4

IPSec Mechanism

4.1. Review of IP protocols

The Internet Protocol (IP) is used by routers to transfer packets between the two hosts, the source and destination. To do this, the routers use the destination address of the IP header and consult a routing table, which provides an output interface depending on the destination IP address.

The IP packet is generated by the source. Its size is defined by the level 2 protocol used by the network to which the host is connected. For example, Ethernet limits the size of the packet or Maximum Transmission Unit (MTU) to 1,500 bytes.

The IP protocol provides an unreliable packet transfer service. The router neither acknowledges data received nor checks flows or errors in the IP packet.

4.1.1. *IPv4 protocol*

The header of the IPv4 protocol contains the following fields (Figure 4.1):

Version: this field, coded on 4 bits, includes the version number of the IP protocol. It has a value of 4.

Internet Header Length (*IHL*): this field, coded on 4 bits, includes the size of the IP header in multiples of 4 bytes. For an optionless header, this field has a value of 5.

Type of Service (*ToS*): this field, coded on 1 byte, was defined in the original version of the standard to implement priority management and give information on the level required for delay, output, reliability and cost.

The *ToS* field is replaced by the DiffServ Code Point (*DSCP*) field coded on 6 bits and the Explicit Congestion Notification (*ECN*) field coded on 2 bits.

DSCP marking is used to build service quality in IP networks. Routers use marking to select Per-Hop Behavior (PHB) corresponding to a specific process for data transfer.

The ECN mechanism relies on the use of fields in the IP and Transmission Control Protocol (TCP) protocol headers to alert first the destination station, and then the source station if congestion develops, causing reduced rate from the source.

Total Length: this field, coded on 2 bytes, contains the size of the packet, including the IPv4 header and encapsulated data. It authorizes a packet length of 65,535 bytes.

Such a length is impractical; however, it is generally limited by the level 2 protocol.

The router ensures the interconnection of networks, the connectivity function of which is ensured by technology different to IP, such as Ethernet for example. Therefore, it enables a change of networks; that is, a modification of the physical layer (level 1 layer) and the data link layer (level 2 layer).

The transfer executed by the router can send the packet onto a new network, the MTU value of which is less than the original one. In this case, the router executes a fragmentation of the IPv4 packet in order to adapt itself to the new MTU value. The reassembly of the fragments to recreate the original packet is done by the destination. The MTU has a minimum value of 576 bytes.

Identification: this field, coded on 2 bytes, contains a value enabling the reassembly on reception of the IP packet fragments.

Flags: this field contains three bits used in the following way:

– the first bit is always positioned at zero;

– the second Don't Fragment (DF) bit enables the authorization (bit at zero) or forbidding (bit atone) of fragmentation;

– the Third More Fragment (MF) bit identifies the last fragment (bit at zero) or intermediary fragments (bit at one).

Fragment Offset: this field, coded on 12 bits, indicates the positioning of the fragment in the initial packet. The measurement is done in multiples of 8 bytes. The value of the first fragment field is 0.

Time to Live (*TTL*): this field, coded on 1 byte, contains the maximum number of routers crossed by the packet. Each router crossed decreases the value of this field by one unit. When the value reaches 0, the packet is deleted, and an Internet Control Message Protocol (ICMPv4) error message is sent to the source.

Protocol: this field, coded on 1 byte, contains a value used to identify the type of data encapsulated by the IPv4 header (Table 4.1).

Protocol field	Data encapsulated
1	ICMPv4 (Internet Control Message Protocol) message
2	IGMP (Internet Group Management Protocol) message
4	IPv4 packet
6	TCP (transmission control protocol) segment
17	UDP (user datagram protocol) segment
41	IPv6 packet
50	ESP (encapsulating security payload) extension
51	AH (authentication header) extension
89	OSPFv2 (open shortest path first) message

Table 4.1. *Protocol field values*

Header Checksum: this field, coded on 2 bytes, contains a checksum calculated only on the header.

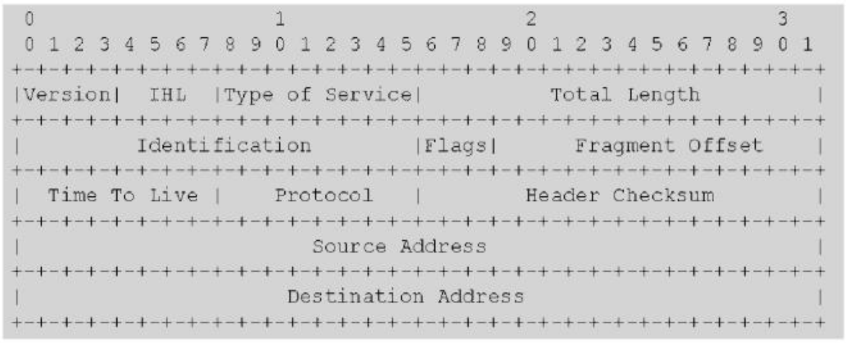
Source Address: this field, coded on 4 bytes, contains the IPv4 address of the packet source.

Destination Address: this field, coded on 4 bytes, contains the IPv4 address of the packet destination.

Theoretically, the IPv4 header can contain zero, one or several options. The options initially defined are no longer used. Conversely, a new option, *Router Alert*, was subsequently introduced to force the router to analyze encapsulated data, even though it is not the destination. This option is used in association with certain ReSerVation Protocol (RSVP) messages.

4.1.2. *IPv6 protocol*

The IP version 6 (IPv6) protocol is the new protocol intended to succeed the IPv4 protocol. The principal modification contributed by the IPv6 protocol concerns the size (16 bytes) allocated to the source and destination addresses, which results in a larger header (40 bytes).



```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4.1. *IPv4 header format*

The IPv6 protocol has introduced simplifications compared to the IPv4 protocol, enabling better performance with regard to packet processing by routers. The following fields have been removed from the IPv6 header:

– *IHL*, since the IPv6 header has a fixed length of 40 bytes;

– *Identification*, *Flags*, *Fragment Offset*, since fragmentation is the subject of an extension;

– *Header Checksum*, since transmission is considered to be of good quality, and binary errors are rare.

Options have been removed from the basic header and replaced by new headers called extensions. Other than the *Hop-by-Hop* option, which is processed by all intermediary routers, the other options are only taken into account by the destinations.

Fragmentation of the IPv4 packet is executed by the router, which is not the case in IPv6. When an IPv6 packet is sent, the source must discover the minimum MTU value. For applications that cannot segment the message, fragmentation of the IPv6 packet is executed by the source. This provision improves router performance.

The header of the IPv6 protocol contains the following fields (Figure 4.2):

Version: this field, coded on 4 bits, displays the version number of the IP protocol. It has a value of 6;

Traffic Class: this field, coded on 1 byte, is equivalent to the *DSCP* and *ECN* fields in the IPv4 header;

Flow Label: this field, coded on 20 bits, displays a unique number chosen by the source. Its purpose is to facilitate the work of routers to implement classification functions for a flow;

Payload Length: this field, coded on 2 bytes, displays the size of the data encapsulated by the IP header. It is different from the *Total Length* field in the IPv4 header, which indicates the packet size;

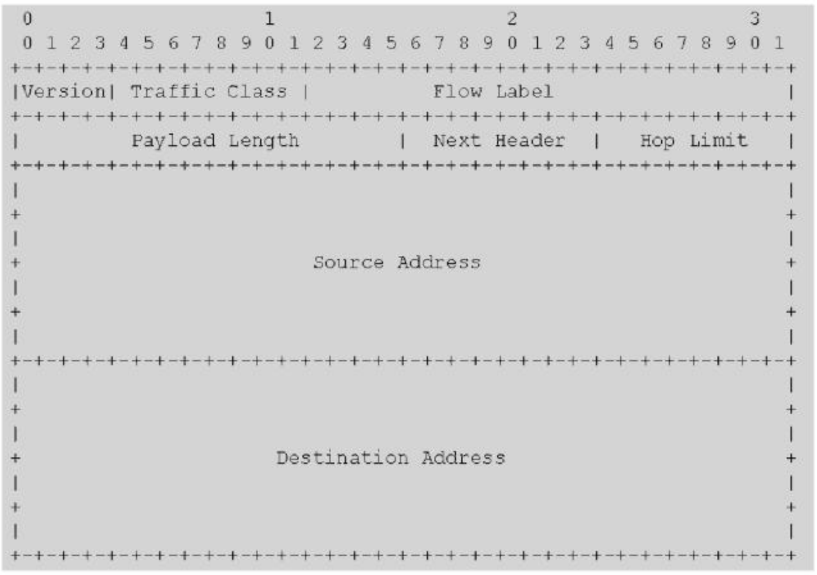


Figure 4.2. *IPv6 header format*

Next Header: this field, coded on 1 byte, displays the identifier of the next header of data encapsulated by the IP header. This field plays the same role as the *Protocol* field in the IPv4 header. New values for this field have been defined to take extensions into account (Table 4.2);

Next header field	Data encapsulated
0	*Hop-by-Hop* extension
4	IPv4 packet
6	TCP (transmission control protocol) segment
17	UDP (user datagram protocol) segment
41	IPv6 packet
43	*Routing* extension
44	*Fragment* extension
50	ESP (encapsulating security payload) extension
51	AH (authentication header) extension
58	ICMPv6 (Internet Control Message Protocol) message
59	End of extensions
60	*Destination* extension
89	OSPFv3 (open shortest path first) message

Table 4.2. *Next Header field values*

Hop Limit: this field, coded on 1 byte, is equivalent to the *TTL* field in the IPv4 header;

Source Address: this field, coded on 16 bytes, shows the IP address of the packet source;

Destination Address: this field, coded on 16 bytes, shows the IP address of the packet destination.

4.2. IPSec architecture

Security services (authentication, integrity and confidentiality) are offered in an identical way in IPv4 and IPv6. Their implementation is optional in IPv4 and mandatory in IPv6. Their use is optional.

The Internet Protocol Security (IPSec) mechanism can be used with LAN hosts or security gateways located at the interface between the LAN and WAN networks.

The first application consists of linking remote LAN networks via an untrusted WAN network. The IPSec mechanism is deployed in the security gateways of each LAN network. In this case, the two LAN networks are considered protected networks (Figure 4.3).

The second application consists of linking a host to a remote LAN network via a LAN network (the host's) and an untrusted WAN network. The IPSec mechanism is deployed in the host and a security gateway located in the remote LAN network, considered a protected network (Figure 4.3).

The third application consists of linking two hosts via two LAN networks and an untrusted WAN network. The IPSec mechanism is deployed from end-to-end, in each host (Figure 4.3).

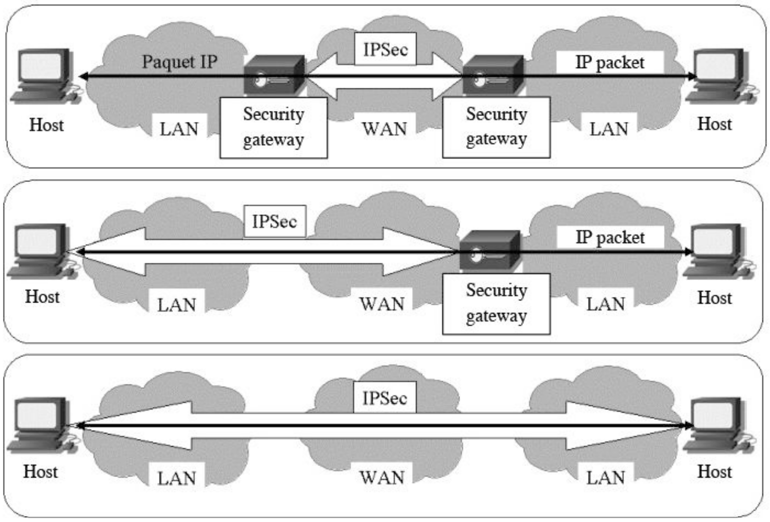


Figure 4.3. *IPSec architecture*

4.2.1. *Security headers*

The IPSec mechanism introduces two IPv4 or IPv6 header extensions:

– The Authentication Header (AH) is designed to ensure the integrity and authentication of IP packets without data encryption (no confidentiality).

– The Encapsulating Security Payload (ESP) ensures the integrity, authentication and confidentiality of IP packets.

4.2.1.1. *AH extension*

The AH extension offers authentication and data integrity services and enables protection against IP packet replay. The same AH extension is used in association with an IPv4 or IPv6 header. The presence of the extension is indicated by the *Next Header* (in IPv6) or *Protocol* (in IPv4) field of the previous header, with a value of 51.

In addition to the *Next Header* field, the AH extension contains the following fields (Figure 4.4):

Payload Length: this field, coded on 1 byte, provides the extension size in multiples of 4 bytes, not including the first 8 bytes. The size of the extension in IPv6 must remain a multiple of 8 bytes.

Security Parameters Index (*SPI*): this field, coded on 4 bytes, contains a value pertaining to the previously negotiated Security Association (SA).

Sequence Number: this field, coded on 4 bytes, contains a value increased by one unit for each IPv4 or IPv6 packet transmitted. This field enables protection against replay. This field has a value of 1 for the first packet transmitted. When the counter reaches the maximum value, a new SA must be negotiated in order to avoid the start of a new cycle.

An Extended Sequence Number (ESN) coded on 8 bytes constitutes an option making it possible for the lifetime of the SA to be prolonged. In order to preserve the structure of the extension, the 32 least significant bits are transmitted in the *Sequence Number* field. However, the data digest is calculated on all 64 bits.

Integrity Check Value (*ICV*): this field is coded on a multiple of 4 bytes and contains the data seal ensuring authentication and integrity checking.

```
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 4.4. *AH extension format*

```
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)                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Sequence Number                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Payload Data (variable)                     |
~                                                               ~
|                                                               |
+                 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 |          Padding (0-255 bytes)              |
+-+-+-+-+-+-+-+-+ +             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |   Pad Length  |  Next Header  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              Integrity Check Value-ICV (variable)             |
~                                                               ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4.5. *ESP extension format*

4.2.1.2. *ESP extension*

The ESP extension provides a confidentiality service in addition to those offered by the AH extension. The same ESP extension is used in association with an IPv4 or IPv6 header. The presence of the extension is indicated by the *Next Header* (in IPv6) or *Protocol* (in IPv4) field of the previous header, with a value of 50.

The ESP extension contains the same fields as the AH extension. It starts with the *SPI* and *Sequence Number* fields (Figure 4.5). After these fields comes the encapsulated data, which may contain synchronization data of the initialization vector (IV) encryptor. Following the encapsulated data, the extension ends with the following fields: *Padding*, *Pad Length*, *Next Header* and optionally *ICV* (authentication data).

The *Padding* field is necessary when block encryption is used, and the block must be of a certain size, and to align the packet size with a multiple of 4 bytes.

4.2.1.3. *Modes*

For transport mode, the AH or ESP header is inserted between the IP header and the source IP packet payload. In the IPv6 environment, the AH or ESP header appears after the *Hop-by-Hop*, *Destination*, *Routing*, and *Fragment* extensions.

For tunnel mode, the AH or ESP header encapsulates the source IP packet, and the whole is encapsulated in its turn by a new IP header. The tunnel corresponds to a data structure in which an IP packet contains another IP packet.

When the AH header is used, authentication is applied to the whole packet except the variable fields of the IP header (Figure 4.6).

The variable fields of the IPv4 header are set at zero to calculate the authentication digest:

– *DSCP*: the value of this field can be modified by an intermediary router when it checks traffic characteristics;

– *ECN*: the value of this field can be modified by an intermediary router to alert the destination that congestion is developing;

– *DF*: this bit can be set at one by an intermediary router;

– *Fragment Offset*: insertion of the AH header occurs on non-fragmented IP packets, and therefore this field has a value of zero;

– *TTL*: the value of this field is decreased by one unit per each router crossed;

– *Checksum*: the value of this field is recalculated as soon as a field in the IP header changes value.

The group of IPv4 header options is considered as a single entity. Some options can be modified by an intermediary router. If a single modifiable option appears, the group of options is set at zero for the authentication digest calculation.

The variable fields of the IPv6 header are set at zero for the authentication digest calculation. These are identical fields to the ones in the IPv4 header (*DSCP*, *ECN* and *Hop Limit*), as well as the *Flow Label* field.

The *Hop-by-Hop* and *Destination* extensions of the IPv6 header have a bit that indicates whether the option can be modified by an intermediary router or not. If this bit is set at one, the extension is set at zero for the authentication digest calculation.

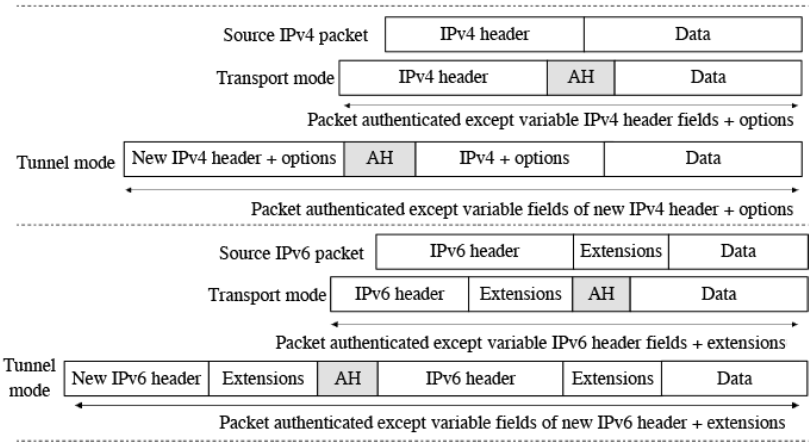

Source IPv4 packet
IPv4 header
Data
Transport mode
IPv4 header
AH
Data
Packet authenticated except variable IPv4 header fields + options
Tunnel mode
New IPv4 header + options
AH
IPv4 + options
Data
Packet authenticated except variable fields of new IPv4 header + options
Source IPv6 packet
IPv6 header
Extensions
Data
Transport mode
IPv6 header
Extensions
AH
Data
Packet authenticated except variable IPv6 header fields + extensions
Tunnel mode
New IPv6 header
Extensions
AH
IPv6 header
Extensions
Data
Packet authenticated except variable fields of new IPv6 header + extensions

Figure 4.6. *Position of AH header*

When the ESP header is used in transport mode, the confidentiality service is applied to the encapsulated data and ESP tail. Authentication and integrity services cover the ESP header, encapsulated data and ESP tail (Figure 4.7).

When the ESP header is used in tunnel mode, the confidentiality service is applied to the source IP packet and ESP tail. Authentication and integrity services cover the ESP header, source IP packet, and ESP tail (Figure 4.7).

Note that in transport mode, the *Destination* extension can appear before, after or simultaneously before and after the AH or ESP extension.

4.2.2. *Security association*

A SA is a simple connection between two end points offering security services (confidentiality, integrity and authentication) to traffic. Security services are provided by the use of AH or ESP extensions. To secure a two-directional communication between two end points, an SA pair is required. The Internet Key Exchange (IKE) protocol dynamically ensures the creation of the SA.

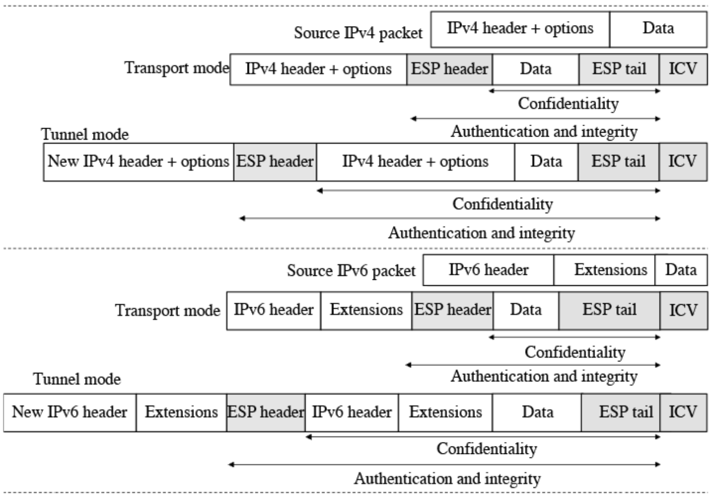


Figure 4.7. *Position of ESP header*

An SA contains the following parameters:

– the authentication algorithm and the key in order to generate the AH extension;

– the encryption algorithm and the key in order to generate the ESP extension;

– the authentication algorithm and the key in order to generate the ESP extension, if this service is used;

– the lifetime of the SA;

– the encapsulation mode (tunnel or transport).

The IPSec mechanism defines three databases:

– Security Policy Database (SPD). This defines the security policy to be applied to input and output traffic for a host or a security gateway;

– Security Association Database (SAD). This contains the parameters applied to an SA;

– Peer Authorization Database (PAD). This provides a link between the IKEv2 protocol and SPD database.

The selector is the mechanism enabling identification at the source of the SA to be applied to traffic. The selector uses fields (*Protocol* or *Next Header* and source or destination IP address) of the IP headers and fields (source or destination port) of the TCP or User Datagram Protocol (UDP) headers.

When a packet originates from the interface of the protected network (outgoing packet), the selector makes it possible to obtain entry to the SPD database that defines the process to be applied to the packet:

– BYPASS: the packet is transmitted without a security service;

– DISCARD: the packet is deleted;

– PROTECT: the security service is applied to the packet. If the SA is not established, the IKE protocol is invoked. If the SA exists, the database returns a pointer to the SAD database.

The deletion of a packet causes the generation from the security gateway toward the source of an ICMP message with the following characteristics:

– in the IPv4 environment, Type = 3 (destination unreachable) and Code = 13 (Communication Administratively Prohibited);

– in the IPv6 environment, Type = 1 (destination unreachable) and Code = 1 (Communication with Destination Administratively Prohibited).

The Security Parameter Index (SPI) is a field in the AH or ESP header used by the destination to identify the SA in a unique way. The destination uses this index to extract the SA from the SAD database.

When the packet originates from the interface of the non-protected network (incoming packet), the SPD database is consulted if the packet is not protected, and the instruction (BYPASS or DISCARD) is applied. If the IP packet is protected, the SPI field is used to recover the SA.

4.2.3. *PMTU processing*

The introduction of ESP or AH extensions has an impact on the packet size, which can exceed the MTU value. The Path MTU (PMTU) mechanism used to find the MTU value on the route makes it possible to avoid packet fragmentation by intermediary routers.

In the IPv4 environment, the PMTU mechanism uses the DF bit in the IP header positioned at one. In the tunnel mode used between two security gateways, it is, therefore, necessary for the security gateway to copy the value of this bit in the new header.

The PMTU mechanism uses an ICMP message indicating the MTU value. This message is sent back by an intermediary router, with the following characteristics:

– in the IPv4 environment, Type = 3 (Destination Unreachable) and Code = 4 (Fragmentation needed and DF set);

– in the IPv6 environment, *Type* = 2 (*Packet Too Big*) and *Code* = 0 (*Fragmentation needed*).

Thus, the security gateway can receive this message from an unauthenticated source (an intermediary router) in the SA. It must then analyze the content of the ICMP message to recover the information that will grant it access to the SAD database (IP addresses and SPI field). It must resend an ICMP message to the host with a new PMTU value taking the size of the AH or ESP extensions into account.

4.3. IKEv2 protocol

The IKEv2 protocol is more simplified than the previous version. It combines the functionalities defined in IKEv1 and Internet Security Association And Key Management Protocol (ISAKMP) while removing unnecessary processes. It eliminates the generic character of the previous version, integrating the domain of interpretation DOI function, which defines the parameters specific to the ESP/AH SA.

Each IKEv2 message is composed of an HDR header and a sequence of blocks. The IKEv2 message is encapsulated by a UDP header with source and destination port values of 500 or 4500. When the 4500 port is used, the IKEv2 message is preceded by 4 bytes at zero.

4.3.1. *Message header*

The header of the IKE message contains the following fields (Figure 4.8):

Initiator's SPI: this field, coded on 8 bytes, incorporates a value chosen by the initiator. It initializes identification of the IKE SA.

Responder's SPI: this field, coded on 8 bytes, incorporates a value chosen by the responder. It completes the identification of the IKE SA.

Next Payload: this field, coded on 1 byte, incorporates the indication of the type of block following the header (Table 4.3).

Notation	Designation
SA	*Security Association*
KE	*Key Exchange*
IDi	*Identification – initiator*
IDr	*Identification – responder*
CERT	*Certificate*
CERTREQ	*Certificate Request*
AUTH	*Authentication*
Ni	*Nonce – initiator*
Nr	*Nonce – responder*
N	*Notification*
D	*Delete*
V	*Vendor ID*
TSi	*Traffic Selector – initiator*
TSr	*Traffic Selector – responder*
SK	*Encrypted and Authenticated*
CP	*Configuration*
EAP	*Extensible Authentication*

Table 4.3. *Block types*

Major Version: this field, coded on 4 bits, indicates the maximum value of the IKE protocol version that can be used. This value is equal to 2 for the implementation of the IKEv2 protocol.

Minor Version: this field, coded on 4 bits, indicates the minimum value of the IKE protocol version. This value is equal to 0 for the implementation of the IKEv2 protocol.

Exchange Type: this field, coded on 1 byte, indicates the type of exchange to which the message belongs:

– IKE_SA_INIT: this exchange concerns the first phase of the establishment of the IKE SA.

– IKE_AUTH: this exchange concerns the second phase of the establishment of the IKE SA.

– CREATE_CHILD_SA: this exchange concerns the establishment of the ESP/AH SA.

– INFORMATIONAL: this exchange concerns event notification.

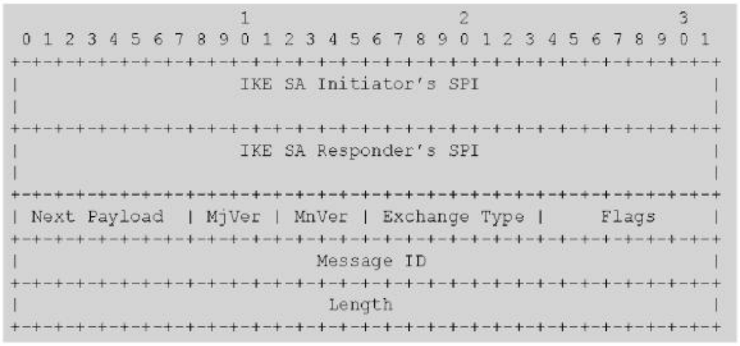
Each type of exchange imposes a certain number of required blocks composing the message and defines optional blocks.

Flags: this field includes the following three flags:

– R (response): this flag, positioned at one, indicates that this message is a response. An IKE termination must not respond to a response except when authentication has failed.

– V (version): this flag, positioned at one, indicates that the IKE termination is able to process a version higher than the one shown in the *Major Version* field.

– I (initiator): this flag, positioned at one, indicates that the message is generated by the initiator of the IKE SA.



```
                    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 4.8. *IKE message header format*

Message ID: this field, coded on 4 bytes, is an identifier used to control the retransmission of lost messages and to correlate the request and response. It also protects against replay attacks.

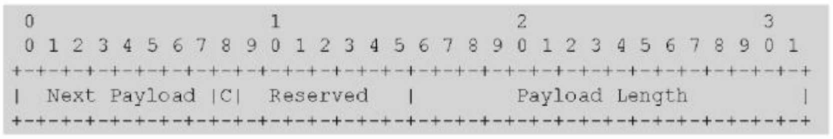
Length: this field, coded on 4 bytes, includes the IKE message size.

4.3.2. *Blocks*

Each block starts with a generic header containing the *Next Payload* field, the C (critical) bit and the *Payload Length* field (Figure 4.9). The *Next Payload* field indicates the type of block that comes next, thus enabling chaining. The C bit determines the process to be executed when the receiver does not recognize the block:

– if the C bit is positioned at one, the receiver rejects the message;

– if the C bit is positioned at zero, the receiver ignores the block and processes the next block.



```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Payload |C|  Reserved    |         Payload Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4.9. *Format of generic block header*

4.3.2.1. *SA block*

The SA block is used for the negotiation of IKE and ESP/AH SA parameters. An SA block can contain several proposals (P) ranked in order of preference. Each proposal defines a protocol (IKE, ESP or AH) and the value of the SPI index. Each proposal includes several transformations (T), and each transformation includes one or more attributes (A).

The transformation T involves the following operations:

– the encryption algorithm ENCR. This operation is used for negotiation pertaining to the IKE and ESP protocols.

ENCR_DES_IV64
ENCR_DES 2
ENCR_3DES
ENCR_RC5
ENCR_IDEA

ENCR_CAST
ENCR_BLOWFISH
ENCR_3IDEA
ENCR_DES_IV32
ENCR_NULL
ENCR_AES_CBC
ENCR_AES_CTR

– the pseudo-random function PRF. This operation is used for negotiation pertaining to the IKE protocol.

PRF_HMAC_MD5
PRF_HMAC_SHA1
PRF_HMAC_TIGER

– the integrity algorithm INTEG. This operation is used for negotiation pertaining to IKE, AH and (optionally) ESP protocols.

NONE
AUTH_HMAC_MD5_96
AUTH_HMAC_SHA1_96
AUTH_DES_MAC
AUTH_KPDK_MD5
AUTH_AES_XCBC_96

– the Diffie–Hellman (D-H) group. This operation is used for negotiation pertaining to IKE and (optionally) AH and ESP protocols.

NONE
768-bit MODP
1024-bit MODP
1536-bit MODP
2048-bit MODP
3072-bit MODP
4096-bit MODP
6144-bit MODP
8192-bit MODP

– the ESN. This operation is used for negotiation pertaining to the AH and ESP protocols.

No ESNs
ESNs

The attribute A specifies the length of the encryption algorithm key defined in the ENCR transformation. The other transformations, PRF, INTEG, D-H and ESN have no attributes.

4.3.2.2. *KE block*

The Key Exchange (KE) block contains the public Diffie–Hellman value, enabling each end point (initiator and responder) to construct a shared secret. The block also mentions the D-H group defined in the SA block.

4.3.2.3. *IDi and IDr blocks*

Identification initiator (IDi) and Identification responder (IDr) blocks contain an identification of the initiator of the IKE message and the responder. This identification is based on an IPv4 or IPv6 address, a name, a messaging address or a group of bytes.

ID_IPV4_ADDR
ID_FQDN
ID_RFC822_ADDR
ID_IPV6_ADDR
ID_DER_ASN1_DN
ID_DER_ASN1_GN
ID_KEY_ID

4.3.2.4. *CERT block*

The certificate (CERT) block provides a means of transporting a certificate or information pertaining to authentication.

PKCS #7 wrapped X.509 certificate
PGP certificate
DNS signed key

X.509 certificate – signature
Kerberos token
Certificate revocation list
Authority revocation list
SPKI certificate
X.509 certificate – attribute
Raw RSA key
Hash and URL of X.509 certificate
Hash and URL of X.509 bundle

4.3.2.5. *CERTREQ block*

The certificate request (CERTREQ) block is a request pertaining to a certificate. It is used in the response of the IKE_INIT_SA exchange response or in the IKE_AUTH exchange request. It also indicates the certification authority for the required certificate.

4.3.2.6. *The AUTH block*

The authentication (AUTH) block contains the authentication digest of the message. The block also specifies the method used.

RSA digital signature
Shared key message integrity code
DSS digital signature

4.3.2.7. *Ni and Nr blocks*

Nonce initiator (Ni) and nonce responder (Nr) blocks contain a random number generated by the initiator and responder. These numbers are used in the creation of derived keys.

4.3.2.8. *N block*

The notification (N) blocks contain error messages indicating the reason why the SA cannot be established.

UNSUPPORTED_CRITICAL_PAYLOAD
INVALID_IKE_SPI
INVALID_MAJOR_VERSION
INVALID_SYNTAX
INVALID_MESSAGE_ID
INVALID_SPI
NO_PROPOSAL_CHOSEN
INVALID_KE_PAYLOAD
AUTHENTICATION_FAILED
SINGLE_PAIR_REQUIRED
NO_ADDITIONAL_SAS
INTERNAL_ADDRESS_FAILURE
FAILED_CP_REQUIRED
TS_UNACCEPTABLE
INVALID_SELECTORS
TEMPORARY_FAILURE
CHILD_SA_NOT_FOUND

The N block also contains status messages that an SA management process wishes to communicate to a remote process.

INITIAL_CONTACT
SET_WINDOW_SIZE
ADDITIONAL_TS_POSSIBLE
IPCOMP_SUPPORTED
NAT_DETECTION_SOURCE_IP
NAT_DETECTION_DESTINATION_IP
COOKIE
USE_TRANSPORT_MODE
HTTP_CERT_LOOKUP_SUPPORTED
REKEY_SA
ESP_TFC_PADDING_NOT_SUPPORTED
NON_FIRST_FRAGMENTS_ALSO

4.3.2.9. *D block*

The Delete (D) block includes the SPI index of the SA that the message source wishes to delete. For an AH or ESP protocol, it is possible to specify several index values. It is also possible to string several D blocks together in a single IKEv2 message.

4.3.2.10. *V block*

The Vendor ID (V) block announces that the message source is capable of accepting private extensions of the IKEv2 protocol. These extensions can involve the introduction of new blocks, new types of exchange or new notification information.

4.3.2.11. *TS block*

The Traffic Selector (TS) block identifies the flows for which the ESP/AH SA is implemented. Flow determination is based on the following information:

– the type of data encapsulated by the IP header, stated in the *Protocol* fields of the IPv4 header or the *Next Header* field of the IPv6 header;

– the range of source and destination IP addresses;

– the range of source and destination port numbers if the IP header encapsulates UDP or TCP segments;

– the ICMP message type and code.

4.3.2.12. *SK block*

The SK (encrypted and authenticated) block is always located at the end of the IKEv2 message. The encryption and integrity algorithms of the IKEv2 algorithms are negotiated during the implementation of the IKEv2 SA.

4.3.2.13. *CP block*

The Configuration (CP) block is used to exchange configuration information between the two end points. In the case of an ESP/AH SA between a host and a security gateway, the host can request information concerning a host in the protected network.

INTERNAL_IP4_ADDRESS
INTERNAL_IP4_NETMASK
INTERNAL_IP4_DNS
INTERNAL_IP4_NBNS
INTERNAL_IP4_DHCP
APPLICATION_VERSION
INTERNAL_IP6_ADDRESS
INTERNAL_IP6_DNS
INTERNAL_IP6_DHCP
INTERNAL_IP4_SUBNET
SUPPORTED_ATTRIBUTES
INTERNAL_IP6_SUBNET

4.3.2.14. *EAP block*

The Extensible Authentication Protocol (EAP) block enables authentication of the IKE SA by the EAP protocol.

4.3.3. *Procedure*

4.3.3.1. *IKE_SA_INIT exchange*

The first exchange, IKE_SA_INIT, negotiates cryptographic algorithms and random numbers and executes a Diffie–Hellman exchange in order to create an IKE SA.

The initiator generates an IKE message containing the HDR header and the SAi1, KEi and Ni blocks. The HDR header contains the initiator's SPI index, version numbers and flags. The SAi1 block contains the cryptographic algorithms proposed by the initiator for the IKE SA. The KEi block includes the Diffie–Hellman group and public value. The Ni block displays the initiator's random number (Figure 4.10).

The responder chooses a cryptograph series from the initiator's proposals and includes it in the SAr1 block. It completes the exchange of Diffie–Hellman keys with the KEr block. It sends its random number in the Nr block (Figure 4.10). It can possibly communicate a list of certificate authorities in the CERTREQ block.

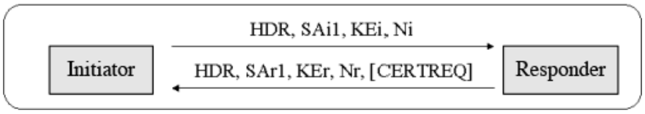


Figure 4.10. *IKE_SA_INIT exchange*

At this stage of the negotiation, each end point can generate the SKEYSEED key. The keys used for encryption and integrity of IKE messages are produced by the SKEYSEED key and are known as SK_e (encryption) and SK_a (integrity). Message protection involves only the blocks; the header is not included.

The two different directions of traffic use different keys. The keys used to protect messages from the initiator are SK_ai and SK_ei. The keys used to protect messages in the other direction are SK_ar and SK_er.

Other keys are also derived from the SKEYSEED key. The SK_d key is used to derive the keys used in the ESP/AH SA. The SK_p key is used to calculate the AUTH block digest.

The SKEYSEED key is calculated using the Diffie–Hellman secret (D-H key) and the random numbers Ni and Nr:

SKEYSEED = PRF (D-H key, Ni | Nr)

The keys SK_d, SK_ai, SK_ar, SK_ei, SK_er, SK_pi and SK_pr are generated as follows:

SK_d = PRF (SKEYSEED, Ni | Nr | SPIi | SPIr | 0x01)
SK_ai = PRF (SKEYSEED, SK_d | Ni | Nr | SPIi | SPIr | 0x02)
SK_ar = PRF (SKEYSEED, SK_ai | Ni | Nr | SPIi | SPIr | 0x03)
SK_ei = PRF (SKEYSEED, SK_ar | Ni | Nr | SPIi | SPIr | 0x04)
SK_er = PRF (SKEYSEED, SK_ei | Ni | Nr | SPIi | SPIr | 0x05)
SK_pi = PRF (SKEYSEED, SK_er | Ni | Nr | SPIi | SPIr | 0x06)
SK_pr = PRF (SKEYSEED, SK_pi | Ni | Nr | SPIi | SPIr | 0x07)

4.3.3.2. *IKE_AUTH exchange*

The second exchange, IKE_AUTH, is used to authenticate previous IKE messages and communicate identities and possibly exchange certificates, as well as to establish the first AH/ESP SA. These messages are completely encrypted and protected by the keys established during the IKE_SA_INIT exchange.

The initiator indicates its identity with the IDi block. It authenticates its identity and protects the integrity of the first message in the IKE_SA_INIT exchange using the AUTH block. It can possibly send its certificate in the CERT block and the certification authority's identity in the useful CERTREQ payload load (Figure 4.11).

The optional IDr block enables the initiator to specify which of the responder's identities it wishes to communicate with (Figure 4.11).

The initiator starts the AH/ESP SA negotiation with the SAi2 block. The TSi block specifies the characteristics of the packets transferred by the initiator. The TSr block specifies the address for packets transferred to the responder (Figure 4.11).

The notation SK{ ... } indicates that the blocks are completely encrypted and protected (Figure 4.11).

The responder communicates its identity in the IDr block. It may send a certificate. It authenticates its identity and protects the integrity of the second message of the IKE_SA_INIT exchange. It completes the negotiation of the ESP/AH SA with the SAr2 block (Figure 4.11).

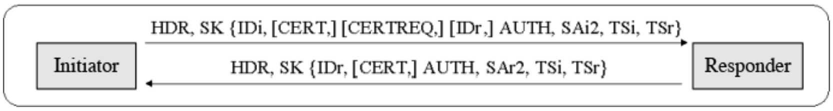

HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, SAi2, TSi, TSr}
Initiator
HDR, SK {IDr, [CERT,] AUTH, SAr2, TSi, TSr}
Responder

Figure 4.11. *IKE_AUTH exchange*

4.3.3.3. *CREATE_CHILD_SA exchange*

The CREATE_CHILD_SA exchange is used to create the ESP/AH SA and to renew IKE and ESP/AH SA keys.

For the exchange involving the creation of the ESP/AH SA, the initiator finalizes the SA in the SA block and the traffic selectors proposed for the SA in the TSi and TSr blocks. It transmits a random number in the Ni block and optionally a Diffie–Hellman value in the KEi block (Figure 4.12).

The responder confirms the SA offer in the SA block. It transmits a Diffie–Hellman value in the KEr block, if the KEi block has been included in the request (Figure 4.12).

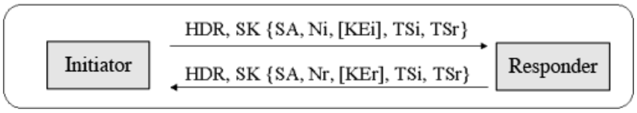

Initiator
HDR, SK {SA, Ni, [KEi], TSi, TSr}
HDR, SK {SA, Nr, [KEr], TSi, TSr}
Responder

Figure 4.12. *CREATE_CHILD_SA exchange creation of ESP/AH SA*

The KEYMAT key, used for the ESP/AH SA, is derived from the SK_d key and the random numbers Ni and Nr. If the exchange contains Diffie–Hellman values, the secret D-H key obtained also participates in the creation of the KEYMAT key.

KEYMAT = PRF (SK_d, Ni | Nr)

KEYMAT = PRF (SK_d, D-H key | Ni | Nr)

To renew the IKE SA key, the initiator sends the SA in the SA block, a random number in the Ni block and a Diffie–Hellman value in the KEi block. The initiator's new SPI index is provided in the SA block (Figure 4.13).

The responder confirms the offer in the SA block. It transmits a Diffie–Hellman value in the KEr block. The responder's new SPI index is provided in the SA block (Figure 4.13).

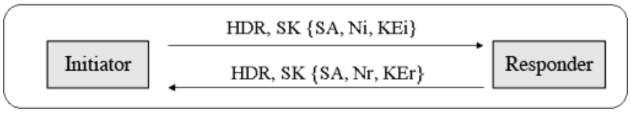


Figure 4.13. *CREATE_CHILD_SA exchange renewal of IKE SA key*

The new SKEYSEED key is calculated from the old SK_d key, the D-H secret key and the random numbers Ni and Nr.

SKEYSEED = PRF (old SK_d, D-H key | Ni | Nr)

To renew the ESP/AH SA key, the messages transmitted by the initiator and the responder are similar to the ones used in the creation of the SA. The initiator's request contains an N block (REKEY_SA) containing the SPI index value of the new SA (Figure 4.14).

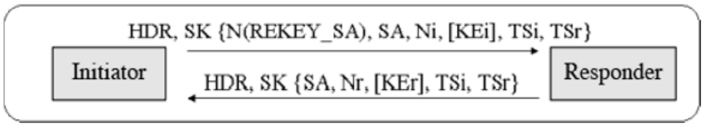

HDR, SK {N(REKEY_SA), SA, Ni, [KEi], TSi, TSr}
Initiator
HDR, SK {SA, Nr, [KEr], TSi, TSr}
Responder

Figure 4.14. *CREATE_CHILD_SA key renewal of ESP/AH SA key*