241-255	

For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

Name	Number	Defined In
RESERVED	0	
ENCR_DES_IV64	1	(RFC1827)
ENCR_DES	2	(RFC2405), [DES]
ENCR_3DES	3	(RFC2451)
ENCR_RC5	4	(RFC2451)
ENCR_IDEA	5	(RFC2451), [IDEA]
ENCR_CAST	6	(RFC2451)
ENCR_BLOWFISH	7	(RFC2451)
ENCR_3IDEA	8	(RFC2451)

```
     ENCR_DES_IV32              9
     RESERVED                  10
     ENCR_NULL                 11              (RFC2410)
     ENCR_AES_CBC              12              (RFC3602)
     ENCR_AES_CTR              13              (RFC3664)

     values 14-1023 are reserved to IANA. Values 1024-65535 are
     for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

     Name                     Number              Defined In
     RESERVED                   0
     PRF_HMAC_MD5               1                 (RFC2104), [MD5]
     PRF_HMAC_SHA1              2                 (RFC2104), [SHA]
     PRF_HMAC_TIGER             3                 (RFC2104)
     PRF_AES128_XCBC            4                 (RFC3664)

     values 5-1023 are reserved to IANA. Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

     Name                     Number                Defined In
     NONE                      0
     AUTH_HMAC_MD5_96          1                     (RFC2403)
     AUTH_HMAC_SHA1_96         2                     (RFC2404)
     AUTH_DES_MAC              3
     AUTH_KPDK_MD5             4                     (RFC1826)
     AUTH_AES_XCBC_96          5                     (RFC3566)

     values 6-1023 are reserved to IANA. Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

     Name                                Number
     NONE                               0
     Defined in Appendix B              1 - 2
     RESERVED                           3 - 4
     Defined in [ADDGROUP]              5
     RESERVED TO IANA                   6 - 13
     Defined in [ADDGROUP]              14 - 18
     RESERVED TO IANA                   19 - 1023
     PRIVATE USE                        1024-65535

For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

     Name                                Number
     No Extended Sequence Numbers       0
     Extended Sequence Numbers          1
     RESERVED                           2 - 65535


--------------------
```

Identifier: RQ_002_6281
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:
When sending a Proposal Substructure containing one or more Transform Substructure within a Security Association Payload, an IKE implementation MUST set the Transform Length field in each Transform Substructure to the length of the substructure in octets

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Transform Substructure

- 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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.

- RESERVED - MUST be sent as zero; MUST be ignored on receipt.

- **Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes**.

- Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.

- Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

	Transform Type	Used In
RESERVED	0	
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)
RESERVED TO IANA	6-240	
PRIVATE USE	241-255	

For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

Name	Number	Defined In
RESERVED	0	
ENCR_DES_IV64	1	(RFC1827)
ENCR_DES	2	(RFC2405), [DES]
ENCR_3DES	3	(RFC2451)
ENCR_RC5	4	(RFC2451)
ENCR_IDEA	5	(RFC2451), [IDEA]
ENCR_CAST	6	(RFC2451)
ENCR_BLOWFISH	7	(RFC2451)

```
     ENCR_3IDEA                8             (RFC2451)
     ENCR_DES_IV32             9
     RESERVED                 10
     ENCR_NULL                11             (RFC2410)
     ENCR_AES_CBC             12             (RFC3602)
     ENCR_AES_CTR             13             (RFC3664)

     values 14-1023 are reserved to IANA.  Values 1024-65535 are
     for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

     Name                    Number              Defined In
     RESERVED                  0
     PRF_HMAC_MD5              1                 (RFC2104), [MD5]
     PRF_HMAC_SHA1             2                 (RFC2104), [SHA]
     PRF_HMAC_TIGER            3                 (RFC2104)
     PRF_AES128_XCBC           4                 (RFC3664)

     values 5-1023 are reserved to IANA.  Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

     Name                    Number                Defined In
     NONE                     0
     AUTH_HMAC_MD5_96         1                     (RFC2403)
     AUTH_HMAC_SHA1_96        2                     (RFC2404)
     AUTH_DES_MAC             3
     AUTH_KPDK_MD5            4                     (RFC1826)
     AUTH_AES_XCBC_96         5                     (RFC3566)

     values 6-1023 are reserved to IANA.  Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

     Name                               Number
     NONE                              0
     Defined in Appendix B             1 - 2
     RESERVED                          3 - 4
     Defined in [ADDGROUP]             5
     RESERVED TO IANA                  6 - 13
     Defined in [ADDGROUP]             14 - 18
     RESERVED TO IANA                  19 - 1023
     PRIVATE USE                       1024-65535

For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

     Name                               Number
     No Extended Sequence Numbers      0
     Extended Sequence Numbers         1
     RESERVED                          2 - 65535


--------------------
```

Identifier: RQ_002_6282
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:

When sending a Proposal Substructure containing one or more Transform Substructure within a Security Association Payload, an IKE implementation MUST set the Transform Type field in each Transform Substructure to one of the following values according to the type of transform being specified in the substructure:

```
 Transform Type               Value
 ---------------------------------
 Reserved                     0
 Encryption Algorithm         1
 Pseudo-Random Function       2
 Integrity Algorithm          3
 Diffie-Hellman Group         4
 Extended Sequence Numbers    5
 Reserved  to IANA            6 to 240
 Private Use                  241 to 255
```

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 8:  Transform Substructure
```

o 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

o Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.

o Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.

o Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

	Transform Type	**Used In**
RESERVED	**0**	
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)**
RESERVED TO IANA	**6-240**	
PRIVATE USE	**241-255**	

For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

Name	Number	Defined In
RESERVED	0	
ENCR_DES_IV64	1	(RFC1827)
ENCR_DES	2	(RFC2405), [DES]
ENCR_3DES	3	(RFC2451)
ENCR_RC5	4	(RFC2451)
ENCR_IDEA	5	(RFC2451), [IDEA]
ENCR_CAST	6	(RFC2451)
ENCR_BLOWFISH	7	(RFC2451)
ENCR_3IDEA	8	(RFC2451)
ENCR_DES_IV32	9	
RESERVED	10	
ENCR_NULL	11	(RFC2410)
ENCR_AES_CBC	12	(RFC3602)
ENCR_AES_CTR	13	(RFC3664)

values 14-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

Name	Number	Defined In
RESERVED	0	
PRF_HMAC_MD5	1	(RFC2104), [MD5]
PRF_HMAC_SHA1	2	(RFC2104), [SHA]
PRF_HMAC_TIGER	3	(RFC2104)
PRF_AES128_XCBC	4	(RFC3664)

values 5-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

Name	Number	Defined In
NONE	0	
AUTH_HMAC_MD5_96	1	(RFC2403)
AUTH_HMAC_SHA1_96	2	(RFC2404)
AUTH_DES_MAC	3	
AUTH_KPDK_MD5	4	(RFC1826)
AUTH_AES_XCBC_96	5	(RFC3566)

values 6-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

Name	Number
NONE	0
Defined in Appendix B	1 - 2
RESERVED	3 - 4
Defined in [ADDGROUP]	5
RESERVED TO IANA	6 - 13
Defined in [ADDGROUP]	14 - 18
RESERVED TO IANA	19 - 1023
PRIVATE USE	1024-65535

For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

Name	Number
No Extended Sequence Numbers	0
Extended Sequence Numbers	1
RESERVED	2 - 65535

Identifier: RQ_002_6283
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:

When sending a Proposal Substructure containing one or more Transform Substructure within a Security Association Payload, an IKE implementation MUST set the Transform ID field in each Transform Substructure to one of the following values if the Transform Type in the same substructure is set to 1 - Encryption Algorithm (ENCR):

```
Name                    Value
-----------------------------
Reserved                0
ENCR_DES_IV64           1
ENCR_DES                2
ENCR_3DES               3
ENCR_RC5                4
ENCR_IDEA               5
ENCR_CAST               6
ENCR_BLOWFISH           7
ENCR_3IDEA              8
ENCR_DES_IV32           9
Reserved                10
ENCR_NULL               11
ENCR_AES_CBC            12
ENCR_AES_CTR            13
Reserved to IANA        14 to 1023
Private Use             1024 to 65535
```

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 8:  Transform Substructure
```

- o 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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.
- o RESERVED - MUST be sent as zero; MUST be ignored on receipt.
- o Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.
- o Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.
- o Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

```
                                  Transform    Used In
                                     Type
     RESERVED                         0
```

```
     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)
     RESERVED TO IANA                6-240
     PRIVATE USE                     241-255
```

For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

Name	Number	Defined In
RESERVED	**0**	
ENCR_DES_IV64	**1**	**(RFC1827)**
ENCR_DES	**2**	**(RFC2405), [DES]**
ENCR_3DES	**3**	**(RFC2451)**
ENCR_RC5	**4**	**(RFC2451)**
ENCR_IDEA	**5**	**(RFC2451), [IDEA]**
ENCR_CAST	**6**	**(RFC2451)**
ENCR_BLOWFISH	**7**	**(RFC2451)**
ENCR_3IDEA	**8**	**(RFC2451)**
ENCR_DES_IV32	**9**	
RESERVED	**10**	
ENCR_NULL	**11**	**(RFC2410)**
ENCR_AES_CBC	**12**	**(RFC3602)**
ENCR_AES_CTR	**13**	**(RFC3664)**

values 14-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

Name	Number	Defined In
RESERVED	0	
PRF_HMAC_MD5	1	(RFC2104), [MD5]
PRF_HMAC_SHA1	2	(RFC2104), [SHA]
PRF_HMAC_TIGER	3	(RFC2104)
PRF_AES128_XCBC	4	(RFC3664)

values 5-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

Name	Number	Defined In
NONE	0	
AUTH_HMAC_MD5_96	1	(RFC2403)
AUTH_HMAC_SHA1_96	2	(RFC2404)
AUTH_DES_MAC	3	
AUTH_KPDK_MD5	4	(RFC1826)
AUTH_AES_XCBC_96	5	(RFC3566)

values 6-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

Name	Number
NONE	0
Defined in Appendix B	1 - 2
RESERVED	3 - 4
Defined in [ADDGROUP]	5
RESERVED TO IANA	6 - 13
Defined in [ADDGROUP]	14 - 18
RESERVED TO IANA	19 - 1023
PRIVATE USE	1024-65535

For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

Name	Number
No Extended Sequence Numbers	0
Extended Sequence Numbers	1
RESERVED	2 - 65535

Identifier: RQ_002_6284
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:
When sending a Proposal Substructure containing one or more Transform Substructure within a Security Association Payload, an IKE implementation MUST set the Transform ID field in each Transform Substructure to one of the following values if the Transform Type in the same substructure is set to 2 - Pseudo-Random Function (PRF):

```
Name                  Value
---------------------------
None                  0
PRF_HMAC_MD5          1
PRF_HMAC_SHA1         2
PRF_HMAC_TIGER        3
PRF_AES128_XCBC       4
Reserved to IANA      5 to 1023
Private Use           1024 to 65535
```

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Transform Substructure

- 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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.

- RESERVED - MUST be sent as zero; MUST be ignored on receipt.

- Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.

- Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.

- Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

```
                               Transform    Used In
                                  Type
     RESERVED                       0
     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)
     RESERVED TO IANA                6-240
     PRIVATE USE                     241-255
```

For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

Name	Number	Defined In
RESERVED	0	
ENCR_DES_IV64	1	(RFC1827)
ENCR_DES	2	(RFC2405), [DES]
ENCR_3DES	3	(RFC2451)
ENCR_RC5	4	(RFC2451)
ENCR_IDEA	5	(RFC2451), [IDEA]
ENCR_CAST	6	(RFC2451)
ENCR_BLOWFISH	7	(RFC2451)
ENCR_3IDEA	8	(RFC2451)
ENCR_DES_IV32	9	
RESERVED	10	
ENCR_NULL	11	(RFC2410)
ENCR_AES_CBC	12	(RFC3602)
ENCR_AES_CTR	13	(RFC3664)

values 14-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

Name	**Number**	**Defined In**
RESERVED	**0**	
PRF_HMAC_MD5	**1**	**(RFC2104), [MD5]**
PRF_HMAC_SHA1	**2**	**(RFC2104), [SHA]**
PRF_HMAC_TIGER	**3**	**(RFC2104)**
PRF_AES128_XCBC	**4**	**(RFC3664)**

values 5-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

Name	Number	Defined In
NONE	0	
AUTH_HMAC_MD5_96	1	(RFC2403)
AUTH_HMAC_SHA1_96	2	(RFC2404)
AUTH_DES_MAC	3	
AUTH_KPDK_MD5	4	(RFC1826)
AUTH_AES_XCBC_96	5	(RFC3566)

values 6-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

Name	Number
NONE	0
Defined in Appendix B	1 - 2
RESERVED	3 - 4
Defined in [ADDGROUP]	5
RESERVED TO IANA	6 - 13
Defined in [ADDGROUP]	14 - 18
RESERVED TO IANA	19 - 1023
PRIVATE USE	1024-65535

For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

Name	Number
No Extended Sequence Numbers	0
Extended Sequence Numbers	1
RESERVED	2 - 65535

Identifier: RQ_002_6285
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:

When sending a Proposal Substructure containing one or more Transform Substructure within a Security Association Payload, an IKE implementation MUST set the Transform ID field in each Transform Substructure to one of the following values if the Transform Type in the same substructure is set to 3 - Integrity Algorithm:

```
Name                 Value
--------------------------
NONE                 0
AUTH_HMAC_MD5_96     1
AUTH_HMAC_SHA1_96    2
AUTH_DES_MAC         3
AUTH_KPDK_MD5        4
AUTH_AES_XCBC_96     5
Reserved to IANA     6 to 1023
Private Use          1024 to 65535
```

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        Figure 8:  Transform Substructure
```

o 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

o Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.

o Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.

o Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

```
                                Transform    Used In
                                   Type
     RESERVED                        0
     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)
     RESERVED TO IANA                6-240
     PRIVATE USE                     241-255
```

```
For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

     Name                     Number           Defined In
     RESERVED                   0
     ENCR_DES_IV64              1               (RFC1827)
     ENCR_DES                   2               (RFC2405), [DES]
     ENCR_3DES                  3               (RFC2451)
     ENCR_RC5                   4               (RFC2451)
     ENCR_IDEA                  5               (RFC2451), [IDEA]
     ENCR_CAST                  6               (RFC2451)
     ENCR_BLOWFISH              7               (RFC2451)
     ENCR_3IDEA                 8               (RFC2451)
     ENCR_DES_IV32              9
     RESERVED                  10
     ENCR_NULL                 11               (RFC2410)
     ENCR_AES_CBC              12               (RFC3602)
     ENCR_AES_CTR              13               (RFC3664)

     values 14-1023 are reserved to IANA.  Values 1024-65535 are
     for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

     Name                     Number               Defined In
     RESERVED                   0
     PRF_HMAC_MD5               1                  (RFC2104), [MD5]
     PRF_HMAC_SHA1              2                  (RFC2104), [SHA]
     PRF_HMAC_TIGER             3                  (RFC2104)
     PRF_AES128_XCBC            4                  (RFC3664)

     values 5-1023 are reserved to IANA. Values 1024-65535 are for
     private use among mutually consenting parties.
```

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

```
     Name                     Number                 Defined In
     NONE                      0
     AUTH_HMAC_MD5_96          1                      (RFC2403)
     AUTH_HMAC_SHA1_96         2                      (RFC2404)
     AUTH_DES_MAC              3
     AUTH_KPDK_MD5             4                      (RFC1826)
     AUTH_AES_XCBC_96          5                      (RFC3566)
```

values 6-1023 are reserved to IANA. Values 1024-65535 are for private use among mutually consenting parties.

```
For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

     Name                                Number
     NONE                               0
     Defined in Appendix B              1 - 2
     RESERVED                           3 - 4
     Defined in [ADDGROUP]              5
     RESERVED TO IANA                   6 - 13
     Defined in [ADDGROUP]              14 - 18
     RESERVED TO IANA                   19 - 1023
     PRIVATE USE                        1024-65535

For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

     Name                                Number
     No Extended Sequence Numbers       0
     Extended Sequence Numbers          1
     RESERVED                           2 - 65535


--------------------
```

Identifier: RQ_002_6286
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:

When sending a Proposal Substructure containing one or more Transform Substructure within a Security Association Payload, an IKE implementation MUST set the Transform ID field in each Transform Substructure to one of the following values if the Transform Type in the same substructure is set to 4 - Diffie-Hellman group

```
Name                                                Value
----------------------------------------------------------
NONE                                                0
Group 1 (as defined in Appendix B of RFC4306)       1
Group 2 (as defined in Appendix B of RFC4306)       2
Reserved                                            3 to 4
Group 5 (as defined in RFC3526)                     5
Reserved to IANA                                    6 to 13
Group 14 (as defined in RFC3526)                    14
Group 15 (as defined in RFC3526)                    15
Group 16 (as defined in RFC3526)                    16
Group 17 (as defined in RFC3526)                    17
Group 18 (as defined in RFC3526)                    18
Reserved to IANA                                    19 to 1023
Private Use                                         1024 to 65535
```

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Transform Substructure

o 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

o Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.

o Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.

o Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

```
                              Transform    Used In
                                 Type
RESERVED                          0
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)
RESERVED TO IANA                  6-240
PRIVATE USE                       241-255
```

For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

```
Name                     Number           Defined In
RESERVED                   0
ENCR_DES_IV64              1               (RFC1827)
ENCR_DES                   2               (RFC2405), [DES]
```

```
     ENCR_3DES                  3              (RFC2451)
     ENCR_RC5                   4              (RFC2451)
     ENCR_IDEA                  5              (RFC2451), [IDEA]
     ENCR_CAST                  6              (RFC2451)
     ENCR_BLOWFISH              7              (RFC2451)
     ENCR_3IDEA                 8              (RFC2451)
     ENCR_DES_IV32              9
     RESERVED                  10
     ENCR_NULL                 11              (RFC2410)
     ENCR_AES_CBC              12              (RFC3602)
     ENCR_AES_CTR              13              (RFC3664)

     values 14-1023 are reserved to IANA. Values 1024-65535 are
     for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

     Name                     Number                Defined In
     RESERVED                   0
     PRF_HMAC_MD5               1                   (RFC2104), [MD5]
     PRF_HMAC_SHA1              2                   (RFC2104), [SHA]
     PRF_HMAC_TIGER             3                   (RFC2104)
     PRF_AES128_XCBC            4                   (RFC3664)

     values 5-1023 are reserved to IANA. Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

     Name                     Number                  Defined In
     NONE                      0
     AUTH_HMAC_MD5_96          1                       (RFC2403)
     AUTH_HMAC_SHA1_96         2                       (RFC2404)
     AUTH_DES_MAC              3
     AUTH_KPDK_MD5             4                       (RFC1826)
     AUTH_AES_XCBC_96          5                       (RFC3566)

     values 6-1023 are reserved to IANA.  Values 1024-65535 are for
     private use among mutually consenting parties.
```

For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

Name	**Number**
NONE	**0**
Defined in Appendix B	**1 - 2**
RESERVED	**3 - 4**
Defined in [ADDGROUP]	**5**
RESERVED TO IANA	**6 - 13**
Defined in [ADDGROUP]	**14 - 18**
RESERVED TO IANA	**19 - 1023**
PRIVATE USE	**1024-65535**

```
For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

     Name                                Number
     No Extended Sequence Numbers       0
     Extended Sequence Numbers          1
     RESERVED                           2 - 65535


--------------------
```

Identifier: RQ_002_6287
RFC Clause: 3.3.2
Type: Mandatory
Applies to: Host

Requirement:

When sending a Proposal Substructure containing one or more Transform Substructure within a Security Association Payload, an IKE implementation MUST set the Transform ID field in each Transform Substructure to one of the following values if the Transform Type in the same substructure is set to 5 - Extended Sequence Numbers

```
Name                               Value
----------------------------------------
No Extended Sequence Numbers       0
Extended Sequence Numbers          1
Reserved                           2 to 65535
```

RFC Text:

```
                     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                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 8:  Transform Substructure
```

o 0 (last) or 3 (more) (1 octet) - Specifies whether this is the last Transform Substructure in the Proposal. This syntax is inherited from ISAKMP, but is unnecessary because the last Proposal could be identified from the length of the SA. 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.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

o Transform Length - The length (in octets) of the Transform Substructure including Header and Attributes.

o Transform Type (1 octet) - The type of transform being specified in this transform. Different protocols support different transform types. For some protocols, some of the transforms may be optional. If a transform is optional and the initiator wishes to propose that the transform be omitted, no transform of the given type is included in the proposal. If the initiator wishes to make use of the transform optional to the responder, it includes a transform substructure with transform ID = 0 as one of the options.

o Transform ID (2 octets) - The specific instance of the transform type being proposed.

Transform Type Values

```
                                Transform    Used In
                                   Type
     RESERVED                        0
     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)
     RESERVED TO IANA                6-240
     PRIVATE USE                     241-255
```

For Transform Type 1 (Encryption Algorithm), defined Transform IDs are:

```
     Name                     Number           Defined In
     RESERVED                    0
     ENCR_DES_IV64               1              (RFC1827)
```

```
     ENCR_DES                    2             (RFC2405), [DES]
     ENCR_3DES                   3             (RFC2451)
     ENCR_RC5                    4             (RFC2451)
     ENCR_IDEA                   5             (RFC2451), [IDEA]
     ENCR_CAST                   6             (RFC2451)
     ENCR_BLOWFISH               7             (RFC2451)
     ENCR_3IDEA                  8             (RFC2451)
     ENCR_DES_IV32               9
     RESERVED                   10
     ENCR_NULL                  11             (RFC2410)
     ENCR_AES_CBC               12             (RFC3602)
     ENCR_AES_CTR               13             (RFC3664)

     values 14-1023 are reserved to IANA. Values 1024-65535 are
     for private use among mutually consenting parties.

For Transform Type 2 (Pseudo-random Function), defined Transform IDs are:

     Name                     Number              Defined In
     RESERVED                    0
     PRF_HMAC_MD5                1                (RFC2104), [MD5]
     PRF_HMAC_SHA1               2                (RFC2104), [SHA]
     PRF_HMAC_TIGER              3                (RFC2104)
     PRF_AES128_XCBC             4                (RFC3664)

     values 5-1023 are reserved to IANA. Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 3 (Integrity Algorithm), defined Transform IDs are:

     Name                     Number                Defined In
     NONE                       0
     AUTH_HMAC_MD5_96           1                    (RFC2403)
     AUTH_HMAC_SHA1_96          2                    (RFC2404)
     AUTH_DES_MAC               3
     AUTH_KPDK_MD5              4                    (RFC1826)
     AUTH_AES_XCBC_96           5                    (RFC3566)

     values 6-1023 are reserved to IANA. Values 1024-65535 are for
     private use among mutually consenting parties.

For Transform Type 4 (Diffie-Hellman Group), defined Transform IDs are:

     Name                               Number
     NONE                              0
     Defined in Appendix B             1 - 2
     RESERVED                          3 - 4
     Defined in [ADDGROUP]             5
     RESERVED TO IANA                  6 - 13
     Defined in [ADDGROUP]             14 - 18
     RESERVED TO IANA                  19 - 1023
     PRIVATE USE                       1024-65535
```

For Transform Type 5 (Extended Sequence Numbers), defined Transform IDs are:

```
     Name                               Number
     No Extended Sequence Numbers      0
     Extended Sequence Numbers         1
     RESERVED                          2 - 65535

--------------------
```

Identifier: RQ_002_6288
RFC Clause: 3.3.3
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST support the following transform types:

```
  Encryption Algorithm
  Pseudo-Random Function
  Integrity Algorithm
  Diffie-Hellman Group
```

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

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

Identifier: RQ_002_6289
RFC Clause: 3.3.3
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation which also supports the ESP and AH protocols MUST support the Extended Sequence Numbers transform type

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

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

Identifier: RQ_002_6290
RFC Clause: 3.3.3
Type: Optional
Applies to: Host

Requirement:
When sending an ESP Proposal Substructure within a Security Association Payload, an IKE implementation MAY omit the Integrity Transform Substructure if the only Integrity value it supports is "None"

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

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

Identifier: RQ_002_6291
RFC Clause: 3.3.3
Type: Optional
Applies to: Host

Requirement:
When sending an ESP Proposal Substructure within a Security Association Payload, an IKE implementation MAY omit the Diffie-Hellman Group Transform Substructure if the only Diffie-Hellman Group value it supports is "None"

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

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

Identifier: RQ_002_6292
RFC Clause: 3.3.3
Type: Optional
Applies to: Host

Requirement:
When sending an AH Proposal Substructure within a Security Association Payload, an IKE implementation MAY omit the Diffie-Hellman Group Transform Substructure if the only Diffie-Hellman Group value it supports is "None"

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

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

Identifier: RQ_002_6293
RFC Clause: 3
Type: Mandatory
Applies to: Host

Requirement:
Whenever an IKE implementation sends an IKE message, it MUST set all fields identified in RFC 4306 as "Reserved" to zero (0)

RFC Text:
*** General Text which appears in numerous locations throughout RFC 4306 Section 3 ***

 -- X(reserved) - **These bits MUST be cleared when sending**
                   and MUST be ignored on receipt.

Identifier: RQ_002_6294
RFC Clause: 3
Type: Mandatory
Applies to: Host

Requirement:
Whenever an IKE implementation receives an IKE message, it MUST ignore all fields identified in RFC4306 as "Reserved"

RFC Text:
*** General Text which appears in numerous locations throughout RFC 4306 Section 3 ***

```
 --  X(reserved) - These bits MUST be cleared when sending
                   and MUST be ignored on receipt.
```

Identifier: RQ_002_6295
RFC Clause: 3.3.4
Type: Optional
Applies to: Host

Requirement:
An IKE implementation may support suites of Transform Types (to support protocols other than ESP and AH, for example) in addition to those mandated by RFC4306

RFC Text:
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. 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. The implementation MUST reject SA proposals that are not authorized by these IKE suite controls. Note that cryptographic suites that MUST be implemented need not be configured as acceptable to local policy.

Identifier: RQ_002_6296
RFC Clause: 3.3.5
Type: Optional
Applies to: Host

Requirement:

A Transform Substructure within a Proposal Substructure of an IKE Security Association Payload MAY include Transform Attributes

RFC Text:

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. 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.

```
                     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                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Data Attributes

o Attribute Type (2 octets) - Unique identifier for each type of attribute (see below).

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.

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.

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. If the AF bit is a one (1), the Attribute Value has a length of 2 octets.

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.

Attributes described as basic MUST NOT be encoded using the variable-length encoding. Variable-length attributes MUST NOT be encoded as basic even if their value can fit into two octets. NOTE: This is a change from IKEv1, where increased flexibility may have simplified the composer of messages but certainly complicated the parser.

Attribute Type	Value	Attribute Format
RESERVED	0-13	
Key Length (in bits)	14	TV
RESERVED	15-17	
RESERVED TO IANA	18-16383	
PRIVATE USE	16384-32767	

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.

- 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). This attribute MUST NOT be used when the specified
 Encryption Algorithm uses a fixed-length key.

--------------------
```

Identifier: RQ_002_6297
RFC Clause: 3.3.5
Type: Mandatory
Applies to: Host

Requirement:

If included in a Transform Substructure within a Proposal Substructure of an IKE Security Association Payload, Transport Attributes MUST be constructed in the following format:

```
Octet            Field
-----------------------
1 & 2            Attribute Type
3 & 4            Attribute Value
```

RFC Text:

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. 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.

```
                     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                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Data Attributes

- **Attribute Type (2 octets) - Unique identifier for each type of attribute (see below).**

  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.

- 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.

- 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. If the AF bit is a one (1), the Attribute Value has a length of 2 octets.

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.

Attributes described as basic MUST NOT be encoded using the variable-length encoding. Variable-length attributes MUST NOT be encoded as basic even if their value can fit into two octets. NOTE: This is a change from IKEv1, where increased flexibility may have simplified the composer of messages but certainly complicated the parser.

Attribute Type	Value	Attribute Format
RESERVED	0-13	
Key Length (in bits)	14	TV
RESERVED	15-17	
RESERVED TO IANA	18-16383	
PRIVATE USE	16384-32767	

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.

```
- 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). This attribute MUST NOT be used when the specified
 Encryption Algorithm uses a fixed-length key.

--------------------
```

Identifier: RQ_002_6298
RFC Clause: 3.3.5
Type: Mandatory
Applies to: Host

Requirement:

When sending a Transform Substructure containing one or more Transform Attributes within a Security Association Payload, an IKE implementation MUST set the Attribute Value field in each Transform Attribute either to 14 (Key Length) with the Attribute Format Flag (AF) set to 1 or to a value in the range 18 to 32767 (for private use only between mutually consenting parties)

RFC Text:

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. 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.

```
                     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                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Data Attributes

o Attribute Type (2 octets) - Unique identifier for each type of attribute (see below).

   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.

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.

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. If the AF bit is a one (1), the Attribute Value has a length of 2 octets.

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.

Attributes described as basic MUST NOT be encoded using the variable-length encoding. Variable-length attributes MUST NOT be encoded as basic even if their value can fit into two octets. NOTE: This is a change from IKEv1, where increased flexibility may have simplified the composer of messages but certainly complicated the parser.

Attribute Type	Value	Attribute Format
RESERVED	0-13	
Key Length (in bits)	14	TV
RESERVED	15-17	
RESERVED TO IANA	18-16383	
PRIVATE USE	16384-32767	

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.

- 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). This attribute MUST NOT be used when the specified
 Encryption Algorithm uses a fixed-length key.

--------------------
```

Identifier: RQ_002_6299
RFC Clause: 3.3.5
Type: Mandatory
Applies to: Host

Requirement:

When sending a Transform Substructure within a Security Association Payload , if the Transform Type is set to Type 1 (Encryption Algorithm) and the Transform ID indicates an algorithm using a variable length key (AES_CBC or AES_CTR) then an IKE implementation MUST set the Attribute Type in the associated Transform Attribute to the value 14 (Key Length), the Attribute Format (AF) flag (most significant bit of the Attribute Type field) to 1 and the Attribute Value to the required key length, in bits, for the selected encryption algorithm

RFC Text:

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. 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.

```
                     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                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Data Attributes

- Attribute Type (2 octets) - Unique identifier for each type of attribute (see below).

  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.

- 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.

- 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. If the AF bit is a one (1), the Attribute Value has a length of 2 octets.

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.

Attributes described as basic MUST NOT be encoded using the variable-length encoding. Variable-length attributes MUST NOT be encoded as basic even if their value can fit into two octets. NOTE: This is a change from IKEv1, where increased flexibility may have simplified the composer of messages but certainly complicated the parser.

Attribute Type	Value	Attribute Format
RESERVED	0-13	
Key Length (in bits)	14	TV
RESERVED	15-17	
RESERVED TO IANA	18-16383	
PRIVATE USE	16384-32767	

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.

- 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). This attribute MUST NOT be used when the specified Encryption Algorithm uses a fixed-length key.

Identifier: RQ_002_6300
RFC Clause: 3.3.6
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives a proposal for a set of choices of IPsec protocols with associated Transforms and Attributes to be used within a Security Association, it MUST select a single complete proposal or reject them all and return an IKE INFORMATIONAL message containing a Notify payload with the Error Type set to NO_PROPOSAL_CHOSEN

RFC Text:
During security association negotiation, initiators present offers to responders. **Responders MUST select a single complete set of parameters from the offers (or reject all offers if none are acceptable)**. If there are multiple proposals, the responder MUST choose a single proposal number and return all of the Proposal substructures with that Proposal number. If there are multiple Transforms with the same type, the responder MUST choose a single one. Any attributes of a selected transform MUST be returned unmodified. 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

----

Identifier: RQ_002_6301
RFC Clause: 3.3.6
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation selects a proposal from a received set of choices of IPsec protocols to be used within a Security Association, the selected cryptographic suite MUST contain exactly one transform of each type included in the proposal.

RFC Text:
During security association negotiation, initiators present offers to responders. Responders MUST select a single complete set of parameters from the offers (or reject all offers if none are acceptable). **If there are multiple proposals, the responder MUST choose a single proposal number and return all of the Proposal substructures with that Proposal number. If there are multiple Transforms with the same type, the responder MUST choose a single one.** Any attributes of a selected transform MUST be returned unmodified. 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

----

Identifier: RQ_002_6302
RFC Clause: 3.3.6
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE response to its proposal of a possible set of choices of IPsec protocols to be used within a Security Association, it MUST reject the response if the parameters in the acceptance are not consistent with one of its original proposals

RFC Text:
During security association negotiation, initiators present offers to responders. Responders MUST select a single complete set of parameters from the offers (or reject all offers if none are acceptable). If there are multiple proposals, the responder MUST choose a single proposal number and return all of the Proposal substructures with that Proposal number. If there are multiple Transforms with the same type, the responder MUST choose a single one. Any attributes of a selected transform MUST be returned unmodified. **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**

Identifier: RQ_002_6303
RFC Clause: 3.3.6
Type: Recommended
Applies to: Host

Requirement:
An IKE implementation SHOULD NOT reject a proposal within a Security Association Payload for the single reason that one (or more) of the associated Transform Attribute values are outside the range that the implementation is configured to accept but are legitimate values which would result in greater security on the resultant SA

RFC Text:
Implementation Note:

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.** 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.

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".

Identifier: RQ_002_6304
RFC Clause: 3.4
Type: Mandatory
Applies to: Host

Requirement:
A Key Exchange Payload i an IKE packet MUST be constructed as follows:

Octet	Field
1 to 4	IKE Generic Payload Header
5 & 6	Diffie-Hellman Group Number
7 & 8	Reserved
9 to End	Key Exchange Data

RFC Text:
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.

```
                     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                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Key Exchange Payload Format

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

The DH Group # identifies the Diffie-Hellman group in which the Key Exchange Data was computed (see section 3.3.2). If the selected proposal uses a different Diffie-Hellman group, the message MUST be rejected with a Notify payload of type INVALID_KE_PAYLOAD.

The payload type for the Key Exchange payload is thirty four (34)

Identifier: RQ_002_6305
RFC Clause: 3.4
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends a packet containing a Key Exchange Payload it MUST set the Diffie-Hellman Group Number field to one of the following values as appropriate:

Name	Value
NONE	0
Group 1 (as defined in Appendix B of RFC4306)	1
Group 2 (as defined in Appendix B of RFC4306)	2
Reserved	3 to 4
Group 5 (as defined in RFC3526)	5
Reserved to IANA	6 to 13
Group 14 (as defined in RFC3526)	14
Group 15 (as defined in RFC3526)	15
Group 16 (as defined in RFC3526)	16
Group 17 (as defined in RFC3526)	17
Group 18 (as defined in RFC3526)	18
Reserved to IANA	19 to 1023
Private Use	1024 to 65535

RFC Text:
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.

```
                    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                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Key Exchange Payload Format

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

The DH Group # identifies the Diffie-Hellman group in which the Key Exchange Data was computed (see section 3.3.2). If the selected proposal uses a different Diffie-Hellman group, the message MUST be rejected with a Notify payload of type INVALID_KE_PAYLOAD.

The payload type for the Key Exchange payload is thirty four (34)

Identifier: RQ_002_6306
RFC Clause: 3.4
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation receives a packet containing a Key Exchange Payload which identifies a Diffie-Hellman Group Number which is not the same as the Group identified in the previously selected SA proposal, it MUST reject the Key Exchange Payload with a Notify Payload with the Error Type set to INVALID_KE_PAYLOAD

RFC Text:

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.

```
                      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                       ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

           Figure 10:  Key Exchange Payload Format
```

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

The DH Group # identifies the Diffie-Hellman group in which the Key Exchange Data was computed (see section 3.3.2). **If the selected proposal uses a different Diffie-Hellman group, the message MUST be rejected with a Notify payload of type INVALID_KE_PAYLOAD**.

The payload type for the Key Exchange payload is thirty four (34)

Identifier: RQ_002_6307
RFC Clause: 3.4
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Key Exchange Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Key Exchange Payload) to the value thirty-four (34)

RFC Text:
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.

```
                     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                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: Key Exchange Payload Format

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

The DH Group # identifies the Diffie-Hellman group in which the Key Exchange Data was computed (see section 3.3.2). If the selected proposal uses a different Diffie-Hellman group, the message MUST be rejected with a Notify payload of type INVALID_KE_PAYLOAD.

The payload type for the Key Exchange payload is thirty four (34)

Identifier: RQ_002_6308
RFC Clause: 3.5
Type: Mandatory
Applies to: Host

Requirement:

An Identification Payload in an IKE packet MUST be constructed as follows:

```
Octet            Field
----------------------
1 to 4           IKE Generic Payload Header
5                ID Type
6 to 8           Reserved
9 to End         Identification Data
```

RFC Text:

The Identification Payload consists of the IKE generic payload header followed by identification fields as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ID Type     !                 RESERVED                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Identification Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Identification Payload Format

o ID Type (1 octet) - Specifies the type of Identification being used.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

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

The payload types for the Identification Payload are thirty five (35) for IDi and thirty six (36) for IDr.

The following table lists the assigned values for the Identification Type field, followed by a description of the Identification Data which follows:

```
ID Type                          Value
-------                          -----
RESERVED                           0

ID_IPV4_ADDR                       1

      A single four (4) octet IPv4 address.

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.).

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.

Reserved to IANA                   4

ID_IPV6_ADDR                       5

      A single sixteen (16) octet IPv6 address.

Reserved to IANA                   6 - 8

ID_DER_ASN1_DN                     9
```

The binary Distinguished Encoding Rules (DER) encoding of an ASN.1 X.500 Distinguished Name [X.501].

ID_DER_ASN1_GN 10

The binary DER encoding of an ASN.1 X.500 GeneralName [X.509].

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.

Reserved to IANA 12-200

Reserved for private use 201-255

Two implementations will interoperate only if each can generate a type of ID acceptable to the other. 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, and MUST be configurable to accept all of these types. Implementations SHOULD be capable of generating and accepting all of these types. IPv6-capable implementations MUST additionally be configurable to accept ID_IPV6_ADDR. IPv6-only implementations MAY be configurable to send only ID_IPV6_ADDR

Identifier: RQ_002_6309
RFC Clause: 3.5
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE packet containing an Identification Payload, it MUST set the correct value from the following table into the ID Type field to characterize the type of identification included in the Identification Data field:

ID Type	Value	Meaning
RESERVED	0	
ID_IPV4_ADDR	1	A single four (4) octet IPv4 address.
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.).
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.
Reserved to IANA	4	
ID_IPV6_ADDR	5	A single sixteen (16) octet IPv6 address.
Reserved to IANA	6 to 8	
ID_DER_ASN1_DN	9	The binary Distinguished Encoding Rules (DER) encoding of an ASN.1 X.500 Distinguished Name [X.501].
ID_DER_ASN1_GN	10	The binary DER encoding of an ASN.1 X.500 GeneralName [X.509].
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.
Reserved to IANA	12-200	
Reserved for private use	201-255	

RFC Text:

The Identification Payload consists of the IKE generic payload header followed by identification fields as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ID Type     !                 RESERVED                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Identification Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Identification Payload Format

o ID Type (1 octet) - Specifies the type of Identification being used.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

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

The payload types for the Identification Payload are thirty five (35) for IDi and thirty six (36) for IDr.

The following table lists the assigned values for the Identification Type field, followed by a description of the Identification Data which follows:

ID Type **Value**
------- **-----**
RESERVED **0**

ID_IPV4_ADDR **1**

A single four (4) octet IPv4 address.

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.).

 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.

 Reserved to IANA                  4

 ID_IPV6_ADDR                      5

      A single sixteen (16) octet IPv6 address.

 Reserved to IANA                  6 - 8

 ID_DER_ASN1_DN                    9

      The binary Distinguished Encoding Rules (DER) encoding of an
      ASN.1 X.500 Distinguished Name [X.501].

 ID_DER_ASN1_GN                    10

      The binary DER encoding of an ASN.1 X.500 GeneralName
      [X.509].

 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.

 Reserved to IANA                  12-200

 Reserved for private use          201-255
```

Two implementations will interoperate only if each can generate a type of ID acceptable to the other. 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, and MUST be configurable to accept all of these types. Implementations SHOULD be capable of generating and accepting all of these types. IPv6-capable implementations MUST additionally be configurable to accept ID_IPV6_ADDR. IPv6-only implementations MAY be configurable to send only ID_IPV6_ADDR

Identifier: RQ_002_6310
RFC Clause: 3.5
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing an Initiator's Identification Payload (IDi), it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Identification Payload) to the value thirty-five (35)

RFC Text:
The Identification Payload consists of the IKE generic payload header followed by identification fields as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ID Type     !                 RESERVED                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Identification Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Identification Payload Format

o ID Type (1 octet) - Specifies the type of Identification being used.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

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

The payload types for the Identification Payload are thirty five (35) for IDi and thirty six (36) for IDr.

The following table lists the assigned values for the Identification Type field, followed by a description of the Identification Data which follows:

```
ID Type                           Value
-------                           -----
RESERVED                            0

ID_IPV4_ADDR                        1

      A single four (4) octet IPv4 address.

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.).

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.

Reserved to IANA                    4

ID_IPV6_ADDR                        5

      A single sixteen (16) octet IPv6 address.

Reserved to IANA                    6 - 8

ID_DER_ASN1_DN                      9

      The binary Distinguished Encoding Rules (DER) encoding of an
      ASN.1 X.500 Distinguished Name [X.501].
```

```
 ID_DER_ASN1_GN                    10

       The binary DER encoding of an ASN.1 X.500 GeneralName
       [X.509].

 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.

 Reserved to IANA                  12-200

 Reserved for private use          201-255
```

Two implementations will interoperate only if each can generate a type of ID acceptable to the other. 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, and MUST be configurable to accept all of these types. Implementations SHOULD be capable of generating and accepting all of these types. IPv6-capable implementations MUST additionally be configurable to accept ID_IPV6_ADDR. IPv6-only implementations MAY be configurable to send only ID_IPV6_ADDR

Identifier: RQ_002_6311
RFC Clause: 3.5
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Responder's Identification Payload (IDr), it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Identification Payload) to the value thirty-six (36)

RFC Text:
The Identification Payload consists of the IKE generic payload header followed by identification fields as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ID Type     !                 RESERVED                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Identification Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 11:  Identification Payload Format
```

o ID Type (1 octet) - Specifies the type of Identification being used.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

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

The payload types for the Identification Payload are thirty five (35) for IDi **and thirty six (36) for IDr**.

The following table lists the assigned values for the Identification Type field, followed by a description of the Identification Data which follows:

```
ID Type                          Value
-------                          -----
RESERVED                           0

ID_IPV4_ADDR                       1

      A single four (4) octet IPv4 address.

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.).

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.

Reserved to IANA                   4

ID_IPV6_ADDR                       5

      A single sixteen (16) octet IPv6 address.

Reserved to IANA                   6 - 8

ID_DER_ASN1_DN                     9

      The binary Distinguished Encoding Rules (DER) encoding of an
      ASN.1 X.500 Distinguished Name [X.501].
```

```
 ID_DER_ASN1_GN                    10

       The binary DER encoding of an ASN.1 X.500 GeneralName
       [X.509].

 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.

 Reserved to IANA                  12-200

 Reserved for private use          201-255
```

Two implementations will interoperate only if each can generate a type of ID acceptable to the other. 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, and MUST be configurable to accept all of these types. Implementations SHOULD be capable of generating and accepting all of these types. IPv6-capable implementations MUST additionally be configurable to accept ID_IPV6_ADDR. IPv6-only implementations MAY be configurable to send only ID_IPV6_ADDR

Identifier: RQ_002_6312
RFC Clause: 3.5
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST be able to support the ID_IPV6_ADDR ID Type in an outgoing Identification Payload

RFC Text:
The Identification Payload consists of the IKE generic payload header followed by identification fields as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ID Type     !                 RESERVED                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Identification Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 11:  Identification Payload Format
```

o ID Type (1 octet) - Specifies the type of Identification being used.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

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

The payload types for the Identification Payload are thirty five (35) for IDi and thirty six (36) for IDr.

The following table lists the assigned values for the Identification Type field, followed by a description of the Identification Data which follows:

```
ID Type                          Value
-------                          -----
RESERVED                           0

ID_IPV4_ADDR                       1

      A single four (4) octet IPv4 address.

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.).

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.

Reserved to IANA                   4

ID_IPV6_ADDR                       5

      A single sixteen (16) octet IPv6 address.

Reserved to IANA                   6 - 8

ID_DER_ASN1_DN                     9

      The binary Distinguished Encoding Rules (DER) encoding of an
      ASN.1 X.500 Distinguished Name [X.501].

ID_DER_ASN1_GN                     10
```

The binary DER encoding of an ASN.1 X.500 GeneralName [X.509].

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.

Reserved to IANA 12-200

Reserved for private use 201-255

Two implementations will interoperate only if each can generate a type of ID acceptable to the other. 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, and MUST be configurable to accept all of these types. Implementations SHOULD be capable of generating and accepting all of these types. IPv6-capable implementations MUST additionally be configurable to accept ID_IPV6_ADDR. **IPv6-only implementations MAY be configurable to send only ID_IPV6_ADDR**

Identifier: RQ_002_6313
RFC Clause: 3.5
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST accept all of the following ID Types in incoming Identification Payloads:

ID_IPV4_ADDR,
ID_IPV6_ADR,
ID_FQDN,
ID_RFC822_ADDR, and
ID_KEY_ID

RFC Text:
The Identification Payload consists of the IKE generic payload header followed by identification fields as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ID Type     !                 RESERVED                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Identification Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Identification Payload Format

o ID Type (1 octet) - Specifies the type of Identification being used.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

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

The payload types for the Identification Payload are thirty five (35) for IDi and thirty six (36) for IDr.

The following table lists the assigned values for the Identification Type field, followed by a description of the Identification Data which follows:

```
ID Type                           Value
-------                           -----
RESERVED                            0

ID_IPV4_ADDR                        1

      A single four (4) octet IPv4 address.

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.).

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.

Reserved to IANA                    4

ID_IPV6_ADDR                        5

      A single sixteen (16) octet IPv6 address.

Reserved to IANA                    6 - 8

ID_DER_ASN1_DN                      9
```

The binary Distinguished Encoding Rules (DER) encoding of an ASN.1 X.500 Distinguished Name [X.501].

ID_DER_ASN1_GN 10

The binary DER encoding of an ASN.1 X.500 GeneralName [X.509].

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.

Reserved to IANA 12-200

Reserved for private use 201-255

Two implementations will interoperate only if each can generate a type of ID acceptable to the other. 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, **and MUST be configurable to accept all of these types**. Implementations SHOULD be capable of generating and accepting all of these types. IPv6-capable implementations MUST additionally be configurable to accept ID_IPV6_ADDR. IPv6-only implementations MAY be configurable to send only ID_IPV6_ADDR

Identifier: RQ_002_6314
RFC Clause: 3.5
Type: Recommended
Applies to: Host

Requirement:
In addition to ID_IPV6_ADDR, an IKE implementation SHOULD support all of the following ID Types in outgoing Identification Payloads:

ID_IPV4_ADDR,
ID_FQDN,
ID_RFC822_ADDR, and
ID_KEY_ID

RFC Text:
The Identification Payload consists of the IKE generic payload header followed by identification fields as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   ID Type     !                 RESERVED                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Identification Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Identification Payload Format

o ID Type (1 octet) - Specifies the type of Identification being used.

o RESERVED - MUST be sent as zero; MUST be ignored on receipt.

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

The payload types for the Identification Payload are thirty five (35) for IDi and thirty six (36) for IDr.

The following table lists the assigned values for the Identification Type field, followed by a description of the Identification Data which follows:

```
ID Type                          Value
-------                          -----
RESERVED                           0

ID_IPV4_ADDR                       1

      A single four (4) octet IPv4 address.

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.).

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.

Reserved to IANA                   4

ID_IPV6_ADDR                       5

      A single sixteen (16) octet IPv6 address.

Reserved to IANA                   6 - 8

ID_DER_ASN1_DN                     9
```

```
      The binary Distinguished Encoding Rules (DER) encoding of an
      ASN.1 X.500 Distinguished Name [X.501].

 ID_DER_ASN1_GN                     10

      The binary DER encoding of an ASN.1 X.500 GeneralName
      [X.509].

 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.

 Reserved to IANA                   12-200

 Reserved for private use           201-255
```

Two implementations will interoperate only if each can generate a type of ID acceptable to the other. 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, and MUST be configurable to accept all of these types. **Implementations SHOULD be capable of generating and accepting all of these types**. IPv6-capable implementations MUST additionally be configurable to accept ID_IPV6_ADDR. IPv6-only implementations MAY be configurable to send only ID_IPV6_ADDR

Identifier: RQ_002_6315
RFC Clause: 3.6
Type: Recommended
Applies to: Host

Requirement:
An IKE implementation SHOULD include a Certificate payload in an IKE exchange if certificates or other authentication-related data are available and if the IKE peer has not previously sent a Notify Payload with the Status Type set to HTTP_CERT_LOOKUP_SUPPORTED

RFC Text:
The Certificate Payload, denoted CERT in this memo, provides a means to transport certificates or other authentication-related information via IKE. **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**. 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.

The Certificate Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                       Certificate Data                        ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 12:  Certificate Payload Format
```

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

Certificate Encoding	Value
RESERVED	0
PKCS #7 wrapped X.509 certificate	1
PGP Certificate	2
DNS Signed Key	3
X.509 Certificate - Signature	4
Kerberos Token	6
Certificate Revocation List (CRL)	7
Authority Revocation List (ARL)	8
SPKI Certificate	9
X.509 Certificate - Attribute	10
Raw RSA Key	11
Hash and URL of X.509 certificate	12
Hash and URL of X.509 bundle	13
RESERVED to IANA	14 - 200
PRIVATE USE	201 - 255

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

The payload type for the Certificate Payload is thirty seven (37).

Identifier: RQ_002_6316
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:

An Identification Payload in an IKE packet MUST be constructed as follows:

```
Octet           Field
----------------------
1 to 4          IKE Generic Payload Header
5               Certificate Encoding indicator
6 to End        Certificate Data
```

RFC Text:

The Certificate Payload, denoted CERT in this memo, provides a means to transport certificates or other authentication-related information via IKE. 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. 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.

The Certificate Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                       Certificate Data                        ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

           Figure 12:  Certificate Payload Format

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

      Certificate Encoding               Value
      --------------------               -----
      RESERVED                             0
      PKCS #7 wrapped X.509 certificate    1
      PGP Certificate                      2
      DNS Signed Key                       3
      X.509 Certificate - Signature        4
      Kerberos Token                       6
      Certificate Revocation List (CRL)    7
      Authority Revocation List (ARL)      8
      SPKI Certificate                     9
      X.509 Certificate - Attribute       10
      Raw RSA Key                         11
      Hash and URL of X.509 certificate   12
      Hash and URL of X.509 bundle        13
      RESERVED to IANA                  14 - 200
      PRIVATE USE                      201 - 255

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

The payload type for the Certificate Payload is thirty seven (37).

Identifier: RQ_002_6317
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing an Certificate Payload, it MUST set the correct value from the following table into the Certificate Encoding field to indicate the method used to encode the authentication information included in the Certificate Data field:

Certificate Encoding	Value
RESERVED	0
PKCS #7 wrapped X.509 certificate	1
PGP Certificate	2
DNS Signed Key	3
X.509 Certificate - Signature	4
Kerberos Token	6
Certificate Revocation List (CRL)	7
Authority Revocation List (ARL)	8
SPKI Certificate	9
X.509 Certificate - Attribute	10
Raw RSA Key	11
Hash and URL of X.509 certificate	12
Hash and URL of X.509 bundle	13
RESERVED to IANA	14 - 200
PRIVATE USE	201 - 255

RFC Text:
The Certificate Payload, denoted CERT in this memo, provides a means to transport certificates or other authentication-related information via IKE. 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. 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.

The Certificate Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                       Certificate Data                        ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Certificate Payload Format

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

Certificate Encoding	**Value**
RESERVED	**0**
PKCS #7 wrapped X.509 certificate	**1**
PGP Certificate	**2**
DNS Signed Key	**3**
X.509 Certificate - Signature	**4**
Kerberos Token	**6**
Certificate Revocation List (CRL)	**7**
Authority Revocation List (ARL)	**8**
SPKI Certificate	**9**
X.509 Certificate - Attribute	**10**
Raw RSA Key	**11**
Hash and URL of X.509 certificate	**12**
Hash and URL of X.509 bundle	**13**
RESERVED to IANA	**14 - 200**
PRIVATE USE	**201 - 255**

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


The payload type for the Certificate Payload is thirty seven (37).

--------------------
```

Identifier: RQ_002_6318
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing a Certificate Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Certificate Payload) to the value thirty-seven (37)

RFC Text:

The Certificate Payload, denoted CERT in this memo, provides a means to transport certificates or other authentication-related information via IKE. 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. 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.

The Certificate Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                       Certificate Data                        ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

           Figure 12:  Certificate Payload Format
```

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

```
      Certificate Encoding               Value
      --------------------               -----
      RESERVED                             0
      PKCS #7 wrapped X.509 certificate    1
      PGP Certificate                      2
      DNS Signed Key                       3
      X.509 Certificate - Signature        4
      Kerberos Token                       6
      Certificate Revocation List (CRL)    7
      Authority Revocation List (ARL)      8
      SPKI Certificate                     9
      X.509 Certificate - Attribute       10
      Raw RSA Key                         11
      Hash and URL of X.509 certificate   12
      Hash and URL of X.509 bundle        13
      RESERVED to IANA                  14 - 200
      PRIVATE USE                      201 - 255
```

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

The payload type for the Certificate Payload is thirty seven (37).

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:

X.509 Certificate - Signature (4) contains a DER encoded X.509 certificate whose public key is used to validate the sender's AUTH payload.

Certificate Revocation List (7) contains a DER encoded X.509 certificate revocation list.

Raw RSA Key (11) contains a PKCS #1 encoded RSA key (see [RSA] and [PKCS1]).

```
 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. 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].

 Use the following ASN.1 definition for an X.509 bundle:

       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
```

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). Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6319
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE packet containing a Certificate Payload with the Certificate Encoding field set to "X.509 Certificate - Signature" (value 4), it MUST ensure that the Certificate Data field contains a X.509 certificate encoded using the ASN.1 Distinguished Encoding Rules (DER)

RFC Text:

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:

X.509 Certificate - Signature (4) contains a DER encoded X.509 certificate whose public key is used to validate the sender's AUTH payload.

Certificate Revocation List (7) contains a DER encoded X.509 certificate revocation list.

Raw RSA Key (11) contains a PKCS #1 encoded RSA key (see [RSA] and [PKCS1]).

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. 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].

Use the following ASN.1 definition for an X.509 bundle:

```
      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
```

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). Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6320
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE packet containing a Certificate Payload with the Certificate Encoding field set to "Certificate Revocation List" (value 7), it MUST ensure that the Certificate Data field contains a X.509 certificate revocation list encoded using the ASN.1 Distinguished Encoding Rules (DER)

RFC Text:

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:

X.509 Certificate - Signature (4) contains a DER encoded X.509 certificate whose public key is used to validate the sender's AUTH payload.

Certificate Revocation List (7) contains a DER encoded X.509 certificate revocation list.

Raw RSA Key (11) contains a PKCS #1 encoded RSA key (see [RSA] and [PKCS1]).

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. 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].

Use the following ASN.1 definition for an X.509 bundle:

```
      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
```

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). Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6321
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing a Certificate Payload with the Certificate Encoding field set to "Raw RSA Key" (value 11), it MUST ensure that the Certificate Data field contains a RSA key encoded using Public-Key Cryptography Standards (PKCS) #1

RFC Text:
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:

X.509 Certificate - Signature (4) contains a DER encoded X.509 certificate whose public key is used to validate the sender's AUTH payload.

Certificate Revocation List (7) contains a DER encoded X.509 certificate revocation list.

Raw RSA Key (11) contains a PKCS #1 encoded RSA key (see [RSA] and [PKCS1]).

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. 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].

Use the following ASN.1 definition for an X.509 bundle:

```
 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
```

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). Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6322
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE packet containing a Certificate Payload with the Certificate Encoding field set to "Hash and URL of X.509 bundle" (value 13), it MUST ensure that the Certificate Data field contains a X.509 Bundle encoded using the Secure Hash Standard and conforming to the following ASN.1 definition:

```
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
```

RFC Text:

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:

X.509 Certificate - Signature (4) contains a DER encoded X.509 certificate whose public key is used to validate the sender's AUTH payload.

Certificate Revocation List (7) contains a DER encoded X.509 certificate revocation list.

Raw RSA Key (11) contains a PKCS #1 encoded RSA key (see [RSA] and [PKCS1]).

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. 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].

Use the following ASN.1 definition for an X.509 bundle:

```
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
```

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). Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6323
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST be able to send up to four (4) X.509 certificates in Certificate Payloads for each authentication attempt

RFC Text:
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). Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6324
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST be able to accept up to four (4) X.509 certificates in Certificate Payloads for each authentication attempt

RFC Text:
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). Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6325
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST support the Hash and URL of X.509 certificate encoding in outgoing Certificate Payloads

RFC Text:
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).** Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6326
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST support the Hash and URL of X.509 certificate encoding in incoming Certificate Payloads

RFC Text:
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)**. Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6327
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST support the Hash and URL of X.509 bundle encoding in outgoing Certificate Payloads

RFC Text:
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)**. Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6328
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST support the Hash and URL of X.509 bundle encoding in incoming Certificate Payloads

RFC Text:
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)**. Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. 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.

Identifier: RQ_002_6329
RFC Clause: 3.6
Type: Recommended
Applies to: Host

Requirement:
An IKE implementation SHOULD support the Raw RSA keys encoding in outgoing Certificate Payloads

RFC Text:
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). **Implementations SHOULD be capable of being configured to send and accept Raw RSA keys**. 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.

Identifier: RQ_002_6330
RFC Clause: 3.6
Type: Recommended
Applies to: Host

Requirement:
An IKE implementation SHOULD support the Raw RSA keys encoding in incoming Certificate Payloads

RFC Text:
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). **Implementations SHOULD be capable of being configured to send and accept Raw RSA keys**. 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.

Identifier: RQ_002_6331
RFC Clause: 3.6
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends multiple Certificate Payloads in support of a single Authentication attempt, it MUST ensure that the first Certificate Payload contains the public key used to sign the associated Authentication Payload

RFC Text:
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). Implementations SHOULD be capable of being configured to send and accept Raw RSA keys. **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.

Identifier: RQ_002_6332
RFC Clause: 3.7
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Certificate Request Payload in an IKE_INIT_SA response

RFC Text:
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. If multiple CAs are trusted and the cert encoding does not allow a list, then multiple Certificate Request payloads SHOULD be transmitted.

The Certificate Request Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                    Certification Authority                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Certificate Request Payload Format

- o Certificate Encoding (1 octet) - Contains an encoding of the type or format of certificate requested. Values are listed in section 3.6.

- o Certification Authority (variable length) - Contains an encoding of an acceptable certification authority for the type of certificate requested.

The payload type for the Certificate Request Payload is thirty eight (38).

Identifier: RQ_002_6333
RFC Clause: 3.7
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Certificate Request Payload in an IKE_AUTH request

RFC Text:
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. If multiple CAs are trusted and the cert encoding does not allow a list, then multiple Certificate Request payloads SHOULD be transmitted.

The Certificate Request Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                    Certification Authority                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Certificate Request Payload Format

o Certificate Encoding (1 octet) - Contains an encoding of the type or format of certificate requested. Values are listed in section 3.6.

o Certification Authority (variable length) - Contains an encoding of an acceptable certification authority for the type of certificate requested.

The payload type for the Certificate Request Payload is thirty eight (38).

Identifier: RQ_002_6334
RFC Clause: 3.7
Type: Recommended
Applies to: Host

Requirement:
An IKE implementation SHOULD include a separate Certificate Request Payload in the IKE_SA_INIT response or IKE_AUTH request for each available Certification Authority

RFC Text:
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. **If multiple CAs are trusted and the cert encoding does not allow a list, then multiple Certificate Request payloads SHOULD be transmitted.**

The Certificate Request Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                    Certification Authority                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Certificate Request Payload Format

o Certificate Encoding (1 octet) - Contains an encoding of the type or format of certificate requested. Values are listed in section 3.6.

o Certification Authority (variable length) - Contains an encoding of an acceptable certification authority for the type of certificate requested.

The payload type for the Certificate Request Payload is thirty eight (38).

Identifier: RQ_002_6335
RFC Clause: 3.7
Type: Mandatory
Applies to: Host

Requirement:
An Identification Payload in an IKE packet MUST be constructed as follows:

```
Octet            Field
----------------------
1 to 4           IKE Generic Payload Header
5                Certificate Encoding indicator
6 to end         Certification Authority identifier
```

RFC Text:
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. If multiple CAs are trusted and the cert encoding does not allow a list, then multiple Certificate Request payloads SHOULD be transmitted.

The Certificate Request Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                    Certification Authority                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Certificate Request Payload Format

o Certificate Encoding (1 octet) - Contains an encoding of the type or format of certificate requested. Values are listed in section 3.6.

o Certification Authority (variable length) - Contains an encoding of an acceptable certification authority for the type of certificate requested.

The payload type for the Certificate Request Payload is thirty eight (38).

Identifier: RQ_002_6336
RFC Clause: 3.7
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing an Certificate Request Payload, it MUST set the correct value from the following table into the Certificate Encoding field to indicate the method to be used by the responding IKE endpoint to encode the authentication information included in the Certificate Data field of the returned Certificate Payload:

Certificate Encoding	Value
RESERVED	0
PKCS #7 wrapped X.509 certificate	1
PGP Certificate	2
DNS Signed Key	3
X.509 Certificate - Signature	4
Kerberos Token	6
Certificate Revocation List (CRL)	7
Authority Revocation List (ARL)	8
SPKI Certificate	9
X.509 Certificate - Attribute	10
Raw RSA Key	11
Hash and URL of X.509 certificate	12
Hash and URL of X.509 bundle	13
RESERVED to IANA	14 - 200
PRIVATE USE	201 - 255

RFC Text:
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. If multiple CAs are trusted and the cert encoding does not allow a list, then multiple Certificate Request payloads SHOULD be transmitted.

The Certificate Request Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Cert Encoding !                                               !
+-+-+-+-+-+-+-+-+                                               !
~                    Certification Authority                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Certificate Request Payload Format

- **Certificate Encoding (1 octet) - Contains an encoding of the type or format of certificate requested. Values are listed in section 3.6.**

- Certification Authority (variable length) - Contains an encoding of an acceptable certification authority for the type of certificate requested.

The payload type for the Certificate Request Payload is thirty eight (38).

Identifier: RQ_002_6337
RFC Clause: 3.7
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing an Certificate Request Payload (CERTREQ), it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Certificate Request Payload) to the value thirty-eight (38)

RFC Text:
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. If multiple CAs are trusted and the cert encoding does not allow a list, then multiple Certificate Request payloads SHOULD be transmitted.

The Certificate Request Payload is defined as follows:

```
                     1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !C!  RESERVED   !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Cert Encoding !                                               !
 +-+-+-+-+-+-+-+-+                                               !
 ~                    Certification Authority                    ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Certificate Request Payload Format

o Certificate Encoding (1 octet) - Contains an encoding of the type or format of certificate requested. Values are listed in section 3.6.

o Certification Authority (variable length) - Contains an encoding of an acceptable certification authority for the type of certificate requested.

The payload type for the Certificate Request Payload is thirty eight (38).

Identifier: RQ_002_6338
RFC Clause: 3.7
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing an Certificate Request Payload (CERTREQ), it MUST set the Certification Authority field to a value constructed as a concatenated list of SHA-1 hashes of the public keys of the trusted Certification Authorities for the certificate type indicated in the Certificate Encoding field

RFC Text:
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.

Identifier: RQ_002_6339
RFC Clause: 3.7
Type: Recommended
Applies to: Host

Requirement:
If an IKE implementation is enabled to send Certificate payloads and has access to a certificate that satisfies the criteria specified in a received Certificate Request payload, it SHOULD send a corresponding Certificate payload back to the originator of the Certificate Request payload.

RFC Text:
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.

Identifier: RQ_002_6340
RFC Clause: 3.7
Type: Recommended
Applies to: Host

Requirement:
If an IKE implementation is enabled to send Certificate payloads but has no access to a certificate that satisfies the criteria specified in a received Certificate Request payload, it SHOULD send no response to the originator of the Certificate Request payload.

RFC Text:
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.

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).

Identifier: RQ_002_6341
RFC Clause: 3.8
Type: Mandatory
Applies to: Host

Requirement:

An Authentication Payload in an IKE packet MUST be constructed as follows:

```
Octet            Field
----------------------
1 to 4           IKE Generic Payload Header
5                Authentication Method
6 to 8           Reserved
7 to End         Authentication Data (as described in RFC 4306 section 2.15)
```

RFC Text:

The Authentication Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Auth Method   !                RESERVED                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                      Authentication Data                      ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Authentication Payload Format

o Auth Method (1 octet) - Specifies the method of authentication   used. Values defined are:

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]).

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

DSS Digital Signature (3) - Computed as specified in section 2.15 using a DSS private key (see [DSS]) over a SHA-1 hash.

The values 0 and 4-200 are reserved to IANA. The values 201-255 are available for private use.

o Authentication Data (variable length) - see section 2.15.

The payload type for the Authentication Payload is thirty nine (39)

Identifier: RQ_002_6342
RFC Clause: 3.8
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE packet containing an Authentication payload, it MUST set the Authentication Method field to one of the following values:

```
 Value           Authentication Method
 ------------------------------------
 0               Reserved for IANA
 1               RSA Digital Signature
 2               Shared Key Message Integrity Code
 3               DSS Digital Signature
 4 - 200         Reserved for IANA
 201 - 255       For private use
```

RFC Text:

The Authentication Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Auth Method   !                RESERVED                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                      Authentication Data                      ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

           Figure 14:  Authentication Payload Format
```

o Auth Method (1 octet) - Specifies the method of authentication   used. Values defined are:

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]).

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

DSS Digital Signature (3) - Computed as specified in section 2.15 using a DSS private key (see [DSS]) over a SHA-1 hash.

The values 0 and 4-200 are reserved to IANA. The values 201-255 are available for private use.

o Authentication Data (variable length) - see section 2.15.

The payload type for the Authentication Payload is thirty nine (39)

Identifier: RQ_002_6343
RFC Clause: 3.8
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing an Authentication Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Authentication Payload) to the value thirty-nine (39)

RFC Text:

The Authentication Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Auth Method   !                RESERVED                       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                      Authentication Data                      ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

           Figure 14:  Authentication Payload Format
```

o Auth Method (1 octet) - Specifies the method of authentication used. Values defined are:

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]).

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

DSS Digital Signature (3) - Computed as specified in section 2.15 using a DSS private key (see [DSS]) over a SHA-1 hash.

The values 0 and 4-200 are reserved to IANA. The values 201-255 are available for private use.

o Authentication Data (variable length) - see section 2.15.

The payload type for the Authentication Payload is thirty nine (39)

Identifier: RQ_002_6344
RFC Clause: 3.9
Type: Mandatory
Applies to: Host

Requirement:
A Nonce Payload in an IKE packet MUST be constructed as follows:

```
  Octet           Field
  ----------------------
  1 to 4          IKE Generic Payload Header
  5 to End        Nonce Data
```

RFC Text:
The Nonce Payload is defined as follows:

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !C!  RESERVED   !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~                            Nonce Data                         ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: Nonce Payload Format

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

The payload type for the Nonce Payload is forty (40).

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

----

Identifier: RQ_002_6345
RFC Clause: 3.9
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing a Nonce payload, it MUST set the Nonce Data field to a random (but previously unused within the context of the current Security Association) value of between 16 and 256 octets in length

RFC Text:
The Nonce Payload is defined as follows:

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !C!  RESERVED   !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~                            Nonce Data                         ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: Nonce Payload Format

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

The payload type for the Nonce Payload is forty (40).

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

----

Identifier: RQ_002_6346
RFC Clause: 3.9
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Nonce Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Nonce Payload) to the value forty (40)

RFC Text:
The Nonce Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                            Nonce Data                         ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: Nonce Payload Format

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

The payload type for the Nonce Payload is forty (40).

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

----

Identifier: RQ_002_6347
RFC Clause: 3.10
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Notify Payload in an IKE response message (to specify why the associated IKE request was rejected), in an IKE 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)

RFC Text:
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.

Identifier: RQ_002_6348
RFC Clause: 3.10
Type: Mandatory
Applies to: Host

Requirement:
A Notify Payload in an IKE packet MUST be constructed as follows:

```
Octet                      Field
-------------------------------
1 to 4                     IKE Generic Payload Header
5                          Protocol Identifier
6                          Security Parameter Index (SPI) Size
7 - 8                      Notify Message Type
9 to SPI length + 9        Security Parameter Index (SPI)
SPI length + 10 to End     Notification Length
```

RFC Text:
The Notify Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                Security Parameter Index (SPI)                 ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Notification Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Notify Payload Format

o Protocol ID (1 octet) - If this notification concerns an existing SA, this field indicates the type of that SA. For IKE_SA notifications, this field MUST be one (1). For notifications concerning IPsec SAs this field MUST contain either (2) to indicate AH or (3) to indicate ESP. For notifications that do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt. All other values for this field are reserved to IANA for future assignment.

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.

o Notify Message Type (2 octets) - Specifies the type of notification message.

o SPI (variable length) - Security Parameter Index.

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 below).

The payload type for the Notify Payload is forty one (41).

Identifier: RQ_002_6349
RFC Clause: 3.10
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE packet containing a Notify Payload, it MUST set the Protocol Identifier field to one of the following values:

```
Protocol ID      Protocol
-------------------------
0                Not related to an existing SA
1                IKE (related to an IKE_SA)
2                AH (related to an IPsec SA)
3                ESP (related to an IPsec SA)
4 to 255         Reserved for IANA
```

RFC Text:

The Notify Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                Security Parameter Index (SPI)                 ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Notification Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

         Figure 16:  Notify Payload Format
```

o Protocol ID (1 octet) - If this notification concerns an existing SA, this field indicates the type of that SA. For IKE_SA notifications, this field MUST be one (1). For notifications concerning IPsec SAs this field MUST contain either (2) to indicate AH or (3) to indicate ESP. For notifications that do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt. All other values for this field are reserved to IANA for future assignment.

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.

o Notify Message Type (2 octets) - Specifies the type of notification message.

o SPI (variable length) - Security Parameter Index.

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 below).

The payload type for the Notify Payload is forty one (41).

Identifier: RQ_002_6350
RFC Clause: 3.10
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing a Notify Payload, it MUST set the SPI Size field to the length in octets of the Security Parameter Index if an SPI is applicable

RFC Text:
The Notify Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                Security Parameter Index (SPI)                 ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Notification Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Notify Payload Format

o Protocol ID (1 octet) - If this notification concerns an existing SA, this field indicates the type of that SA. For IKE_SA notifications, this field MUST be one (1). For notifications concerning IPsec SAs this field MUST contain either (2) to indicate AH or (3) to indicate ESP. For notifications that do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt. All other values for this field are reserved to IANA for future assignment.

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.

o Notify Message Type (2 octets) - Specifies the type of notification message.

o SPI (variable length) - Security Parameter Index.

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 below).

The payload type for the Notify Payload is forty one (41).

Identifier: RQ_002_6351
RFC Clause: 3.10
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing a Notify Payload, it MUST set the SPI Size field to zero if no SPI is applicable

RFC Text:
The Notify Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                Security Parameter Index (SPI)                 ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Notification Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Notify Payload Format

o Protocol ID (1 octet) - If this notification concerns an existing SA, this field indicates the type of that SA. For IKE_SA notifications, this field MUST be one (1). For notifications concerning IPsec SAs this field MUST contain either (2) to indicate AH or (3) to indicate ESP. For notifications that do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt. All other values for this field are reserved to IANA for future assignment.

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.

o Notify Message Type (2 octets) - Specifies the type of notification message.

o SPI (variable length) - Security Parameter Index.

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 below).

The payload type for the Notify Payload is forty one (41).

Identifier: RQ_002_6352
RFC Clause: 3.10
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing a Notify Payload, it MUST set the SPI Size field to zero if the notification concerns the current IKE_SA

RFC Text:
The Notify Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                Security Parameter Index (SPI)                 ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Notification Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Notify Payload Format

o Protocol ID (1 octet) - If this notification concerns an existing SA, this field indicates the type of that SA. For IKE_SA notifications, this field MUST be one (1). For notifications concerning IPsec SAs this field MUST contain either (2) to indicate AH or (3) to indicate ESP. For notifications that do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt. All other values for this field are reserved to IANA for future assignment.

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.**

o Notify Message Type (2 octets) - Specifies the type of notification message.

o SPI (variable length) - Security Parameter Index.

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 below).

The payload type for the Notify Payload is forty one (41).

Identifier: RQ_002_6353
RFC Clause: 3.10
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing a Notify Payload with the Message Type set to INVALID_IKE_SPI or INVALID_SPI, it MUST set the Security Parameter Index (SPI) field to the value of the invalid SPI

RFC Text:
The Notify Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                Security Parameter Index (SPI)                 ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Notification Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Notify Payload Format

o Protocol ID (1 octet) - If this notification concerns an existing SA, this field indicates the type of that SA. For IKE_SA notifications, this field MUST be one (1). For notifications concerning IPsec SAs this field MUST contain either (2) to indicate AH or (3) to indicate ESP. For notifications that do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt. All other values for this field are reserved to IANA for future assignment.

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.

o Notify Message Type (2 octets) - Specifies the type of notification message.

o SPI (variable length) - Security Parameter Index.

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 below).

The payload type for the Notify Payload is forty one (41).

Identifier: RQ_002_6354
RFC Clause: 3.10
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Notify Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Notify Payload) to the value forty-one (41)

RFC Text:
The Notify Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Protocol ID  !   SPI Size    !      Notify Message Type      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                Security Parameter Index (SPI)                 ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                       Notification Data                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Notify Payload Format

o Protocol ID (1 octet) - If this notification concerns an existing SA, this field indicates the type of that SA. For IKE_SA notifications, this field MUST be one (1). For notifications concerning IPsec SAs this field MUST contain either (2) to indicate AH or (3) to indicate ESP. For notifications that do not relate to an existing SA, this field MUST be sent as zero and MUST be ignored on receipt. All other values for this field are reserved to IANA for future assignment.

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.

o Notify Message Type (2 octets) - Specifies the type of notification message.

o SPI (variable length) - Security Parameter Index.

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 below).

The payload type for the Notify Payload is forty one (41).

Identifier: RQ_002_6355
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Notify Payload, it MUST set the Notify Message Type field to one of the following values:

```
Value            Message Type
----------------------------
0                Reserved
1                UNSUPPORTED_CRITICAL_PAYLOAD
4                INVALID_IKE_SPI
5                INVALID_MAJOR_VERSION
7                INVALID_SYNTAX
9                INVALID_MESSAGE_ID
11               INVALID_SPI
14               NO_PROPOSAL_CHOSEN
17               INVALID_KE_PAYLOAD
24               AUTHENTICATION_FAILED
34               SINGLE_PAIR_REQUIRED
35               NO_ADDITIONAL_SAS
36               INTERNAL_ADDRESS_FAILURE
37               FAILED_CP_REQUIRED
38               TS_UNACCEPTABLE
39               INVALID_SELECTORS
40 - 8191        Reserved for IANA
8192 - 16383     Private Use Error Types
16384            INITIAL_CONTACT
16385            SET_WINDOW_SIZE
16386            ADDITIONAL_TS_POSSIBLE
16387            IPCOMP_SUPPORTED
16388            NAT_DETECTION_SOURCE_IP
16389            NAT_DETECTION_DESTINATION_IP
16390            COOKIE
16391            USE_TRANSPORT_MODE
16392            HTTP_CERT_LOOKUP_SUPPORTED
16393            REKEY_SA
16394            ESP_TFC_PADDING_NOT_SUPPORTED
16395            NON_FIRST_FRAGMENTS_ALSO
16396 - 40959    Reserved for IANA
40960 - 65535    Private Use Status Types
```

RFC Text:
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.

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

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

Identifier: RQ_002_6356
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE response containing a Notify Payload with the Notify Message Type field set to an Error Type value (between 0 and 16383) that it does not recognize, it MUST assume that the corresponding request has failed entirely

RFC Text:
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.

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

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

Identifier: RQ_002_6357
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE request containing a Notify Payload with the Notify Message Type field set to an Error Type value (between 0 and 16383) that it does not recognize, it MUST ignore the payload

RFC Text:
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.

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

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

Identifier: RQ_002_6358
RFC Clause: 3.10.1
Type: Recommended
Applies to: Host

Requirement:
When an IKE implementation receives an IKE request containing a Notify Payload with the Notify Message Type field set to an Error Type value (between 0 and 16383) that it does not recognize, it SHOULD log the receipt of the unrecognized payload

RFC Text:

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.

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

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

Identifier: RQ_002_6359
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation receives an IKE message containing a Notify Payload with the Notify Message Type field set to a Status Type value (between 16384 and 65535) that it does not recognize, it MUST ignore the payload

RFC Text:

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.

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

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

Identifier: RQ_002_6360
RFC Clause: 3.10.1
Type: Recommended
Applies to: Host

Requirement:

When an IKE implementation receives an IKE message containing a Notify Payload with the Notify Message Type field set to a Status Type value (between 16384 and 65535) that it does not recognize, it SHOULD log the receipt of the unrecognized payload

RFC Text:

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.

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

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

Identifier: RQ_002_6361
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Notify Payload with the Notify Message Type set to a Status Type value (16384 to 65535) in any IKE message

RFC Text:
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.

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

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

Identifier: RQ_002_6362
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message with the "Critical" flag set in the IKE Header but the payload type is not recognized, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to UNSUPPORTED_CRITICAL_PAYLOAD and the Notification Data field set to the received unrecognized Payload Type value.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ------------------------------           -----
   ....

   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.
   ....
```

Identifier: RQ_002_6363
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message with an unrecognized Security Parameter Index in the IKE_SA Responder's SPI field in the IKE Header, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to INVALID_IKE_SPI and the Security Parameter Index field set to the received unrecognized SPI value.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   INVALID_IKE_SPI                          4

       Indicates an IKE message was received with an unrecognized
       destination SPI.  This usually indicates that the recipient
       has rebooted and forgotten the existence of an IKE_SA.
   ....
```

Identifier: RQ_002_6364
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message with the Major Version field in the IKE Header set to a value that is not supported by the recipient, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to INVALID_MAJOR_VERSION.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   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.

   ....
```

Identifier: RQ_002_6365
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message with a field type, length or value which is incorrect or out of range in the IKE Header or payload, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to INVALID_SYNTAX.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   INVALID_SYNTAX                            7

       Indicates the IKE message that was received was invalid
       because some type, length, or value was out of range or
       because the request was rejected for policy reasons.  To
       avoid a 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.  To
       avoid leaking information to someone probing a node, this
       status MUST be sent in response to any error not covered by
       one of the other status types.  To aid debugging, more
       detailed error information SHOULD be written to a console or
       log.

   ....
```

Identifier: RQ_002_6366
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message which cannot be processed for a reason that does not relate to any of the other predefined IKE Notify Message Error Type values, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to INVALID_SYNTAX.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   INVALID_SYNTAX                            7

       Indicates the IKE message that was received was invalid
       because some type, length, or value was out of range or
       because the request was rejected for policy reasons.  To
       avoid a 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.  To
       avoid leaking information to someone probing a node, this
       status MUST be sent in response to any error not covered by
       one of the other status types.  To aid debugging, more
       detailed error information SHOULD be written to a console or
       log.

   ....
```

Identifier: RQ_002_6367
RFC Clause: 3.10.1
Type: Recommended
Applies to: Host

Requirement:

```
When an IKE implementation receives an IKE message which cannot be processed for a reason that does
not relate to any of the predefined IKE Notify Message Error Type values, it SHOULD either record
details of the failure in a log or send these details to an operator's console.
```

RFC Text:

```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------            -----
   ....

   INVALID_SYNTAX                              7

       Indicates the IKE message that was received was invalid
       because some type, length, or value was out of range or
       because the request was rejected for policy reasons.  To
       avoid a 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.  To
       avoid leaking information to someone probing a node, this
       status MUST be sent in response to any error not covered by
       one of the other status types.  To aid debugging, more
       detailed error information SHOULD be written to a console or
       log.

   ....

--------------------
```

Identifier: RQ_002_6368
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:

```
When an IKE implementation receives an encrypted IKE message which requires a Notify message with
the Error Type set to the value INVALID_SYNTAX to be sent to the originator, it MUST NOT send this
response in encrypted form if either the Message ID in the received IKE Header or the Integrity
Checksum Data in the corresponding Encrypted Payload is invalid.
```

RFC Text:

```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------            -----
   ....

   INVALID_SYNTAX                              7

       Indicates the IKE message that was received was invalid
       because some type, length, or value was out of range or
       because the request was rejected for policy reasons.  To
       avoid a 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.  To
       avoid leaking information to someone probing a node, this
       status MUST be sent in response to any error not covered by
       one of the other status types.  To aid debugging, more
       detailed error information SHOULD be written to a console or
       log.

   ....

--------------------
```

Identifier: RQ_002_6369
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message with the Message ID in the IKE Header set to a value which is outside the supported window of identifiers, it MAY initiate an IKE Informational Exchange containing a Notify Payload with the Notify Message Type field set to the value INVALID_MESSAGE_ID and the Notification Data field containing the invalid Message ID from the received message.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   -----------------------------           -----
   ....

   INVALID_MESSAGE_ID                          9

       Sent when an IKE message ID outside the supported window is
       received.  This Notify MUST NOT be sent in a response; the
       invalid request MUST NOT be acknowledged.  Instead, inform
       the other side by initiating an INFORMATIONAL exchange with
       Notification data containing the four octet invalid message
       ID.  Sending this notification is optional, and
       notifications of this type MUST be rate limited.

   ....
```

Identifier: RQ_002_6370
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message with the Message ID in the IKE Header set to a value which is outside the supported window of identifiers, it MUST NOT send an INVALID_MESSAGE_ID notification in an IKE response.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   -----------------------------           -----
   ....

   INVALID_MESSAGE_ID                          9

       Sent when an IKE message ID outside the supported window is
       received.  This Notify MUST NOT be sent in a response; the
       invalid request MUST NOT be acknowledged.  Instead, inform
       the other side by initiating an INFORMATIONAL exchange with
       Notification data containing the four octet invalid message
       ID.  Sending this notification is optional, and
       notifications of this type MUST be rate limited.

   ....
```

Identifier: RQ_002_6371
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation receives an ESP or AH packet with an invalid Security Parameter Index, it MAY initiate an IKE Informational Exchange containing a Notify Payload with the Notify Message Type field set to the value INVALID_SPI and the Notification Data field containing the invalid Security Parameter Index from the received packet.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   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.
       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).

   ....
```

Identifier: RQ_002_6372
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message containing a Security Association Payload but is unable to support any of the cryptographic suite proposals included in the Payload, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to NO_PROPOSAL_CHOSEN.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   NO_PROPOSAL_CHOSEN                      14

       None of the proposed crypto suites was acceptable.


   ....
```

Identifier: RQ_002_6373
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message containing a Key Exchange Payload but the D-H Group # field in the payload indicates a Diffie-Hellman Group Number different to the one selected by the implementation for this exchange, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to INVALID_KE_PAYLOAD and the Notification Data field containing the correct D-H Group #.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   -----------------------------           -----
   ....

   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.

   ....
```

Identifier: RQ_002_6374
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message containing an Authentication Payload but the implementation is unable to authenticate the other IKE end-point using the Authentication Data provided, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to AUTHENTICATION_FAILED.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   -----------------------------           -----
   ....

   AUTHENTICATION_FAILED                    24

       Sent in the response to an IKE_AUTH message when for some
       reason the authentication failed.  There is no associated
       data.

   ....
```

Identifier: RQ_002_6375
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives a CREATE_CHILD_SA request containing a Traffic Selector Payload with multiple Traffic Selectors (and, thus, multiple Starting and Ending Address pairs) but the implementation is only able to accept Traffic Selectors specifying a single pair of addresses, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to SINGLE_PAIR_REQUIRED.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   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.  The

   requestor is expected to respond by requesting an SA for only
   the specific traffic it is trying to forward.

   ....
```

Identifier: RQ_002_6376
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives a CREATE_CHILD_SA request but the implementation is unable to establish any further CHILD_SAs on the specified IKE_SA, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to NO_ADDITIONAL_SAS.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   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.  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.

   ....
```

Identifier: RQ_002_6377
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives a CREATE_CHILD_SA request but the implementation is unable to resolve an internal IPv6 addresses specified in the Configuration Payload, it MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to INTERNAL_ADDRESS_FAILURE.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   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.  If this
   error is generated within an IKE_AUTH exchange, no CHILD_SA will
   be created.

   ....
```

Identifier: RQ_002_6378
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation is configured to expect an IKE_AUTH request from a particular endpoint to include a Configuration Payload with the CFG Type set to CFG_REQUEST and the CFG_REQUEST is not included in the received IKE_AUTH request, the implementation MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to FAILED_CP_REQUIRED.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   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.

   ....
```

Identifier: RQ_002_6379
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE request containing one or more Traffic Selectors but it is unable to accept any of the supplied address-protocol-port combinations, the implementation MUST send an IKE response to the originator containing a Notify Payload with the Notify Message Type field set to TS_UNACCEPTABLE.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   TS_UNACCEPTABLE                          38

   Indicates that none of the addresses/protocols/ports in the
   supplied traffic selectors is acceptable.


   ....
```

Identifier: RQ_002_6380
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
If an IKE implementation receives an ESP or AH packet whose selectors do not match those of the SA on which it was delivered , the implementation MAY send an IKE_INFORMATIONAL exchange to the originator containing a Notify Payload with:

- the Notify Message Type field set to INVALID_SELECTORS;
- the Security Parameter Index field set to the SPI from the received AH or ESP packet; and
- the Notification Data field containing as much of the received AH or ESP packet as will fit into the IKE_INFORMATIONAL exchange message without making it exceed the minimum IPv6 MTU.

RFC Text:
```
NOTIFY MESSAGES - ERROR TYPES           Value
   ----------------------------           -----
   ....

   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).  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.


   ....
```

Identifier: RQ_002_6381
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation restarts following a system failure, it MAY include a Notify Payload with the Notify Message Type set to INITIAL_CONTACT in the IKE_SA request for the first Security Association to be established after the failure

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

   INITIAL_CONTACT                         16384

       This notification asserts that this IKE_SA is the only
       IKE_SA currently active between the authenticated
       identities.  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.  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).
    ....
```

Identifier: RQ_002_6382
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
If an IKE implementation is capable of processing multiple simultaneous IKE exchanges, it MAY include in the initial IKE_SA exchange a Notify Payload with the Notify Message Type set to the value SET_WINDOW_SIZE and the Notify Data field set to the number of simultaneous messages the implementation is able to process in a 4-octet big-endian representation.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

   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. 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. Window size is always
       one until the initial exchanges complete.
    ....
```

Identifier: RQ_002_6383
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation accepts a limited subset of the Traffic Selectors offered in a received Traffic Selector Payload but is able to handle one or more Traffic Selectors not specified in the originator's offer, it MAY include in the IKE response indicating the accepted Traffic Selectors an additional Notify Payload with the Notify Message Type set to the value ADDITIONAL_TS_POSSIBLE

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   ------------------------------           -----
   ....

   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.  It may be sent only as an additional
       payload in a message including accepted TSs.
    ....
```

Identifier: RQ_002_6384
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:

When an IKE implementation that is capable of using IP Compression (IPComp) sends an IKE CHILD_SA request, it MAY include in that message one or more Notify Payloads with the Notify Message Type set to the value IPCOMP_SUPPORTED and the Notify Data field containing a 2-octet IPComp Compression Parameter Index followed by a one-octet transform identifier (from the following list) and, optionally, additional transform-dependent attributes.

```
Transform ID         Value
-------------------------
Reserved             0
IPCOMP_OUI           1
IPCOMP_DEFLATE       2
IPCOMP_LZS           3
IPCOMP_LZJH          4
```

RFC Text:

```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

   IPCOMP_SUPPORTED                        16387
```

This notification may be included only in a message containing an SA payload negotiating a CHILD_SA and indicates a willingness by its sender to use IPComp on this SA. 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.

The transform IDs currently defined are:

NAME	NUMBER	DEFINED IN
RESERVED	**0**	
IPCOMP_OUI	**1**	
IPCOMP_DEFLATE	**2**	**RFC 2394**
IPCOMP_LZS	**3**	**RFC 2395**
IPCOMP_LZJH	**4**	**RFC 3051**

values 5-240 are reserved to IANA. Values 241-255 are for private use among mutually consenting parties.

....

Identifier: RQ_002_6385
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation that is capable of using IP Compression (IPComp) sends an IKE CHILD_SA response, it MAY include in that message only one Notify Payload with the Notify Message Type set to the value IPCOMP_SUPPORTED and the Notify Data field containing a 2-octet IPComp Compression Parameter Index followed by a one-octet transform identifier (from the following list) and, optionally, additional transform-dependent attributes:

```
Transform ID        Value
-------------------------
Reserved            0
IPCOMP_OUI          1
IPCOMP_DEFLATE      2
IPCOMP_LZS          3
IPCOMP_LZJH         4
```

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

   IPCOMP_SUPPORTED                        16387
```

This notification may be included only in a message containing an SA payload negotiating a CHILD_SA and indicates a willingness by its sender to use IPComp on this SA. 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.**

The transform IDs currently defined are:

NAME	**NUMBER**	**DEFINED IN**
RESERVED	**0**	
IPCOMP_OUI	**1**	
IPCOMP_DEFLATE	**2**	**RFC 2394**
IPCOMP_LZS	**3**	**RFC 2395**
IPCOMP_LZJH	**4**	**RFC 3051**

values 5-240 are reserved to IANA. Values 241-255 are for private use among mutually consenting parties.

....

Identifier: RQ_002_6386
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation sends a Notify Payload with the Status Type set to the value, NAT_DETECTION_SOURCE_IP, it MUST set the Notification Data field to the SHA-1 digest (hash) calculated from the Source IP Address inserted in the IPv6 header and the Port Number on which the packet was sent.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

      NAT_DETECTION_SOURCE_IP                  16388

       This notification is used by its recipient to determine
       whether the source is behind a NAT box.  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.  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.  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).
       Alternately, it MAY reject the connection attempt if NAT
       traversal is not supported.

    ....
```

Identifier: RQ_002_6387
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY send more than one a Notify Payload with the Status Type set to the value, NAT_DETECTION_SOURCE_IP in a single IKE message

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

      NAT_DETECTION_SOURCE_IP                  16388

       This notification is used by its recipient to determine
       whether the source is behind a NAT box. 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. 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. 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).
       Alternately, it MAY reject the connection attempt if NAT
       traversal is not supported.

    ....
```

Identifier: RQ_002_6388
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation sends a Notify Payload with the Status Type set to the value, NAT_DETECTION_DESTINATION_IP, it MUST set the Notification Data field to the SHA-1 digest (hash) calculated from the Initiator's SPI followed by the Responder's SPI, the Destination IP Address inserted in the IPv6 header and finally the Port Number to which the packet was sent.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

         NAT_DETECTION_DESTINATION_IP            16389

       This notification is used by its recipient to determine
       whether it is behind a NAT box.  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.  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).  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].  Alternately, it MAY reject the connection
       attempt if NAT traversal is not supported.

    ....
```

Identifier: RQ_002_6389
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Notify Payload with the Status Type set to the value, COOKIE in an IKE_SA_INIT response.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

         COOKIE                                  16390

       This notification MAY be included in an IKE_SA_INIT
       response.  It indicates that the request should be retried
       with a copy of this notification as the first payload.  This
       notification MUST be included in an IKE_SA_INIT request
       retry if a COOKIE notification was included in the initial
       response.  The data associated with this notification MUST
       be between 1 and 64 octets in length (inclusive).

    ....
```

Identifier: RQ_002_6390
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation receives an IKE_SA_INIT response which includes a Notify Payload with the Status Type set to the value, COOKIE, it MUST include a Notify Payload in a retry of the IKE_SA_INIT request with the Status Type set to COOKIE and the Notification Data field set to the value in the received Notify Payload

RFC Text:

```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

         COOKIE                                 16390

       This notification MAY be included in an IKE_SA_INIT
       response.  It indicates that the request should be retried
       with a copy of this notification as the first payload.  This
       notification MUST be included in an IKE_SA_INIT request
       retry if a COOKIE notification was included in the initial
       response.  The data associated with this notification MUST
       be between 1 and 64 octets in length (inclusive).

    ....
```

Identifier: RQ_002_6391
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation sends a Notify Payload with the Status Type set to the value, COOKIE, the value included in the Notify Data field MUST be between 1 (one) and 64 (sixty-four) octets in length.

RFC Text:

```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

         COOKIE                                 16390

       This notification MAY be included in an IKE_SA_INIT
       response. It indicates that the request should be retried
       with a copy of this notification as the first payload.  This
       notification MUST be included in an IKE_SA_INIT request
       retry if a COOKIE notification was included in the initial
       response. The data associated with this notification MUST
       be between 1 and 64 octets in length (inclusive).

    ....
```

Identifier: RQ_002_6392
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Notify Payload with the Status Type set to the value, USE_TRANSPORT_MODE in a CHILD_SA request.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

         USE_TRANSPORT_MODE                       16391

       This notification MAY be included in a request message that
       also includes an SA payload requesting a CHILD_SA.  It
       requests that the CHILD_SA use transport mode rather than
       tunnel mode for the SA created.  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. If this is
       unacceptable to the initiator, the initiator MUST delete the
       SA. Note: Except when using this option to negotiate
       transport mode, all CHILD_SAs will use tunnel mode.

       Note: The ECN decapsulation modifications specified in
       [RFC4301] MUST be performed for every tunnel mode SA created
       by IKEv2.

    ....
```

Identifier: RQ_002_6393
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives a CHILD_SA request containing a Notify Payload with the Status Type set to the value, USE_TRANSPORT_MODE, it MUST include an identical Notify Payload in its response if it is able to comply with the request to use transport mode rather than tunnel mode on the new CHILD_SA.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

         USE_TRANSPORT_MODE                       16391

       This notification MAY be included in a request message that
       also includes an SA payload requesting a CHILD_SA.  It
       requests that the CHILD_SA use transport mode rather than
       tunnel mode for the SA created.  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.  If this is
       unacceptable to the initiator, the initiator MUST delete the
       SA.  Note: Except when using this option to negotiate
       transport mode, all CHILD_SAs will use tunnel mode.

       Note: The ECN decapsulation modifications specified in
       [RFC4301] MUST be performed for every tunnel mode SA created
       by IKEv2.

    ....
```

Identifier: RQ_002_6394
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives a CHILD_SA request containing a Notify Payload with the Status Type set to the value, USE_TRANSPORT_MODE, it MUST NOT include an identical Notify Payload in its response if it is unable to comply with the request to use transport mode rather than tunnel mode on the new CHILD_SA.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   ------------------------------           -----
   ....

         USE_TRANSPORT_MODE                       16391

       This notification MAY be included in a request message that
       also includes an SA payload requesting a CHILD_SA.  It
       requests that the CHILD_SA use transport mode rather than
       tunnel mode for the SA created.  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.  If this is
       unacceptable to the initiator, the initiator MUST delete the
       SA.  Note: Except when using this option to negotiate
       transport mode, all CHILD_SAs will use tunnel mode.

       Note: The ECN decapsulation modifications specified in
       [RFC4301] MUST be performed for every tunnel mode SA created
       by IKEv2.

    ....
```

Identifier: RQ_002_6395
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation sends a CHILD_SA request containing a Notify Payload with the Status Type set to the value, USE_TRANSPORT_MODE, but receives a CHILD_SA response which does not include an identical Notify Payload, it MUST delete the Child Security Association.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   ------------------------------           -----
   ....

         USE_TRANSPORT_MODE                       16391

       This notification MAY be included in a request message that
       also includes an SA payload requesting a CHILD_SA.  It
       requests that the CHILD_SA use transport mode rather than
       tunnel mode for the SA created.  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.  If this is
       unacceptable to the initiator, the initiator MUST delete the
       SA.  Note: Except when using this option to negotiate
       transport mode, all CHILD_SAs will use tunnel mode.

       Note: The ECN decapsulation modifications specified in
       [RFC4301] MUST be performed for every tunnel mode SA created
       by IKEv2.

    ....
```

Identifier: RQ_002_6396
RFC Clause: 3.10.1
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Notify Payload with the Status Type set to the value, HTTP_CERT_LOOKUP_SUPPORTED in any IKE message that can include a Certificate Request Payload.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   ------------------------------           -----
   ....

         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).

    ....

--------------------
```

Identifier: RQ_002_6397
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST include a Notify Payload with the Status Type set to the value, REKEY_SA in a CREATE_CHILD_SA exchanger if the purpose of the exchange is to replace an existing ESP or AH Security Association with a new one.

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   ------------------------------           -----
   ....

         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.

    ....

--------------------
```

Identifier: RQ_002_6398
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation does not support Flow Confidentiality (TFC) padding, it MUST include a Notify Payload in an initial exchange with the Status Type set to the value ESP_TFC_PADDING_NOT_SUPPORT

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

         ESP_TFC_PADDING_NOT_SUPPORTED            16394
```

This notification asserts that the sending endpoint will NOT accept packets that contain Flow Confidentiality (TFC) padding.

....

Identifier: RQ_002_6399
RFC Clause: 3.10.1
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation supports stateful fragment checking for a tunnel-mode Security Association and that SA is to be used for the transmission of non-initial fragments, it MUST include a Notify Payload in the initial exchange with the Status Type set to NON_FIRST_FRAGMENTS_ONLY

RFC Text:
```
NOTIFY MESSAGES - STATUS TYPES           Value
   -----------------------------           -----
   ....

   NON_FIRST_FRAGMENTS_ALSO                  16395
```

Used for fragmentation control. See [RFC4301] for explanation.

....

Identifier: RQ_002_6400
RFC Clause: 1.2
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST include the following elements in an IKE_SA_INIT request:

```
   IKE Header
 + Initiator's Security Association Payload
 + Initiator's key Exchange Payload
 + Initiator's Nonce Payload
```

RFC Text:
The details of the contents of each payload are described in section 3. Payloads that may optionally appear will be shown in brackets, such as [CERTREQ], indicate that optionally a certificate request payload can be included.

The initial exchanges are as follows:

```
  Initiator                          Responder
 -----------                        -----------
  HDR, SAi1, KEi, Ni   -->
```

HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE_SA. The KE payload sends the initiator's Diffie-Hellman value. Ni is the initiator's nonce.

```
                       <--    HDR, SAr1, KEr, Nr, [CERTREQ]
```

Identifier: RQ_002_6401
RFC Clause: 1.2
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST include the following elements in an IKE_SA_INIT response:

```
   IKE Header
 + Responder's Security Association Payload
 + Responder's Key Exchange Payload
 + Responder's Nonce Payload
```

RFC Text:
The details of the contents of each payload are described in section 3. Payloads that may optionally appear will be shown in brackets, such as [CERTREQ], indicate that optionally a certificate request payload can be included.

The initial exchanges are as follows:

```
  Initiator                          Responder
 -----------                        -----------
  HDR, SAi1, KEi, Ni   -->
```

HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE_SA. The KE payload sends the initiator's Diffie-Hellman value. Ni is the initiator's nonce.

```
                       <--    HDR, SAr1, KEr, Nr, [CERTREQ]
```

Identifier: RQ_002_6402
RFC Clause: 1.2
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Certificate Request Payload in an IKE_SA_INIT response

RFC Text:
The details of the contents of each payload are described in section 3. Payloads that may optionally appear will be shown in brackets, such as [CERTREQ], indicate that optionally a certificate request payload can be included.

The initial exchanges are as follows:

```
  Initiator                          Responder
 -----------                        -----------
  HDR, SAi1, KEi, Ni   -->
```

HDR contains the Security Parameter Indexes (SPIs), version numbers, and flags of various sorts. The SAi1 payload states the cryptographic algorithms the initiator supports for the IKE_SA. The KE payload sends the initiator's Diffie-Hellman value. Ni is the initiator's nonce.

```
                       <--    HDR, SAr1, KEr, Nr, [CERTREQ]
```

Identifier: RQ_002_6403
RFC Clause: 1.2
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST include the following elements in an IKE_AUTH request:

```
    IKE Header
  + Initiator's Identification Payload
  + Authentication Payload
  + Initiator's Security Association Payload for the first Child SA
  + Initiator's Traffic Selector Payload
  + Responder's Traffic Selector Payload
```

RFC Text:
At this point in the negotiation, each party can generate SKEYSEED, from which all keys are derived for that IKE_SA. All but the headers of all the messages that follow are encrypted and integrity protected. The keys used for the encryption and integrity protection are derived from SKEYSEED and are known as SK_e (encryption) and SK_a (authentication, a.k.a. integrity protection). A separate SK_e and SK_a is computed for each direction. In addition to the keys SK_e and SK_a derived from the DH value for protection of the IKE_SA, another quantity SK_d is derived and used for derivation of further keying material for CHILD_SAs. The notation SK { ... } indicates that these payloads are encrypted and integrity protected using that direction's SK_e and SK_a.

```
  HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,]
             AUTH, SAi2, TSi, TSr}     -->
```

The initiator asserts its identity with the IDi payload, proves knowledge of the secret corresponding to IDi and integrity protects the contents of the first message using the AUTH payload (see section 2.15). It might also send its certificate(s) in CERT payload(s) and a list of its trust anchors in CERTREQ payload(s). If any CERT payloads are included, the first certificate provided MUST contain the public key used to verify the AUTH field. The optional payload IDr enables the initiator to specify which of the responder's identities it wants to talk to. This is useful when the machine on which the responder is running is hosting multiple identities at the same IP address. The initiator begins negotiation of a CHILD_SA using the SAi2 payload. The final fields (starting with SAi2) are described in the description of the CREATE_CHILD_SA exchange.

```
                              <--    HDR, SK {IDr, [CERT,] AUTH,
                                           SAr2, TSi, TSr}
```

The responder asserts its identity with the IDr payload, optionally sends one or more certificates (again with the certificate containing the public key used to verify AUTH listed first), authenticates its identity and protects the integrity of the second message with the AUTH payload, and completes negotiation of a CHILD_SA with the additional fields described below in the CREATE_CHILD_SA exchange.

Identifier: RQ_002_6404
RFC Clause: 1.2
Type: Optional
Applies to: Host

Requirement:

An IKE implementation MAY include any or all of the following elements in an IKE_AUTH request:

```
Certificate Payload
Certificate Request Payload
Responder's Identification Payload
```

RFC Text:

At this point in the negotiation, each party can generate SKEYSEED, from which all keys are derived for that IKE_SA. All but the headers of all the messages that follow are encrypted and integrity protected. The keys used for the encryption and integrity protection are derived from SKEYSEED and are known as SK_e (encryption) and SK_a (authentication, a.k.a. integrity protection). A separate SK_e and SK_a is computed for each direction. In addition to the keys SK_e and SK_a derived from the DH value for protection of the IKE_SA, another quantity SK_d is derived and used for derivation of further keying material for CHILD_SAs. The notation SK { ... } indicates that these payloads are encrypted and integrity protected using that direction's SK_e and SK_a.

```
HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,]
           AUTH, SAi2, TSi, TSr}     -->
```

The initiator asserts its identity with the IDi payload, proves knowledge of the secret corresponding to IDi and integrity protects the contents of the first message using the AUTH payload (see section 2.15). It might also send its certificate(s) in CERT payload(s) and a list of its trust anchors in CERTREQ payload(s). If any CERT payloads are included, the first certificate provided MUST contain the public key used to verify the AUTH field. The optional payload IDr enables the initiator to specify which of the responder's identities it wants to talk to. This is useful when the machine on which the responder is running is hosting multiple identities at the same IP address. The initiator begins negotiation of a CHILD_SA using the SAi2 payload. The final fields (starting with SAi2) are described in the description of the CREATE_CHILD_SA exchange.

```
                              <--    HDR, SK {IDr, [CERT,] AUTH,
                                           SAr2, TSi, TSr}
```

The responder asserts its identity with the IDr payload, optionally sends one or more certificates (again with the certificate containing the public key used to verify AUTH listed first), authenticates its identity and protects the integrity of the second message with the AUTH payload, and completes negotiation of a CHILD_SA with the additional fields described below in the CREATE_CHILD_SA exchange.

Identifier: RQ_002_6405
RFC Clause: 1.2
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST include the following elements in an IKE_AUTH response:

```
  IKE Header
+ Responder's Identification Payload
+ Authentication Payload
+ Responder's Security Association Payload for the first Child SA
+ Initiator's Traffic Selector Payload
+ Responder's Traffic Selector Payload
```

RFC Text:
At this point in the negotiation, each party can generate SKEYSEED, from which all keys are derived for that IKE_SA. All but the headers of all the messages that follow are encrypted and integrity protected. The keys used for the encryption and integrity protection are derived from SKEYSEED and are known as SK_e (encryption) and SK_a (authentication, a.k.a. integrity protection). A separate SK_e and SK_a is computed for each direction. In addition to the keys SK_e and SK_a derived from the DH value for protection of the IKE_SA, another quantity SK_d is derived and used for derivation of further keying material for CHILD_SAs. The notation SK { ... } indicates that these payloads are encrypted and integrity protected using that direction's SK_e and SK_a.

```
HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,]
           AUTH, SAi2, TSi, TSr}     -->
```

The initiator asserts its identity with the IDi payload, proves knowledge of the secret corresponding to IDi and integrity protects the contents of the first message using the AUTH payload (see section 2.15). It might also send its certificate(s) in CERT payload(s) and a list of its trust anchors in CERTREQ payload(s). If any CERT payloads are included, the first certificate provided MUST contain the public key used to verify the AUTH field. The optional payload IDr enables the initiator to specify which of the responder's identities it wants to talk to. This is useful when the machine on which the responder is running is hosting multiple identities at the same IP address. The initiator begins negotiation of a CHILD_SA using the SAi2 payload. The final fields (starting with SAi2) are described in the description of the CREATE_CHILD_SA exchange.

```
                              <--    HDR, SK {IDr, [CERT,] AUTH,
                                           SAr2, TSi, TSr}
```

The responder asserts its identity with the IDr payload, optionally sends one or more certificates (again with the certificate containing the public key used to verify AUTH listed first), authenticates its identity and protects the integrity of the second message with the AUTH payload, and completes negotiation of a CHILD_SA with the additional fields described below in the CREATE_CHILD_SA exchange.

Identifier: RQ_002_6406
RFC Clause: 1.2
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Certificate Payload in an IKE_AUTH response

RFC Text:
At this point in the negotiation, each party can generate SKEYSEED, from which all keys are derived for that IKE_SA. All but the headers of all the messages that follow are encrypted and integrity protected. The keys used for the encryption and integrity protection are derived from SKEYSEED and are known as SK_e (encryption) and SK_a (authentication, a.k.a. integrity protection). A separate SK_e and SK_a is computed for each direction. In addition to the keys SK_e and SK_a derived from the DH value for protection of the IKE_SA, another quantity SK_d is derived and used for derivation of further keying material for CHILD_SAs. The notation SK { ... } indicates that these payloads are encrypted and integrity protected using that direction's SK_e and SK_a.

```
HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,]
           AUTH, SAi2, TSi, TSr}     -->
```

The initiator asserts its identity with the IDi payload, proves knowledge of the secret corresponding to IDi and integrity protects the contents of the first message using the AUTH payload (see section 2.15). It might also send its certificate(s) in CERT payload(s) and a list of its trust anchors in CERTREQ payload(s). If any CERT payloads are included, the first certificate provided MUST contain the public key used to verify the AUTH field. The optional payload IDr enables the initiator to specify which of the responder's identities it wants to talk to. This is useful when the machine on which the responder is running is hosting multiple identities at the same IP address. The initiator begins negotiation of a CHILD_SA using the SAi2 payload. The final fields (starting with SAi2) are described in the description of the CREATE_CHILD_SA exchange.

```
<--    HDR, SK {IDr, [CERT,] AUTH,
              SAr2, TSi, TSr}
```

The responder asserts its identity with the IDr payload, optionally sends one or more certificates (again with the certificate containing the public key used to verify AUTH listed first), authenticates its identity and protects the integrity of the second message with the AUTH payload, and completes negotiation of a CHILD_SA with the additional fields described below in the CREATE_CHILD_SA exchange.
}{{}

Identifier: RQ_002_6407
RFC Clause: 1.3
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST include the following elements in a CREATE_CHILD_SA request:

```
    IKE Header
  + Initiator's Security Association Payload for the Child SA
  + Initiator's Nonce Payload
```

RFC Text:
A CHILD_SA is created by sending a CREATE_CHILD_SA request. 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. The keying material for the CHILD_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).

In the CHILD_SA created as part of the initial exchange, a second KE payload and nonce MUST NOT be sent. The nonces from the initial exchange are used in computing the keys for the CHILD_SA.

The CREATE_CHILD_SA request contains:

```
  Initiator                                  Responder
 -----------                                -----------
  HDR, SK {[N], SA, Ni, [KEi],
      [TSi, TSr]}             -->
```

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. If this CREATE_CHILD_SA exchange is rekeying an existing SA other than the IKE_SA, the leading N payload of type REKEY_SA MUST identify the SA being rekeyed. If this CREATE_CHILD_SA exchange is not rekeying an existing SA, the N payload MUST be omitted. If the SA offers include different Diffie-Hellman groups, KEi MUST be an element of the group the initiator expects the responder to accept. If it guesses wrong, the CREATE_CHILD_SA exchange will fail, and it will have to retry with a different KEi.

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.

The CREATE_CHILD_SA response contains:

```
                             <--    HDR, SK {SA, Nr, [KEr],
                                          [TSi, TSr]}
```

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. If the responder chooses a cryptographic suite with a different group, it MUST reject the request. The initiator SHOULD repeat the request, but now with a KEi payload from the group the responder selected.

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. Traffic selectors are omitted if this CREATE_CHILD_SA request is being used to change the key of the IKE_SA.

Identifier: RQ_002_6408
RFC Clause: 1.3
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include any or all of the following elements in a CREATE_CHILD_SA request:

```
    Notify Payload
    Initiator's Key Exchange Payload
    Initiator's and Responder's Traffic Selector Payloads
```

RFC Text:

A CHILD_SA is created by sending a CREATE_CHILD_SA request. 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. The keying material for the CHILD_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).

In the CHILD_SA created as part of the initial exchange, a second KE payload and nonce MUST NOT be sent. The nonces from the initial exchange are used in computing the keys for the CHILD_SA.

The CREATE_CHILD_SA request contains:

```
  Initiator                                 Responder
 -----------                               -----------
  HDR, SK {[N], SA, Ni, [KEi],
      [TSi, TSr]}             -->
```

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. If this CREATE_CHILD_SA exchange is rekeying an existing SA other than the IKE_SA, the leading N payload of type REKEY_SA MUST identify the SA being rekeyed. If this CREATE_CHILD_SA exchange is not rekeying an existing SA, the N payload MUST be omitted. If the SA offers include different Diffie-Hellman groups, KEi MUST be an element of the group the initiator expects the responder to accept. If it guesses wrong, the CREATE_CHILD_SA exchange will fail, and it will have to retry with a different KEi.

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.

The CREATE_CHILD_SA response contains:

```
                             <--    HDR, SK {SA, Nr, [KEr],
                                          [TSi, TSr]}
```

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. If the responder chooses a cryptographic suite with a different group, it MUST reject the request. The initiator SHOULD repeat the request, but now with a KEi payload from the group the responder selected.

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. Traffic selectors are omitted if this CREATE_CHILD_SA request is being used to change the key of the IKE_SA.

Identifier: RQ_002_6409
RFC Clause: 1.3
Type: Mandatory
Applies to: Host

Requirement:

An IKE implementation MUST include the following elements in a CREATE_CHILD_SA response:

```
    IKE Header
  + Responder's Security Association Payload for the Child SA
  + Responder's Nonce Payload
```

RFC Text:

A CHILD_SA is created by sending a CREATE_CHILD_SA request. 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. The keying material for the CHILD_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).

In the CHILD_SA created as part of the initial exchange, a second KE payload and nonce MUST NOT be sent. The nonces from the initial exchange are used in computing the keys for the CHILD_SA.

The CREATE_CHILD_SA request contains:

```
  Initiator                                 Responder
 -----------                               -----------
  HDR, SK {[N], SA, Ni, [KEi],
      [TSi, TSr]}             -->
```

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. If this CREATE_CHILD_SA exchange is rekeying an existing SA other than the IKE_SA, the leading N payload of type REKEY_SA MUST identify the SA being rekeyed. If this CREATE_CHILD_SA exchange is not rekeying an existing SA, the N payload MUST be omitted. If the SA offers include different Diffie-Hellman groups, KEi MUST be an element of the group the initiator expects the responder to accept. If it guesses wrong, the CREATE_CHILD_SA exchange will fail, and it will have to retry with a different KEi.

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.

The CREATE_CHILD_SA response contains:

```
                             <--    HDR, SK {SA, Nr, [KEr],
                                          [TSi, TSr]}
```

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. If the responder chooses a cryptographic suite with a different group, it MUST reject the request. The initiator SHOULD repeat the request, but now with a KEi payload from the group the responder selected.

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. Traffic selectors are omitted if this CREATE_CHILD_SA request is being used to change the key of the IKE_SA.

Identifier: RQ_002_6410
RFC Clause: 1.3
Type: Optional
Applies to: Host

Requirement:

An IKE implementation MAY include any or all of the following elements in a CREATE_CHILD_SA response:

```
    Notify Payload
    Responder's Key Exchange Payload
    Initiator's and Responder's Traffic Selector Payloads
```

RFC Text:

A CHILD_SA is created by sending a CREATE_CHILD_SA request. 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. The keying material for the CHILD_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).

In the CHILD_SA created as part of the initial exchange, a second KE payload and nonce MUST NOT be sent. The nonces from the initial exchange are used in computing the keys for the CHILD_SA.

The CREATE_CHILD_SA request contains:

```
  Initiator                                  Responder
 -----------                                -----------
  HDR, SK {[N], SA, Ni, [KEi],
      [TSi, TSr]}             -->
```

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. If this CREATE_CHILD_SA exchange is rekeying an existing SA other than the IKE_SA, the leading N payload of type REKEY_SA MUST identify the SA being rekeyed. If this CREATE_CHILD_SA exchange is not rekeying an existing SA, the N payload MUST be omitted. If the SA offers include different Diffie-Hellman groups, KEi MUST be an element of the group the initiator expects the responder to accept. If it guesses wrong, the CREATE_CHILD_SA exchange will fail, and it will have to retry with a different KEi.

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.

The CREATE_CHILD_SA response contains:

```
                             <--    HDR, SK {SA, Nr, [KEr],
                                          [TSi, TSr]}
```

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. If the responder chooses a cryptographic suite with a different group, it MUST reject the request. The initiator SHOULD repeat the request, but now with a KEi payload from the group the responder selected.

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. Traffic selectors are omitted if this CREATE_CHILD_SA request is being used to change the key of the IKE_SA.

Identifier: RQ_002_6411
RFC Clause: 1.4
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST include an IKE Header plus one or more of the following elements in an IKE INFORMATIONAL request:

```
Notify Payload
Delete Payload
Configuration Payload
```

RFC Text:
The INFORMATIONAL exchange is defined as:

```
 Initiator                         Responder
-----------                       -----------
 HDR, SK {[N,] [D,] [CP,] ...} -->
                             <-- HDR, SK {[N,] [D,] [CP], ...}
```

The processing of an INFORMATIONAL exchange is determined by its component payloads.

Identifier: RQ_002_6412
RFC Clause: 1.4
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST include an IKE Header plus one or more of the following elements in an IKE INFORMATIONAL response:

```
Notify Payload
Delete Payload
Configuration Payload
```

RFC Text:
The INFORMATIONAL exchange is defined as:

```
 Initiator                         Responder
-----------                       -----------
 HDR, SK {[N,] [D,] [CP,] ...} -->
                             <-- HDR, SK {[N,] [D,] [CP], ...}
```

The processing of an INFORMATIONAL exchange is determined by its component payloads.

Identifier: RQ_002_6413
RFC Clause: 3.11
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation sends a Delete Payload containing Multiple SPIs, each SPI MUST relate to the same security protocol.

RFC Text:
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. **It is possible to send multiple SPIs in a Delete payload; however, each SPI MUST be for the same protocol**. Mixing of protocol identifiers MUST NOT be performed in a Delete payload. It is permitted, however, to include multiple Delete payloads in a single INFORMATIONAL exchange where each Delete payload lists SPIs for a different protocol.

Identifier: RQ_002_6414
RFC Clause: 3.11
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY send more than one Delete Payloads in a single INFORMATIONAL exchange message.

RFC Text:
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. It is possible to send multiple SPIs in a Delete payload; however, each SPI MUST be for the same protocol. Mixing of protocol identifiers MUST NOT be performed in a Delete payload. **It is permitted, however, to include multiple Delete payloads in a single INFORMATIONAL exchange where each Delete payload lists SPIs for a different protocol**.

Identifier: RQ_002_6415
RFC Clause: 3.11
Type: Mandatory
Applies to: Host

Requirement:
A Delete Payload in an IKE packet MUST be constructed as follows:

Octet	Field
1 to 4	IKE Generic Payload Header
5	Protocol Identifier
6	Security Parameter Index (SPI) Size
7 - 8	Number of SPIs included in the Payload
9 to end	Security Parameter Index(es)

RFC Text:
The Delete Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Protocol ID   !   SPI Size    !           # of SPIs           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~               Security Parameter Index(es) (SPI)              ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: Delete Payload Format

o Protocol ID (1 octet) - Must be 1 for an IKE_SA, 2 for AH, or 3   for ESP.

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.

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.

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.

The payload type for the Delete Payload is forty two (42)

Identifier: RQ_002_6416
RFC Clause: 3.11
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing a Delete Payload, it MUST set the Protocol Identifier field to one of the following values:

```
 Protocol ID       Protocol
 --------------------------
 1                 IKE (related to an IKE_SA)
 2                 AH  (related to an IPsec SA)
 3                 ESP (related to an IPsec SA)
```

RFC Text:
The Delete Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Protocol ID   !   SPI Size    !           # of SPIs           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~               Security Parameter Index(es) (SPI)              ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

            Figure 17:  Delete Payload Format
```

o Protocol ID (1 octet) - Must be 1 for an IKE_SA, 2 for AH, or 3   for ESP.

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.

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.

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.

The payload type for the Delete Payload is forty two (42)

Identifier: RQ_002_6417
RFC Clause: 3.11
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing a Delete Payload, it MUST set the SPI Size field to the length in octets of the Security Parameter Index(es) included in the payload.

RFC Text:
The Delete Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Protocol ID   !   SPI Size    !           # of SPIs           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~               Security Parameter Index(es) (SPI)              ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: Delete Payload Format

o Protocol ID (1 octet) - Must be 1 for an IKE_SA, 2 for AH, or 3 for ESP.

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.

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.

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.

The payload type for the Delete Payload is forty two (42)

Identifier: RQ_002_6418
RFC Clause: 3.11
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE packet containing a Delete Payload, it MUST set the Number of SPIs field to the number of Security Parameter Indexes included later in the payload

RFC Text:
The Delete Payload is defined as follows:

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !C!  RESERVED   !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Protocol ID   !   SPI Size    !           # of SPIs           !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~               Security Parameter Index(es) (SPI)              ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: Delete Payload Format

o Protocol ID (1 octet) - Must be 1 for an IKE_SA, 2 for AH, or 3 for ESP.

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.

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.

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.

The payload type for the Delete Payload is forty two (42)

Identifier: RQ_002_6419
RFC Clause: 3.11
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE packet containing a Delete Payload, it MUST include an SPI in the Security Parameter Index(es) field for every Security Association to be deleted

RFC Text:

The Delete Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Protocol ID   !   SPI Size    !           # of SPIs           !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~               Security Parameter Index(es) (SPI)              ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: Delete Payload Format

o Protocol ID (1 octet) - Must be 1 for an IKE_SA, 2 for AH, or 3 for ESP.

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.

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.

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.

The payload type for the Delete Payload is forty two (42)

Identifier: RQ_002_6420
RFC Clause: 3.11
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Delete Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Delete Payload) to the value forty-two (42)

RFC Text:
The Delete Payload is defined as follows:

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !C!  RESERVED   !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Protocol ID   !   SPI Size    !           # of SPIs           !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~               Security Parameter Index(es) (SPI)              ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

             Figure 17:  Delete Payload Format
```

o Protocol ID (1 octet) - Must be 1 for an IKE_SA, 2 for AH, or 3   for ESP.

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.

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.

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.

The payload type for the Delete Payload is forty two (42)

Identifier: RQ_002_6421
RFC Clause: 3.12
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends a Vendor ID Payload, it MUST set the Critical flag in the Payload Header to zero (0)

RFC Text:
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.

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. **A Vendor ID payload MUST NOT change the interpretation of any information defined in this specification (i.e., the critical bit MUST be set to 0).** Multiple Vendor ID payloads MAY be sent. An implementation is NOT REQUIRED to send any Vendor ID payload at all.

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

----

Identifier: RQ_002_6422
RFC Clause: 3.12
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY send zero or more Vendor ID Payloads in any IKE message

RFC Text:
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.

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. A Vendor ID payload MUST NOT change the interpretation of any information defined in this specification (i.e., the critical bit MUST be set to 0). **Multiple Vendor ID payloads MAY be sent. An implementation is NOT REQUIRED to send any Vendor ID payload at all.**

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

----

Identifier: RQ_002_6423
RFC Clause: 3.12
Type: Mandatory
Applies to: Host

Requirement:
An IKE implementation MUST ignore any unfamiliar Vendor Identifier received in a Vendor ID Payload.

RFC Text:
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.

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. A Vendor ID payload MUST NOT change the interpretation of any information defined in this specification (i.e., the critical bit MUST be set to 0). Multiple Vendor ID payloads MAY be sent. An implementation is NOT REQUIRED to send any Vendor ID payload at all.

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**

Identifier: RQ_002_6424
RFC Clause: 3.12
Type: Mandatory
Applies to: Host

Requirement:
A Vendor ID Payload in an IKE packet MUST be constructed as follows:

```
  Octet              Field
  ------------------------
  1 to 4             IKE Generic Payload Header
  5 to end           Vendor Identifier
```

RFC Text:
The Vendor ID Payload fields are defined as follows:

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 ! Next Payload  !C!  RESERVED   !         Payload Length        !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !                                                               !
 ~                        Vendor ID (VID)                        ~
 !                                                               !
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 18: Vendor ID Payload Format

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.

The payload type for the Vendor ID Payload is forty three (43)

Identifier: RQ_002_6425
RFC Clause: 3.12
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends a Vendor ID Payload in an IKE message, it MUST set the Vendor ID field to an alpha-numeric value which uniquely identifies the manufacturer of the implementation.

RFC Text:
The Vendor ID Payload fields are defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                        Vendor ID (VID)                        ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 18: Vendor ID Payload Format

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.

The payload type for the Vendor ID Payload is forty three (43)

Identifier: RQ_002_6426
RFC Clause: 3.12
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Vendor ID Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Vendor ID Payload) to the value forty-three (43)

RFC Text:
The Vendor ID Payload fields are defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C!  RESERVED   !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                        Vendor ID (VID)                        ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 18: Vendor ID Payload Format

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.

The payload type for the Vendor ID Payload is forty three (43)

Identifier: RQ_002_6427
RFC Clause: 3.13
Type: Mandatory
Applies to: Host

Requirement:

A Traffic Selector Payload in an IKE packet MUST be constructed as follows:

```
Octet              Field
------------------------
1 to 4             IKE Generic Payload Header
5                  Number of Traffic Selectors included in this payload
6 to 8             Reserved
9 to end           Traffic Selector(s)
```

RFC Text:

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:**

```
                     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>                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: Traffic Selectors Payload Format

o Number of TSs (1 octet) - Number of traffic selectors being provided.

o RESERVED - This field MUST be sent as zero and MUST be ignored on receipt.

o Traffic Selectors (variable length) - One or more individual traffic selectors.

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

The payload type for the Traffic Selector payload is forty four (44) for addresses at the initiator's end of the SA and forty five (45) for addresses at the responder's end.

Identifier: RQ_002_6428
RFC Clause: 3.13
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends a Traffic Selector Payload in an IKE message, it MUST set the Number of TSs field to the number of Traffic Selectors included in the payload

RFC Text:

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:

```
                     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>                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: Traffic Selectors Payload Format

o Number of TSs (1 octet) - Number of traffic selectors being   provided.

o RESERVED - This field MUST be sent as zero and MUST be ignored on receipt.

o Traffic Selectors (variable length) - One or more individual traffic selectors.

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

The payload type for the Traffic Selector payload is forty four (44) for addresses at the initiator's end of the SA and forty five (45) for addresses at the responder's end.

Identifier: RQ_002_6429
RFC Clause: 3.13
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends a Traffic Selector Payload in an IKE message, it MUST include in the payload the exact number of Traffic Selector descriptors indicated by the Number of TSs field

RFC Text:
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:

```
                     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>                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 19:  Traffic Selectors Payload Format
```

o Number of TSs (1 octet) - Number of traffic selectors being   provided.

o RESERVED - This field MUST be sent as zero and MUST be ignored on
   receipt.

o Traffic Selectors (variable length) - One or more individual
traffic selectors.

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

The payload type for the Traffic Selector payload is forty four (44) for addresses at the initiator's end of the SA and forty five (45) for addresses at the responder's end.

Identifier: RQ_002_6430
RFC Clause: 3.13
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing an Initiator's Traffic Selector Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Traffic Selector Payload) to the value forty-four (44)

RFC Text:
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:

```
                     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>                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: Traffic Selectors Payload Format

o Number of TSs (1 octet) - Number of traffic selectors being   provided.

o RESERVED - This field MUST be sent as zero and MUST be ignored on receipt.

o Traffic Selectors (variable length) - One or more individual traffic selectors.

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

The payload type for the Traffic Selector payload is forty four (44) for addresses at the initiator's end of the SA and forty five (45) for addresses at the responder's end.

Identifier: RQ_002_6431
RFC Clause: 3.13
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing an Responder's Traffic Selector Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Traffic Selector Payload) to the value forty-five (45)

RFC Text:
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:

```
                     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>                     ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: Traffic Selectors Payload Format

o Number of TSs (1 octet) - Number of traffic selectors being provided.

o RESERVED - This field MUST be sent as zero and MUST be ignored on receipt.

o Traffic Selectors (variable length) - One or more individual traffic selectors.

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

The payload type for the Traffic Selector payload is forty four (44) for addresses at the initiator's end of the SA **and forty five (45) for addresses at the responder's end.**

Identifier: RQ_002_6432
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:
A Traffic Selector Substructure in an IKE Traffic Selector Payload MUST be constructed in the following format:

```
Octet             Field
-----------------------
1                 Traffic Selector Type
2                 IP Protocol Identifier
3 & 4             Traffic Selector Length
5 & 6             Start Port
7 & 8             End Port
9 to 24           Starting IPv6 Address
24 to 40          Ending IPv6 Address
```

RFC Text:
Traffic Selector

```
                    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*                       ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 20: Traffic Selector

* 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.

Identifier: RQ_002_6433
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the TS Type field in each Traffic Selector Substructure to the value 8 (TS_IPV6_ADDR_RANGE)

RFC Text:

```
o  TS Type (one octet) - Specifies the type of traffic selector.

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.

o  Selector Length - Specifies the length of this Traffic Selector
   Substructure including the header

- - -
- - -


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

 TS Type                            Value
 -------                            -----
 RESERVED                            0-6

 TS_IPV4_ADDR_RANGE                   7

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

 TS_IPV6_ADDR_RANGE                   8

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

 RESERVED TO IANA                     9-240
 PRIVATE USE                          241-255

--------------------
```

Identifier: RQ_002_6434
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:

When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the IP Protocol ID in each Traffic Selector substructure to one of the following values to indicate which IP protocol is used on the corresponding Security Association:

Value	IP Protocol
0	The SA can carry all protocols
1 to 255	Protocol Number as specified in IETF RFC 1700

RFC Text:

o TS Type (one octet) - Specifies the type of traffic selector.

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.

o Selector Length - Specifies the length of this Traffic Selector Substructure including the header

\- - -
\- - -

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

TS Type	Value
RESERVED	0-6
TS_IPV4_ADDR_RANGE	7
TS_IPV6_ADDR_RANGE	8
RESERVED TO IANA	9-240
PRIVATE USE	241-255

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

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

Identifier: RQ_002_6435
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the Selector Length field in each Traffic Selector substructure to the length in octets of the Traffic Selector Substructure (40 octets for IPv6)

RFC Text:
```
o  TS Type (one octet) - Specifies the type of traffic selector.


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.
```

o Selector Length - Specifies the length of this Traffic Selector Substructure including the header

```
- - -
- - -


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

 TS Type                          Value
 -------                          -----
 RESERVED                          0-6

 TS_IPV4_ADDR_RANGE                 7

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

 TS_IPV6_ADDR_RANGE                 8

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

 RESERVED TO IANA                   9-240
 PRIVATE USE                        241-255

--------------------
```

Identifier: RQ_002_6436
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:

When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the Start Port field in each Traffic Selector substructure to the lowest port number allowed by this Traffic Selector unless the associated IP Protocol does not define a port number or if all ports are allowed.

RFC Text:

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.

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.

o Starting Address - The smallest address included in this Traffic Selector (length determined by TS type).

o Ending Address - The largest address included in this Traffic Selector (length determined by TS type).

Systems that are complying with [RFC4301] that wish to indicate "ANY" ports MUST set the start port to 0 and the end port to 65535; note that according to [RFC4301], "ANY" includes "OPAQUE". Systems working with [RFC4301] that wish to indicate "OPAQUE" ports, but not "ANY" ports, MUST set the start port to 65535 and the end port to 0.

Identifier: RQ_002_6437
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the Start Port field in each Traffic Selector substructure to zero (0) either if the associated IP Protocol does not define a port number or if all ports are allowed.

RFC Text:

```
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.

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.

o  Starting Address - The smallest address included in this Traffic
   Selector (length determined by TS type).

o  Ending Address - The largest address included in this Traffic
   Selector (length determined by TS type).
```

Systems that are complying with [RFC4301] that wish to indicate "ANY" ports MUST set the start port to 0 and the end port to 65535; note that according to [RFC4301], "ANY" includes "OPAQUE". Systems working with [RFC4301] that wish to indicate "OPAQUE" ports, but not "ANY" ports, MUST set the start port to 65535 and the end port to 0.

Identifier: RQ_002_6438
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:

When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the End Port field in each Traffic Selector substructure to the highest port number allowed by this Traffic Selector unless the associated IP Protocol does not define a port number or if all ports are allowed.

RFC Text:

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.

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.

o Starting Address - The smallest address included in this Traffic Selector (length determined by TS type).

o Ending Address - The largest address included in this Traffic Selector (length determined by TS type).

Systems that are complying with [RFC4301] that wish to indicate "ANY" ports MUST set the start port to 0 and the end port to 65535; note that according to [RFC4301], "ANY" includes "OPAQUE". Systems working with [RFC4301] that wish to indicate "OPAQUE" ports, but not "ANY" ports, MUST set the start port to 65535 and the end port to 0.

Identifier: RQ_002_6439
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:

When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the End Port field in each Traffic Selector substructure to 65535 either if the associated IP Protocol does not define a port number or if all ports are allowed.

RFC Text:

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.

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.

o Starting Address - The smallest address included in this Traffic Selector (length determined by TS type).

o Ending Address - The largest address included in this Traffic Selector (length determined by TS type).

Systems that are complying with [RFC4301] that wish to indicate "ANY" ports MUST set the start port to 0 and the end port to 65535; note that according to [RFC4301], "ANY" includes "OPAQUE". Systems working with [RFC4301] that wish to indicate "OPAQUE" ports, but not "ANY" ports, MUST set the start port to 65535 and the end port to 0.

Identifier: RQ_002_6440
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the Starting Address field in each Traffic Selector substructure to the lowest IPv6 Address included in this Traffic Selector

RFC Text:
```
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.

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.
```

o Starting Address - The smallest address included in this Traffic Selector (length determined by TS type).

```
o  Ending Address - The largest address included in this Traffic
   Selector (length determined by TS type).
```

Systems that are complying with [RFC4301] that wish to indicate "ANY" ports MUST set the start port to 0 and the end port to 65535; note that according to [RFC4301], "ANY" includes "OPAQUE". Systems working with [RFC4301] that wish to indicate "OPAQUE" ports, but not "ANY" ports, MUST set the start port to 65535 and the end port to 0.

Identifier: RQ_002_6441
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:

When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the Ending Address field in each Traffic Selector substructure to the highest IPv6 Address included in this Traffic Selector

RFC Text:

```
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.

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.


o  Starting Address - The smallest address included in this Traffic
   Selector (length determined by TS type).
```

o Ending Address - The largest address included in this Traffic Selector (length determined by TS type).

Systems that are complying with [RFC4301] that wish to indicate "ANY" ports MUST set the start port to 0 and the end port to 65535; note that according to [RFC4301], "ANY" includes "OPAQUE". Systems working with [RFC4301] that wish to indicate "OPAQUE" ports, but not "ANY" ports, MUST set the start port to 65535 and the end port to 0.

Identifier: RQ_002_6442
RFC Clause: 3.13.1
Type: Mandatory
Applies to: Host

Requirement:
When sending a Traffic Selector Payload containing one or more Traffic Selector substructures, an IKE implementation MUST set the Start Port field in each Traffic Selector substructure to 65535 and the End Port field to 0 if ports are to be considered as "OPAQUE" rather than "ANY" according to the definitions in IETF RFC 4301.

RFC Text:
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.

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.

o Starting Address - The smallest address included in this Traffic Selector (length determined by TS type).

o Ending Address - The largest address included in this Traffic Selector (length determined by TS type).

Systems that are complying with [RFC4301] that wish to indicate "ANY" ports MUST set the start port to 0 and the end port to 65535; note that according to [RFC4301], "ANY" includes "OPAQUE". **Systems working with [RFC4301] that wish to indicate "OPAQUE" ports, but not "ANY" ports, MUST set the start port to 65535 and the end port to 0.**

Identifier: RQ_002_6443
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation sends an Encrypted Payload in an IKE message, that payload MUST be the last payload in the message

RFC Text:
The Encrypted Payload, denoted SK{...} or E in this memo, contains other payloads in encrypted form. **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.

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.

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.

Identifier: RQ_002_6444
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing an Encrypted Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Encrypted Payload) to the value forty-six (46)

RFC Text:

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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

- o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

- o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

- 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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

- o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

- o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

- o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

- o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6445
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:

An Encrypted Payload in an IKE packet MUST be constructed as follows:

```
Octet                                               Field
----------------------------------------------------------
1 to 4                                              IKE Generic Payload Header
5 to (4 + block length of encryption algorithm)     Initialization Vector
followed by (variable length)                       Encrypted IKE payload
followed by (between 0 and 255 octets)              Sender defined padding
followed by (1 octet)                               Pad Length
followed by (algorithm-dependent length)            Integrity Checksum Data
```

RFC Text:

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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

```
o  Pad Length is the length of the Padding field. The sender SHOULD
   set the Pad Length to the minimum value that makes the combination
   of the Payloads, the Padding, and the Pad Length a multiple of the
   block size, but the recipient MUST accept any length that results
   in proper alignment.  This field is encrypted with the negotiated
   cipher.

o  Integrity Checksum Data is the cryptographic checksum of the
   entire message starting with the Fixed IKE Header through the Pad
   Length.  The checksum MUST be computed over the encrypted message.
   Its length is determined by the integrity algorithm negotiated.

--------------------
```

Identifier: RQ_002_6446
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it MUST set the Next Payload field in the Generic Payload header (Octet 1) to the value indicating the type of the first encrypted payload as follows:

Next Payload Type	Notation	Value
RESERVED		1-32
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
RESERVED TO IANA		49-127
PRIVATE USE		128-255

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

- **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.**

- Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

- Initialization Vector - A randomly chosen value whose length is equal to the block length of the underlying encryption algorithm. Recipients MUST accept any value. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

- IKE Payloads are as specified earlier in this section. This field

```
   is encrypted with the negotiated cipher.

o  Padding MAY contain any value chosen by the sender, and MUST have
   a length that makes the combination of the Payloads, the Padding,
   and the Pad Length to be a multiple of the encryption block size.

o  Pad Length is the length of the Padding field. The sender SHOULD
   set the Pad Length to the minimum value that makes the combination
   of the Payloads, the Padding, and the Pad Length a multiple of the
   block size, but the recipient MUST accept any length that results
   in proper alignment.  This field is encrypted with the negotiated
   cipher.

o  Integrity Checksum Data is the cryptographic checksum of the
   entire message starting with the Fixed IKE Header through the Pad
   Length.  The checksum MUST be computed over the encrypted message.
   Its length is determined by the integrity algorithm negotiated.

--------------------
```

Identifier: RQ_002_6447
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it MUST set the Payload Length field to the total number of octets in the Payload Header, Initialization Vector field, the Encrypted IKE Payloads, the Padding, the Pad Length field and the Integrity Checksum Data field

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6448
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it MUST set the Initialization Vector field to a randomly chosen value whose length is equal to the block length of the underlying encryption algorithm.

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6449
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation receives an Encrypted Payload in an IKE message, it MUST accept any value set in the Initialization Vector field.

RFC Text:

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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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**. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6450
RFC Clause: 3.14
Type: Recommended
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it SHOULD select the value to be set in the Initialization Vector field either pseudo-randomly or use the final ciphertext block of the previous message sent.

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. **Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent**. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6451
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends Encrypted Payloads in a sequence of IKE messages, it MUST set values in the Initialization Vector fields that are not:

* the same in each message;
* an easily deducible sequence; and
* ciphertext from a previously received message

RFC Text:

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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. **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**.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6452
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it MUST insert the required IKE payloads encrypted with the negotiated cipher into the Encrypted IKE Payloads field.

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6453
RFC Clause: 3.14
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it MAY insert any value in the Padding field of the payload

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o **Padding MAY contain any value chosen by the sender**, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6454
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it MUST ensure that the combined length of the Encrypted IKE Payloads field, the Padding field and the Pad Length field is a multiple of the chosen encryption block size.

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 21:  Encrypted Payload Format
```

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, **and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size**.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6455
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it MUST set the Pad Length field to the number of octets included in the Padding field of the same payload

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

- o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

- o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

- 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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

- o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

- o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

- o **Pad Length is the length of the Padding field**. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

- o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6456
RFC Clause: 3.14
Type: Recommended
Applies to: Host

Requirement:

When an IKE implementation sends an Encrypted Payload in an IKE message, it SHOULD set the Pad Length field to the minimum value that makes the combination of the Encrypted IKE Payloads field, the Padding field, and the Pad Length a multiple of the selected encryption block size

RFC Text:

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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. **The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads**, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6457
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation receives an Encrypted Payload in an IKE message, it MUST accept any value in the Pad Length field that results in the combination of the Payloads, the Padding, and the Pad Length being a multiple of the selected encryption block size

RFC Text:

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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 21:  Encrypted Payload Format
```

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, **but the recipient MUST accept any length that results in proper alignment**. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6458
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an Encrypted Payload in an IKE message, it MUST encrypt the Padding field with the negotiated cipher.

RFC Text:

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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

- o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

- o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

- 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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

- o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

- o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size. **This field is encrypted with the negotiated cipher**

- o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

- o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6459
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it MUST encrypt the Pad Length field with the negotiated cipher.

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

- o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

- o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

- 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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

- o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

- o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

- o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. **This field is encrypted with the negotiated cipher**.

- o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6460
RFC Clause: 3.14
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an Encrypted Payload in an IKE message, it MUST insert the checksum calculated for the complete IKE message (from IKE Header through to the Pad Length field) into the Integrity Checksum Data field of the payload using the negotiated integrity algorithm.

RFC Text:
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:

```
                     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                    ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Encrypted Payload Format

o Next Payload - The payload type of the first embedded payload. Note that this is an exception in the standard header format, since the Encrypted payload is the last payload in the message and therefore the Next Payload field would normally be zero. But because the content of this payload is embedded payloads and there was no natural place to put the type of the first one, that type is placed here.

o Payload Length - Includes the lengths of the header, IV, Encrypted IKE Payloads, Padding, Pad Length, and Integrity Checksum Data.

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. Senders SHOULD either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. 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.

o IKE Payloads are as specified earlier in this section. This field is encrypted with the negotiated cipher.

o Padding MAY contain any value chosen by the sender, and MUST have a length that makes the combination of the Payloads, the Padding, and the Pad Length to be a multiple of the encryption block size.

o Pad Length is the length of the Padding field. The sender SHOULD set the Pad Length to the minimum value that makes the combination of the Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient MUST accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

o Integrity Checksum Data is the cryptographic checksum of the entire message starting with the Fixed IKE Header through the Pad Length. The checksum MUST be computed over the encrypted message. Its length is determined by the integrity algorithm negotiated.

Identifier: RQ_002_6461
RFC Clause: 3.15
Type: Optional
Applies to: Host

Requirement:
An IKE implementation MAY include a Configuration Payload with the CFG Type field set to either the value CFG_REQUEST (1) or the value CFG_SET (3) in any IKE request message.

RFC Text:
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.

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**. 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. 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

----

Identifier: RQ_002_6462
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE request containing a Configuration Payload with the CFG Type field set to the value CFG_REQUEST (1) and the implementation supports IKE Configuration Payloads, it MUST include in the corresponding IKE response message either:

* a Configuration Payload with the CFG Type field set to the value CFG_REPLY (2); or
* a Notify payload with an error type indicating why the request could not be honoured.

RFC Text:
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.

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. **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. 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

----

Identifier: RQ_002_6463
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE request containing a Configuration Payload with the CFG Type field set to the value CFG_SET (3) and the implementation supports IKE Configuration Payloads, it MUST include in the corresponding IKE response message either:

* a Configuration Payload with the CFG Type field set to the value CFG_ACK (4); or
* a Notify payload with an error type indicating why the request could not be honoured.

RFC Text:
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.

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. **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**. 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

----

Identifier: RQ_002_6464
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE response to its IKE request which contained a Configuration Payload and the response does not contain a corresponding Configuration Payload, it MUST conclude that the responder does not support IKE Configuration Payloads.

RFC Text:
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.

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. 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. 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**

Identifier: RQ_002_6465
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives an IKE request containing a Configuration Payload with the CFG Type field set to the value CFG_REQUEST (1), it MUST include a Configuration Payload in its associated response with the CFG Type field set to CFG_REPLY (2) and with a valid value set in each of the Configuration Attributes provided in the request.

RFC Text:
"CFG_REQUEST/CFG_REPLY" allows an IKE endpoint to request information from its peer. If an attribute in the CFG_REQUEST Configuration Payload is not zero-length, it is taken as a suggestion for that attribute. **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. Requestors MUST ignore returned attributes that they do not recognize.

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. 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.

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.

Identifier: RQ_002_6466
RFC Clause: 3.15
Type: Optional
Applies to: Host

Requirement:
If an IKE implementation receives an IKE request containing a Configuration Payload with the CFG Type field set to the value CFG_REQUEST (1), it MAY include additional Configuration Attributes not provided in the original request.

RFC Text:
"CFG_REQUEST/CFG_REPLY" allows an IKE endpoint to request information from its peer. If an attribute in the CFG_REQUEST Configuration Payload is not zero-length, it is taken as a suggestion for that attribute. 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**. Requestors MUST ignore returned attributes that they do not recognize.

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. 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.

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.

Identifier: RQ_002_6467
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:
If an IKE implementation receives a solicited IKE response containing a Configuration Payload with the CFG Type field set to the value CFG_REPLY (2), it MUST ignore any Configuration Attributes that it does not recognize.

RFC Text:

"CFG_REQUEST/CFG_REPLY" allows an IKE endpoint to request information from its peer. If an attribute in the CFG_REQUEST Configuration Payload is not zero-length, it is taken as a suggestion for that attribute. 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. **Requestors MUST ignore returned attributes that they do not recognize.**

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. 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.

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.

Identifier: RQ_002_6468
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation receives an IKE request containing a Configuration Payload with the CFG Type field set to the value CFG_REQUEST (1) and it is unable support one or more of the Configuration Attribute Types included in the payload, it MUST either:

* include a Configuration Payload in its IKE response with the CFG Type set to CFG_REPLY (2) and with the unsupported Configuration Attributes empty; or
* send the appropriate IKE response without any Configuration Payload included.

RFC Text:

"CFG_REQUEST/CFG_REPLY" allows an IKE endpoint to request information from its peer. If an attribute in the CFG_REQUEST Configuration Payload is not zero-length, it is taken as a suggestion for that attribute. 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. Requestors MUST ignore returned attributes that they do not recognize.

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. 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.

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.

Identifier: RQ_002_6469
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation receives an IKE request containing a Configuration Payload with the CFG Type field set to the value CFG_SET (3) and it is able to implement one or more of the requested configuration changes, it MUST include a Configuration Payload in its IKE response with the CFG Type field set to CFG_ACK (4) and only those Configuration Attributes that it is able to accept, each with its Length field set to zero (0)

RFC Text:

"CFG_SET/CFG_ACK" allows an IKE endpoint to push configuration data to its peer. 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 and it MUST contain the attributes that the responder accepted with zero-length data. Those attributes that it did not accept MUST NOT be in the CFG_ACK Configuration Payload**. If no attributes were accepted, the responder MUST return either an empty CFG_ACK payload or a response message without a CFG_ACK payload. 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. An minimal implementation of this specification MAY ignore CFG_SET payloads.

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. 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.

Identifier: RQ_002_6470
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:

If an IKE implementation receives an IKE request containing a Configuration Payload with the CFG Type field set to the value CFG_SET (3) and it is unable to implement any of the requested configuration changes, it MUST either

* send a Configuration Payload in its IKE response with the CFG Type field set to CFG_ACK (4) but containing no Configuration Attributes; or
* send the appropriate IKE response to the requestor without a Configuration Payload included.

RFC Text:

"CFG_SET/CFG_ACK" allows an IKE endpoint to push configuration data to its peer. 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 and it MUST contain the attributes that the responder accepted with zero-length data. Those attributes that it did not accept MUST NOT be in the CFG_ACK Configuration Payload. **If no attributes were accepted, the responder MUST return either an empty CFG_ACK payload or a response message without a CFG_ACK payload**. 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. An minimal implementation of this specification MAY ignore CFG_SET payloads.

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. 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.

Identifier: RQ_002_6471
RFC Clause: 3.15
Type: Optional
Applies to: Host

Requirement:

If an IKE implementation receives an IKE request containing a Configuration Payload with the CFG Type field set to the value CFG_SET (3), it MAY ignore the payload.

RFC Text:

"CFG_SET/CFG_ACK" allows an IKE endpoint to push configuration data to its peer. 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 and it MUST contain the attributes that the responder accepted with zero-length data. Those attributes that it did not accept MUST NOT be in the CFG_ACK Configuration Payload. If no attributes were accepted, the responder MUST return either an empty CFG_ACK payload or a response message without a CFG_ACK payload. 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. **An minimal implementation of this specification MAY ignore CFG_SET payloads**.

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. 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.

Identifier: RQ_002_6472
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:

A Configuration Payload in an IKE packet MUST be constructed as follows:

```
Octet            Field
-----------------------
1 to 4           IKE Generic Payload Header
5                CFG Type
6 to 8           Reserved
9 to end         Configuration Attributes
```

RFC Text:

The Configuration Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C! RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   CFG Type    !                    RESERVED                   !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Configuration Attributes                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 22: Configuration Payload Format

The payload type for the Configuration Payload is forty seven (47).

o CFG Type (1 octet) - The type of exchange represented by the Configuration Attributes.

CFG Type	Value
RESERVED	0
CFG_REQUEST	1
CFG_REPLY	2
CFG_SET	3
CFG_ACK	4

values 5-127 are reserved to IANA. Values 128-255 are for private use among mutually consenting parties.

o RESERVED (3 octets) - MUST be sent as zero; MUST be ignored on receipt.

o Configuration Attributes (variable length) - These are type length values specific to the Configuration Payload and are defined below. There may be zero or more Configuration Attributes in this payload.

Identifier: RQ_002_6473
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the Configuration Payload) to the value forty-seven (47)

RFC Text:

The Configuration Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C! RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   CFG Type    !                    RESERVED                   !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Configuration Attributes                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 22: Configuration Payload Format

The payload type for the Configuration Payload is forty seven (47).

o CFG Type (1 octet) - The type of exchange represented by the Configuration Attributes.

CFG Type	Value
RESERVED	0
CFG_REQUEST	1
CFG_REPLY	2
CFG_SET	3
CFG_ACK	4

values 5-127 are reserved to IANA. Values 128-255 are for private use among mutually consenting parties.

o RESERVED (3 octets) - MUST be sent as zero; MUST be ignored on receipt.

o Configuration Attributes (variable length) - These are type length values specific to the Configuration Payload and are defined below. There may be zero or more Configuration Attributes in this payload.

Identifier: RQ_002_6474
RFC Clause: 3.15
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing a Configuration Payload, it MUST set the CFG Type field in the payload to one of the following values:

Value	CFG Type
0	Reserved
1	CFG_REQUEST
2	CFG_REPLY
3	CFG_SET
4	CFG_ACK
5 to 127	Reserved for IANA
128 to 255	For Private Use

RFC Text:

The Configuration Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C! RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   CFG Type    !                    RESERVED                   !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Configuration Attributes                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 22: Configuration Payload Format

The payload type for the Configuration Payload is forty seven (47).

- **o CFG Type (1 octet) - The type of exchange represented by the Configuration Attributes.**

CFG Type	**Value**
RESERVED	**0**
CFG_REQUEST	**1**
CFG_REPLY	**2**
CFG_SET	**3**
CFG_ACK	**4**

values 5-127 are reserved to IANA. Values 128-255 are for private use among mutually consenting parties.

o RESERVED (3 octets) - MUST be sent as zero; MUST be ignored on receipt.

o Configuration Attributes (variable length) - These are type length values specific to the Configuration Payload and are defined below. There may be zero or more Configuration Attributes in this payload.

Identifier: RQ_002_6475
RFC Clause: 3.15
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload, it MAY include zero or more Configuration Attributes in the payload.

RFC Text:

The Configuration Payload is defined as follows:

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Next Payload  !C! RESERVED    !         Payload Length        !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   CFG Type    !                    RESERVED                   !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!                                                               !
~                   Configuration Attributes                    ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 22:  Configuration Payload Format
```

The payload type for the Configuration Payload is forty seven (47).

o CFG Type (1 octet) - The type of exchange represented by the Configuration Attributes.

```
        CFG Type       Value
        ===========    =====
        RESERVED         0
        CFG_REQUEST      1
        CFG_REPLY        2
        CFG_SET          3
        CFG_ACK          4
```

values 5-127 are reserved to IANA. Values 128-255 are for private use among mutually consenting parties.

o RESERVED (3 octets) - MUST be sent as zero; MUST be ignored on receipt.

o Configuration Attributes (variable length) - These are type length values specific to the Configuration Payload and are defined below. There may be zero or more Configuration Attributes in this payload.

Identifier: RQ_002_6476
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes one or more Configuration Attributes, each Configuration Attribute substructure MUST be constructed as follows:

```
Octet                   Field
----------------------------
1 (bit 0)               Reserved (set to binary zero)
1 (bits 1 to 7) & 2     Attribute Type
3 & 4                   Length
5 to end                Value
```

RFC Text:
Configuration Attributes

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!R|         Attribute Type      !            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                             Value                             ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 23: Configuration Attribute Format

o Reserved (1 bit) - This bit MUST be set to zero and MUST be ignored on receipt.

o Attribute Type (15 bits) - A unique identifier for each of the Configuration Attribute Types.

o Length (2 octets) - Length in octets of Value.

o Value (0 or more octets) - The variable-length value of this Configuration Attribute.

The following attribute types have been defined:

Attribute Type	Value	Multi-Valued	Length
RESERVED	0		
INTERNAL_IP4_ADDRESS	1	YES*	0 or 4 octets
INTERNAL_IP4_NETMASK	2	NO	0 or 4 octets
INTERNAL_IP4_DNS	3	YES	0 or 4 octets
INTERNAL_IP4_NBNS	4	YES	0 or 4 octets
INTERNAL_ADDRESS_EXPIRY	5	NO	0 or 4 octets
INTERNAL_IP4_DHCP	6	YES	0 or 4 octets
APPLICATION_VERSION	7	NO	0 or more
INTERNAL_IP6_ADDRESS	8	YES*	0 or 17 octets
RESERVED	9		
INTERNAL_IP6_DNS	10	YES	0 or 16 octets
INTERNAL_IP6_NBNS	11	YES	0 or 16 octets
INTERNAL_IP6_DHCP	12	YES	0 or 16 octets
INTERNAL_IP4_SUBNET	13	YES	0 or 8 octets
SUPPORTED_ATTRIBUTES	14	NO	Multiple of 2
INTERNAL_IP6_SUBNET	15	YES	17 octets

* These attributes may be multi-valued on return only if multiple values were requested.

Identifier: RQ_002_6477
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing a Configuration Payload which includes one or more Configuration Attributes, the Attribute Type field in each Configuration Attribute substructure MUST be set to one of the following values:

```
   Attribute Type              Value
   ----------------------------------
    RESERVED                     0
    INTERNAL_IP4_ADDRESS         1
    INTERNAL_IP4_NETMASK         2
    INTERNAL_IP4_DNS             3
    INTERNAL_IP4_NBNS            4
    INTERNAL_ADDRESS_EXPIRY      5
    INTERNAL_IP4_DHCP            6
    APPLICATION_VERSION          7
    INTERNAL_IP6_ADDRESS         8
    RESERVED                     9
    INTERNAL_IP6_DNS            10
    INTERNAL_IP6_NBNS           11
    INTERNAL_IP6_DHCP           12
    INTERNAL_IP4_SUBNET         13
    SUPPORTED_ATTRIBUTES        14
    INTERNAL_IP6_SUBNET         15
    Reserved for IANA           16 to 16383
    For Private Use          16384 to 32767
```

RFC Text:

Configuration Attributes

```
                      1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !R|         Attribute Type      !            Length             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 ~                             Value                             ~
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 23:  Configuration Attribute Format
```

o Reserved (1 bit) - This bit MUST be set to zero and MUST be ignored on receipt.

o **Attribute Type (15 bits) - A unique identifier for each of the Configuration Attribute Types**.

o Length (2 octets) - Length in octets of Value.

o Value (0 or more octets) - The variable-length value of this Configuration Attribute.

The following attribute types have been defined:

Attribute Type	Value	Multi-Valued	Length
RESERVED	**0**		
INTERNAL_IP4_ADDRESS	**1**	**YES***	**0 or 4 octets**
INTERNAL_IP4_NETMASK	**2**	**NO**	**0 or 4 octets**
INTERNAL_IP4_DNS	**3**	**YES**	**0 or 4 octets**
INTERNAL_IP4_NBNS	**4**	**YES**	**0 or 4 octets**
INTERNAL_ADDRESS_EXPIRY	**5**	**NO**	**0 or 4 octets**
INTERNAL_IP4_DHCP	**6**	**YES**	**0 or 4 octets**
APPLICATION_VERSION	**7**	**NO**	**0 or more**
INTERNAL_IP6_ADDRESS	**8**	**YES***	**0 or 17 octets**
RESERVED	**9**		
INTERNAL_IP6_DNS	**10**	**YES**	**0 or 16 octets**
INTERNAL_IP6_NBNS	**11**	**YES**	**0 or 16 octets**
INTERNAL_IP6_DHCP	**12**	**YES**	**0 or 16 octets**
INTERNAL_IP4_SUBNET	**13**	**YES**	**0 or 8 octets**

```
   SUPPORTED_ATTRIBUTES    14    NO    Multiple of 2
   INTERNAL_IP6_SUBNET     15    YES   17 octets

 * These attributes may be multi-valued on return only if multiple
 values were requested.
```

Identifier: RQ_002_6478
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing a Configuration Payload which includes one or more Configuration Attributes, the Length field in each Configuration Attribute substructure MUST be set to the length, in octets, of the Value field in the same substructure.

RFC Text:

Configuration Attributes

```
                     1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 !R|         Attribute Type      !            Length             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 ~                             Value                             ~
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 23:  Configuration Attribute Format

o  Reserved (1 bit) - This bit MUST be set to zero and MUST be
   ignored on receipt.

o  Attribute Type (15 bits) - A unique identifier for each of the
   Configuration Attribute Types.

o  Length (2 octets) - Length in octets of Value.

o  Value (0 or more octets) - The variable-length value of this
   Configuration Attribute.

The following attribute types have been defined:

                              Multi-
   Attribute Type        Value Valued Length
   ====================== ===== ====== ==================
    RESERVED               0
    INTERNAL_IP4_ADDRESS   1    YES*  0 or 4 octets
    INTERNAL_IP4_NETMASK   2    NO    0 or 4 octets
    INTERNAL_IP4_DNS       3    YES   0 or 4 octets
    INTERNAL_IP4_NBNS      4    YES   0 or 4 octets
    INTERNAL_ADDRESS_EXPIRY 5   NO    0 or 4 octets
    INTERNAL_IP4_DHCP      6    YES   0 or 4 octets
    APPLICATION_VERSION    7    NO    0 or more
    INTERNAL_IP6_ADDRESS   8    YES*  0 or 17 octets
    RESERVED               9
    INTERNAL_IP6_DNS      10    YES   0 or 16 octets
    INTERNAL_IP6_NBNS     11    YES   0 or 16 octets
    INTERNAL_IP6_DHCP     12    YES   0 or 16 octets
    INTERNAL_IP4_SUBNET   13    YES   0 or 8 octets
    SUPPORTED_ATTRIBUTES  14    NO    Multiple of 2
    INTERNAL_IP6_SUBNET   15    YES   17 octets

 * These attributes may be multi-valued on return only if multiple
 values were requested.
```

Identifier: RQ_002_6479
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes one or more Configuration Attributes, the Value field in each Configuration Attribute substructure MUST be set to a value which conforms with the contents of both the Content Type field and the Length field in the same substructure.

RFC Text:
Configuration Attributes

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!R|         Attribute Type      !            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                             Value                             ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 23: Configuration Attribute Format

o Reserved (1 bit) - This bit MUST be set to zero and MUST be ignored on receipt.

o Attribute Type (15 bits) - A unique identifier for each of the Configuration Attribute Types.

o Length (2 octets) - Length in octets of Value.

o Value (0 or more octets) - The variable-length value of this Configuration Attribute.

The following attribute types have been defined:

Attribute Type	Value	Multi-Valued	Length
RESERVED	0		
INTERNAL_IP4_ADDRESS	1	YES*	0 or 4 octets
INTERNAL_IP4_NETMASK	2	NO	0 or 4 octets
INTERNAL_IP4_DNS	3	YES	0 or 4 octets
INTERNAL_IP4_NBNS	4	YES	0 or 4 octets
INTERNAL_ADDRESS_EXPIRY	5	NO	0 or 4 octets
INTERNAL_IP4_DHCP	6	YES	0 or 4 octets
APPLICATION_VERSION	7	NO	0 or more
INTERNAL_IP6_ADDRESS	8	YES*	0 or 17 octets
RESERVED	9		
INTERNAL_IP6_DNS	10	YES	0 or 16 octets
INTERNAL_IP6_NBNS	11	YES	0 or 16 octets
INTERNAL_IP6_DHCP	12	YES	0 or 16 octets
INTERNAL_IP4_SUBNET	13	YES	0 or 8 octets
SUPPORTED_ATTRIBUTES	14	NO	Multiple of 2
INTERNAL_IP6_SUBNET	15	YES	17 octets

* These attributes may be multi-valued on return only if multiple values were requested.

Identifier: RQ_002_6480
RFC Clause: 3.15,1
Type: Optional
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing a Configuration Payload, it MAY include more than one Configuration Attribute substructure for each of the following Attribute Types:

```
INTERNAL_IP4_ADDRESS
INTERNAL_IP4_DNS
INTERNAL_IP4_NBNS
INTERNAL_IP4_DHCP
INTERNAL_IP6_ADDRESS
INTERNAL_IP6_DNS
INTERNAL_IP6_NBNS
INTERNAL_IP6_DHCP
INTERNAL_IP4_SUBNET
INTERNAL_IP6_SUBNET
```

RFC Text:

Configuration Attributes

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!R|         Attribute Type      !            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                             Value                             ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 23:  Configuration Attribute Format
```

o Reserved (1 bit) - This bit MUST be set to zero and MUST be ignored on receipt.

o Attribute Type (15 bits) - A unique identifier for each of the Configuration Attribute Types.

o Length (2 octets) - Length in octets of Value.

o Value (0 or more octets) - The variable-length value of this Configuration Attribute.

The following attribute types have been defined:

Attribute Type	Value	Multi-Valued	Length
RESERVED	**0**		
INTERNAL_IP4_ADDRESS	**1**	**YES***	**0 or 4 octets**
INTERNAL_IP4_NETMASK	**2**	**NO**	**0 or 4 octets**
INTERNAL_IP4_DNS	**3**	**YES**	**0 or 4 octets**
INTERNAL_IP4_NBNS	**4**	**YES**	**0 or 4 octets**
INTERNAL_ADDRESS_EXPIRY	**5**	**NO**	**0 or 4 octets**
INTERNAL_IP4_DHCP	**6**	**YES**	**0 or 4 octets**
APPLICATION_VERSION	**7**	**NO**	**0 or more**
INTERNAL_IP6_ADDRESS	**8**	**YES***	**0 or 17 octets**
RESERVED	**9**		
INTERNAL_IP6_DNS	**10**	**YES**	**0 or 16 octets**
INTERNAL_IP6_NBNS	**11**	**YES**	**0 or 16 octets**
INTERNAL_IP6_DHCP	**12**	**YES**	**0 or 16 octets**
INTERNAL_IP4_SUBNET	**13**	**YES**	**0 or 8 octets**
SUPPORTED_ATTRIBUTES	**14**	**NO**	**Multiple of 2**
INTERNAL_IP6_SUBNET	**15**	**YES**	**17 octets**

* These attributes may be multi-valued on return only if multiple values were requested.

Identifier: RQ_002_6481
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing a Configuration Payload, it MUST NOT include more than one Configuration Attribute substructure for each of the following Attribute Types:

```
INTERNAL_IP4_NETMASK
INTERNAL_ADDRESS_EXPIRY
APPLICATION_VERSION
SUPPORTED_ATTRIBUTES
```

RFC Text:

Configuration Attributes

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!R|         Attribute Type      !            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                             Value                             ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 23:  Configuration Attribute Format
```

o Reserved (1 bit) - This bit MUST be set to zero and MUST be ignored on receipt.

o Attribute Type (15 bits) - A unique identifier for each of the Configuration Attribute Types.

o Length (2 octets) - Length in octets of Value.

o Value (0 or more octets) - The variable-length value of this Configuration Attribute.

The following attribute types have been defined:

Attribute Type	Value	Multi-Valued	Length
RESERVED	0		
INTERNAL_IP4_ADDRESS	1	YES*	0 or 4 octets
INTERNAL_IP4_NETMASK	2	NO	0 or 4 octets
INTERNAL_IP4_DNS	3	YES	0 or 4 octets
INTERNAL_IP4_NBNS	4	YES	0 or 4 octets
INTERNAL_ADDRESS_EXPIRY	5	NO	0 or 4 octets
INTERNAL_IP4_DHCP	6	YES	0 or 4 octets
APPLICATION_VERSION	7	NO	0 or more
INTERNAL_IP6_ADDRESS	8	YES*	0 or 17 octets
RESERVED	9		
INTERNAL_IP6_DNS	10	YES	0 or 16 octets
INTERNAL_IP6_NBNS	11	YES	0 or 16 octets
INTERNAL_IP6_DHCP	12	YES	0 or 16 octets
INTERNAL_IP4_SUBNET	13	YES	0 or 8 octets
SUPPORTED_ATTRIBUTES	14	NO	Multiple of 2
INTERNAL_IP6_SUBNET	15	YES	17 octets

* These attributes may be multi-valued on return only if multiple values were requested.

Identifier: RQ_002_6482
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE message containing a Configuration Payload which includes multiple Configuration Attributes of the same Attribute Type, it MUST NOT include more Configuration Attributes of that Attribute Type in its response

RFC Text:
Configuration Attributes

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!R|         Attribute Type      !            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                             Value                             ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

          Figure 23:  Configuration Attribute Format
```

o Reserved (1 bit) - This bit MUST be set to zero and MUST be ignored on receipt.

o Attribute Type (15 bits) - A unique identifier for each of the Configuration Attribute Types.

o Length (2 octets) - Length in octets of Value.

o Value (0 or more octets) - The variable-length value of this Configuration Attribute.

The following attribute types have been defined:

Attribute Type	Value	Multi-Valued	Length
RESERVED	0		
INTERNAL_IP4_ADDRESS	1	YES*	0 or 4 octets
INTERNAL_IP4_NETMASK	2	NO	0 or 4 octets
INTERNAL_IP4_DNS	3	YES	0 or 4 octets
INTERNAL_IP4_NBNS	4	YES	0 or 4 octets
INTERNAL_ADDRESS_EXPIRY	5	NO	0 or 4 octets
INTERNAL_IP4_DHCP	6	YES	0 or 4 octets
APPLICATION_VERSION	7	NO	0 or more
INTERNAL_IP6_ADDRESS	8	YES*	0 or 17 octets
RESERVED	9		
INTERNAL_IP6_DNS	10	YES	0 or 16 octets
INTERNAL_IP6_NBNS	11	YES	0 or 16 octets
INTERNAL_IP6_DHCP	12	YES	0 or 16 octets
INTERNAL_IP4_SUBNET	13	YES	0 or 8 octets
SUPPORTED_ATTRIBUTES	14	NO	Multiple of 2
INTERNAL_IP6_SUBNET	15	YES	17 octets

*** These attributes may be multi-valued on return only if multiple values were requested.**

Identifier: RQ_002_6483
RFC Clause: 3.15.1 ¶
Type: Mandatory
Applies to: Host

Requirement:
In order to request information regarding the internal IPv4 address being used by the other endpoint in an IKE Security Association an IKE implementation MUST send an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP4_ADDRESS and the Value field set to zero

RFC Text:
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 requestor would like to reuse that address. 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. 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].

Identifier: RQ_002_6484
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:
In order to request information regarding the internal IPv6 address being used by the other endpoint in an IKE Security Association an IKE implementation MUST send an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP6_ADDRESS and the Value field set to zero

RFC Text:
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 requestor would like to reuse that address. 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. 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].

Identifier: RQ_002_6485
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:
In order to propose that the other endpoint in an IKE Security Association uses a particular internal IPv4 address, an IKE implementation MUST send an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP4_ADDRESS and the Value field set to valid IPv4 address

RFC Text:
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 requestor would like to reuse that address. 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. 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].

Identifier: RQ_002_6486
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:
In order to propose that the other endpoint in an IKE Security Association uses a particular internal IPv6 address, an IKE implementation MUST send an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP6_ADDRESS and the Value field set to valid IPv6 address

RFC Text:
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 requestor would like to reuse that address. 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. 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].

Identifier: RQ_002_6487
RFC Clause: 3.15.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP4_ADDRESS and the Value field set to valid IPv4 address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to INTERNAL_IP4_NETMASK and the Value field set to the internal network's netmask.

RFC Text:
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.

Identifier: RQ_002_6488
RFC Clause: 3.15.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP4_ADDRESS and the Value field set to valid IPv4 address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to INTERNAL_IP4_DNS and the Value field set to a valid IPv4 address representing the address of a DNS server within its network.

RFC Text:
o **INTERNAL_IP4_DNS, INTERNAL_IP6_DNS - Specifies an address of a DNS server within the network**. Multiple DNS servers MAY be requested. The responder MAY respond with zero or more DNS server attributes.

Identifier: RQ_002_6489
RFC Clause: 3.15.2
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP6_ADDRESS and the Value field set to valid IPv6 address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to INTERNAL_IP6_DNS and the Value field set to a valid IPv6 address representing the address of a DNS server within its network.

RFC Text:
o **INTERNAL_IP4_DNS, INTERNAL_IP6_DNS - Specifies an address of a DNS server within the network**. Multiple DNS servers MAY be requested. The responder MAY respond with zero or more DNS server attributes.

Identifier: RQ_002_6490
RFC Clause: 3.15.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP4_ADDRESS and the Value field set to valid IPv4 address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to INTERNAL_IP4_NBNS and the Value field set to a valid IPv4 address representing the address of NetBios Name Server within its network.

RFC Text:
o **INTERNAL_IP4_DNS**, INTERNAL_IP6_DNS - **Specifies an address of a DNS server within the network**. Multiple DNS servers MAY be requested. The responder MAY respond with zero or more DNS server attributes.

Identifier: RQ_002_6491
RFC Clause: 3.15.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to either INTERNAL_IP4_ADDRESS or INTERNAL_IP6_ADDRESS and the Value field set to valid IP address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to INTERNAL_ADDRESS_EXPIRY and the Value field set to an integer representing the number of seconds that the recipient can continue to use the associated internal IP address

RFC Text:
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. Only one of these attributes MAY be present in the reply.

Identifier: RQ_002_6492
RFC Clause: 3.15.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP4_ADDRESS and the Value field set to valid IP address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to INTERNAL_IP4_DHCP and the Value field set to a valid IPv4 address representing the address to which any internal DHCP requests should be sent

RFC Text:
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. The responder MAY respond with zero or more DHCP server attributes

----

Identifier: RQ_002_6493
RFC Clause: 3.15.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP6_ADDRESS and the Value field set to valid IP address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to INTERNAL_IP6_DHCP and the Value field set to a valid IPv6 address representing the address to which any internal DHCP requests should be sent

RFC Text:
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. The responder MAY respond with zero or more DHCP server attributes

----

Identifier: RQ_002_6494
RFC Clause: 3.15.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to either INTERNAL_IP4_ADDRESS or INTERNAL_IP6_ADDRESS and the Value field set to valid IP address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to APPLICATION_VERSION and the Value field set to a string of printable ASCII characters that is not NULL terminated

RFC Text:
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.

Identifier: RQ_002_6495
RFC Clause: 3.15.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP4_ADDRESSS and the Value field set to valid IPv4 address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to INTERNAL_IP4_SUBNET and the Value field set to a valid IPv4 address followed by its associated netmask

RFC Text:
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. The responder MAY respond with zero or more sub-network attributes.

Identifier: RQ_002_6496
RFC Clause: 3.15.2
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE request containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to SUPPORTED_ATTRIBUTES, it MUST set the Length field to zero and the Value field to zero-length

RFC Text:
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. 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.

Identifier: RQ_002_6497
RFC Clause: 3.15.1
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation receives an IKE request containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to SUPPORTED_ATTRIBUTES, it MUST send an IKE response with a Configuration Payload which includes a Configuration Attribute substructure with the Attribute Type set to SUPPORTED_ATTRIBUTES and the Value field containing the 2-octet identifiers of all of the configuration attributes it supports

RFC Text:
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. **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**.

Identifier: RQ_002_6498
RFC Clause: 3.15.1
Type: Optional
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing a Configuration Payload which includes a Configuration Attribute with the Attribute Type set to INTERNAL_IP6_ADDRESSS and the Value field set to valid IPv6 address, it MAY include an additional Configuration Attribute substructure in the Configuration Payload with the Attribute Type field set to INTERNAL_IP6_SUBNET and the Value field set to a valid IPv6 address followed by a one-octet prefix-length

RFC Text:
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. The responder MAY respond with zero or more sub-network attributes.**

Identifier: RQ_002_6499
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:
An Extensible Authentication Protocol (EAP) Payload in an IKE packet MUST be constructed as follows:

```
Octet           Field
---------------------
1 to 4          IKE Generic Payload Header
5 to end        EAP Message
```

RFC Text:
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.

```
                     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                             ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 24: EAP Payload Format

The payload type for an EAP Payload is forty eight (48).

Identifier: RQ_002_6500
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing an EAP Payload, it MUST set the appropriate Next Payload field (either in the IKE Header or in the Generic Header of the payload preceding the EAP Payload) to the value forty-eight (48)

RFC Text:
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.

```
                     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                             ~
!                                                               !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 24: EAP Payload Format

The payload type for an EAP Payload is forty eight (48).

Identifier: RQ_002_6501
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Messages, it MUST construct the EAP Message substructure as follows:

```
Octet            Field
----------------------
1                Code
2                Identifier
3 & 4            Length
5                Type
6 to end         Type Data
```

RFC Text:

```
                     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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!     Code      ! Identifier    !           Length              !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!     Type      ! Type_Data...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

                 Figure 25:  EAP Message Format
```

o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4).

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. In other messages, this field MAY be set to any value.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. In a Request (1) message, Type indicates the data being requested. 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:

  1 Identity
  2 Notification
  3 Nak (Response Only)
  4 MD5-Challenge
  5 One-Time Password (OTP)
  6 Generic Token Card

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6502
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Message, it MUST set the Code field in the EAP Message substructure to one of the following values, as defined in IETF RFC 3748:

```
Code             Value
----------------------
Request          1
Response         2
Success          3
Failure          4
```

RFC Text:

```
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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Code      ! Identifier    !           Length              !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Type      ! Type_Data...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

                 Figure 25:  EAP Message Format
```

o **Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4)**.

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. In other messages, this field MAY be set to any value.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. In a Request (1) message, Type indicates the data being requested. 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:

```
1  Identity
2  Notification
3  Nak (Response Only)
4  MD5-Challenge
5  One-Time Password (OTP)
6  Generic Token Card
```

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6503
RFC Clause: 3.16
Type: Optional
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Message with the Code field set to Request (1), it MAY set the Identifier field in the EAP Message substructure to any one-octet value

RFC Text:

```
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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Code      ! Identifier    !           Length              !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Type      ! Type_Data...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

                 Figure 25:  EAP Message Format
```

o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4).

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. **In other messages, this field MAY be set to any value**.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. In a Request (1) message, Type indicates the data being requested. 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:

    1 Identity
    2 Notification
    3 Nak (Response Only)
    4 MD5-Challenge
    5 One-Time Password (OTP)
    6 Generic Token Card

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6504
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Message with the Code field set to Response (2), it MUST set the Identifier field in the EAP Message substructure to match the Identifier field in the corresponding EAP request

RFC Text:

```
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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Code      ! Identifier    !           Length              !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Type      ! Type_Data...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

                 Figure 25:  EAP Message Format
```

o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4).

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**. In other messages, this field MAY be set to any value.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. In a Request (1) message, Type indicates the data being requested. 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:

```
  1  Identity
  2  Notification
  3  Nak (Response Only)
  4  MD5-Challenge
  5  One-Time Password (OTP)
  6  Generic Token Card
```

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6505
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Message, it MUST set the Length field in the EAP Message substructure to the length in octets of the EAP Message substructure

RFC Text:

```
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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Code      ! Identifier    !           Length              !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Type      ! Type_Data...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 25: EAP Message Format

o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4).

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. In other messages, this field MAY be set to any value.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. In a Request (1) message, Type indicates the data being requested. 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:

1 Identity
2 Notification
3 Nak (Response Only)
4 MD5-Challenge
5 One-Time Password (OTP)
6 Generic Token Card

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6506
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Message with the Code field set to Request (1), it MUST set the Type field in the EAP Message substructure to one of the following values as defined in IETF RFC3748:

Type	Value
Identity	1
Notification	2
MD5-Challenge	4
One-Time Password (OTP)	5
Generic Token Card	6

RFC Text:

```
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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Code      ! Identifier    !           Length              !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Type      ! Type_Data...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 25: EAP Message Format

o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4).

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. In other messages, this field MAY be set to any value.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. **In a Request (1) message, Type indicates the data being requeste**d. 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:

1 Identity
2 Notification
3 Nak (Response Only)
4 MD5-Challenge
5 One-Time Password (OTP)
6 Generic Token Card

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6507
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Message with the Code field set to Response (2), it MUST set the Type field in the EAP Message substructure either to the value Nak (3) or to the value set in the Type field of the associated EAP request.

RFC Text:

```
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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Code      ! Identifier    !           Length              !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Type      ! Type_Data...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

                 Figure 25:  EAP Message Format
```

o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4).

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. In other messages, this field MAY be set to any value.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. In a Request (1) message, Type indicates the data being requested. **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:

  1 Identity
  2 Notification
  3 Nak (Response Only)
  4 MD5-Challenge
  5 One-Time Password (OTP)
  6 Generic Token Card

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6508
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Message with the Code field set to either Success (3) or Failure (4), it MUST NOT include the Type field or the Type Data field in the EAP Message substructure

RFC Text:

```
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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Code      ! Identifier    !           Length              !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Type      ! Type_Data...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

                 Figure 25:  EAP Message Format
```

o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4).

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. In other messages, this field MAY be set to any value.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). **For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present**. In a Request (1) message, Type indicates the data being requested. 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:

   1 Identity
   2 Notification
   3 Nak (Response Only)
   4 MD5-Challenge
   5 One-Time Password (OTP)
   6 Generic Token Card

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6509
RFC Clause: 3.16
Type: Mandatory
Applies to: Host

Requirement:

When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Message with the Code field set to Response (2), it MUST set the Type Data field to a value relevant to the associated EAP request as defined in IETF RFC3748

RFC Text:

```
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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Code      ! Identifier    !           Length              !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Type      ! Type_Data...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

                 Figure 25:  EAP Message Format
```

o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4).

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. In other messages, this field MAY be set to any value.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. In a Request (1) message, Type indicates the data being requested. 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:

   1 Identity
   2 Notification
   3 Nak (Response Only)
   4 MD5-Challenge
   5 One-Time Password (OTP)
   6 Generic Token Card

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6510
RFC Clause: 3.16
Type: Recommended
Applies to: Host

Requirement:
When an IKE implementation sends an IKE message containing an EAP Payload which includes an EAP Message with the Code field set to Request (1), it SHOULD NOT set the Type field to the value 1 (Identity)

RFC Text:

```
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
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Code      ! Identifier    !           Length              !
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  !     Type      ! Type_Data...
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

                 Figure 25:  EAP Message Format
```

o Code (1 octet) indicates whether this message is a Request (1), Response (2), Success (3), or Failure (4).

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. In other messages, this field MAY be set to any value.

o Length (2 octets) is the length of the EAP message and MUST be four less than the Payload Length of the encapsulating payload.

o Type (1 octet) is present only if the Code field is Request (1) or Response (2). For other codes, the EAP message length MUST be four octets and the Type and Type_Data fields MUST NOT be present. In a Request (1) message, Type indicates the data being requested. 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:

    1 Identity
    2 Notification
    3 Nak (Response Only)
    4 MD5-Challenge
    5 One-Time Password (OTP)
    6 Generic Token Card

o Type_Data (Variable Length) varies with the Type of Request and the associated Response. For the documentation of the EAP methods, see [EAP].

Note that since IKE passes an indication of initiator identity in message 3 of the protocol, the responder SHOULD NOT send EAP Identity requests. The initiator SHOULD, however, respond to such requests if it receives them.

Identifier: RQ_002_6511
RFC Clause: 2.4.
Type: Mandatory
Applies to: Host

Requirement:
An endpoint in an established IKE Security Association MUST conclude that the other endpoint in the SA has failed when a cryptographically protected INITIAL_CONTACT notification is received on a different IKE_SA but to the same authenticated identity

RFC Text:
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). 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. 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. 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). 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.

4.6 Requirements extracted from RFC 2405

Identifier: RQ_002_7000
RFC Clause: 3
Type: Mandatory
Applies to: IPsec host

Requirement:
When using DES-CBC in IPsec ESP the explicit Initialization Vector (IV) of 8 octets (64 bits) MUST be a random value.

RFC Text:
DES-CBC requires an explicit Initialization Vector (IV) of 8 octets (64 bits). This IV immediately precedes the protected (encrypted) payload. **The IV MUST be a random value**.

History

Document history		
V1.1.1	December 2006	Publication