Chapter 13

Security at the Network Layer: IPSec

Chapter taken from below book
In book this chapter is named as 18 so don't confuse.

Chapter 13

Objectives

- To define the architecture of IPSec
- To discuss the application of IPSec in transport and tunnel modes
- To discuss how IPSec can be used to provide only authentication
- To discuss how IPSec can be used to provide both confidentiality and authentication
- To define Security Association and explain how it is implemented for IPSec
- To define Internet Key Exchange and explain how it is used by IPSec.

IETF IPSec Working Group

- The basic Internet Protocol (IP) has absolutely no security concepts integrated
  - Data are transferred in plain text
  - Authentication is not possible, everybody can fake the IP sender address or modify the payload of a packet
- Thus: specification of a security architecture for IP that comprises authentication and encryption mechanism
  - Security mechanisms should be algorithm-independent, cryptographic algorithms can be altered without effecting other parts of the protocol
  - Wide variety of security policies should be supported
- Result:
  - Compatibility with the Internet structure as given by IPv4 was needed to introduce security functions earlier than the "rest" of IPv6
  - Thus: in 1992 the *IP Security Protocol (IPSec) was standardized, together with theInternet Key Exchange (IKE)*

In computing, **Internet Protocol Security** (**IPsec**) is a secure network protocol suite that authenticates and encrypts the packets of data to provide secure encrypted communication between two computers over an Internet Protocol network.

It is used in virtual private networks (VPNs).
IPsec includes protocols for establishing mutual authentication between agents at the beginning of a session and negotiation of cryptographic keys to use during the session.

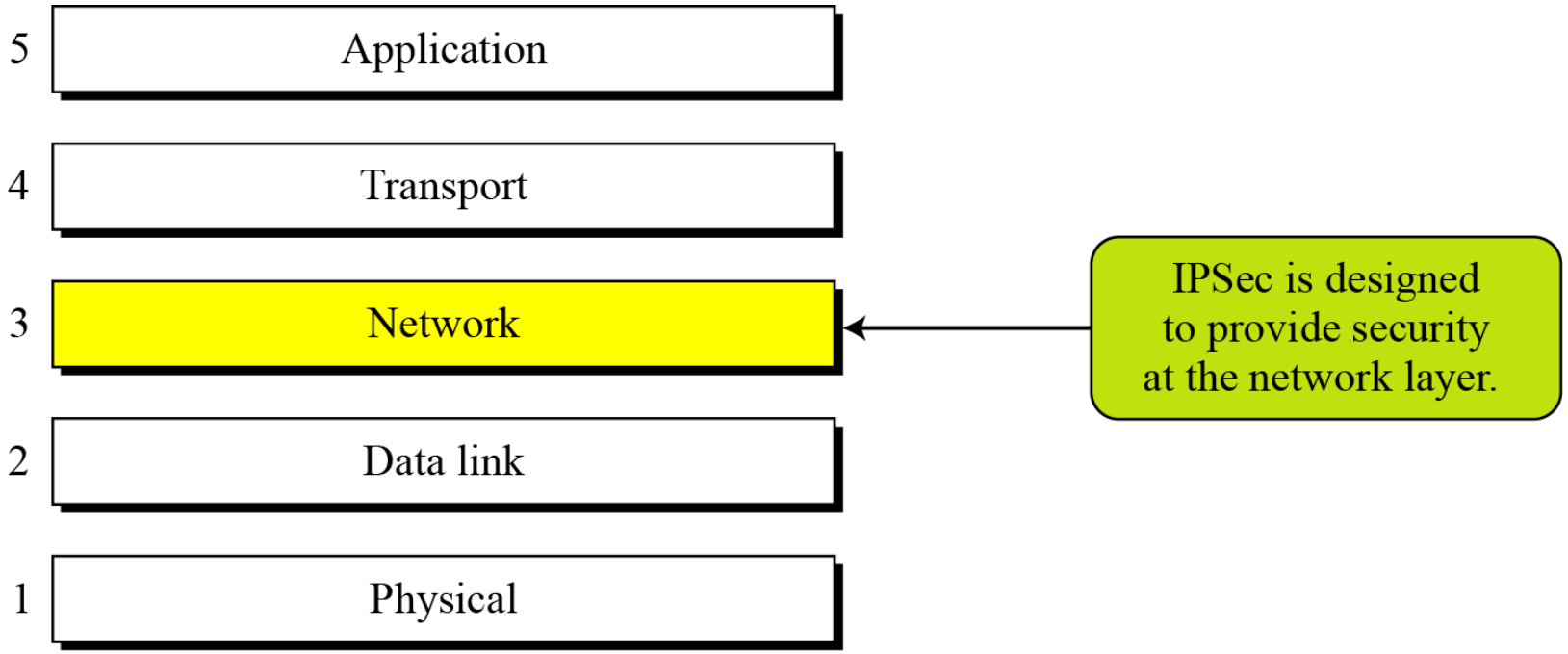
IPsec can protect data flows between a pair of hosts (*host-to-host*), between a pair of security gateways (*network-to-network*), or between a security gateway and a host (*network-to-host*).[1]

IPsec uses cryptographic security services to protect communications over Internet Protocol (IP) networks.

It supports network-level peer authentication, data-origin authentication, data integrity, data confidentiality (encryption), and replay protection.

The initial IPv4 suite was developed with few security provisions. As a part of the IPv4 enhancement, IPsec is a layer 3 OSI model or internet layer end-to-end security scheme. In contrast, while some other Internet security systems in widespread use operate above layer 3, such as Transport Layer Security (TLS) that operates at the Transport Layer and Secure Shell (SSH) that operates at the Application layer, IPsec can automatically secure applications at the IP layer.

Figure 18.1 *TCP/IP Protocol Suite and IPSec*



18-1 TWO MODES

IPSec operates in one of two different modes: transport mode or tunnel mode.

Topics discussed in this section:

IPSec Modes: Transport and Tunnel

Three different basic implementation architectures can be used to provide IPSec facilities to TCP/IP networks. The choice of which implementation we use, as well as whether we implement in end hosts or routers, impacts the specific way that IPSec functions. Two specific *modes* of operation are defined for IPSec that are related to these architectures, called *transport mode* and *tunnel mode*.

IPSec modes are closely related to the function of the two core protocols, the Authentication Header (AH) and Encapsulating Security Payload (ESP). Both of these protocols provide protection by adding to a datagram a header (and possibly other fields) containing security information. The choice of mode does not affect the method by which each generates its header, but rather, changes what specific parts of the IP datagram are protected and how the headers are arranged to accomplish this. In essence, the mode really ***describes***, not ***prescribes*** how AH or ESP do their thing. It is used as the basis for defining other constructs, such as security associations (SAs).

18.1.1 Transport Mode

In transport mode, IPSec protects what is delivered from the transport layer to the network layer.

Note

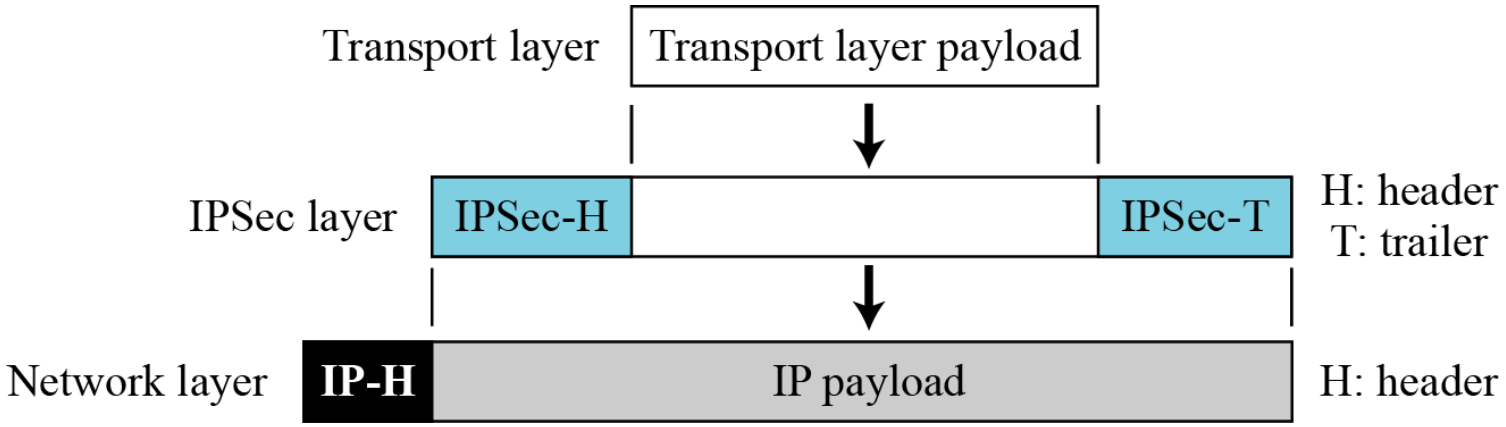
IPSec in transport mode does not protect the IP header; it only protects the information coming from the transport layer.

IPSec Modes: Transport and Tunnel

Transport Mode

As its name suggests, in transport mode, the protocol protects the message passed down to IP from the transport layer. The message is processed by AH/ESP and the appropriate header(s) added in front of the transport (UDP or TCP) header. The IP header is then added in front of that by IP. Another way of looking at this is as follows. Normally the transport layer packages data for transmission and sends it to IP. From IP's perspective, this transport layer message is the payload of the IP datagram. When IPSec is used in transport mode, the IPSec header is applied only over this IP payload, ***not*** the IP header. The AH and/or ESP headers appears between the original, single IP header and the IP payload. This is illustrated in Figure 119.

Figure 18.2 *IPSec in transport mode*



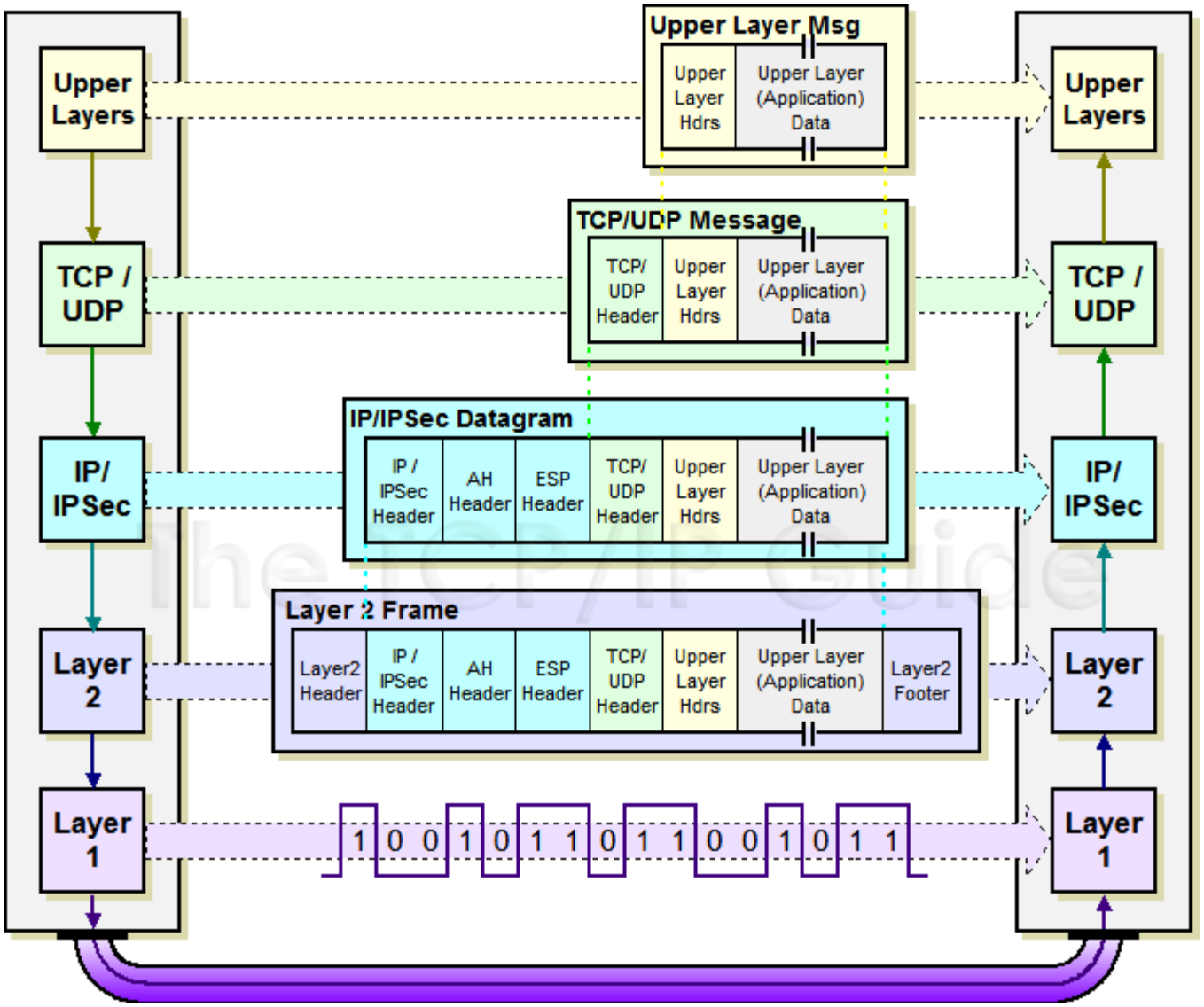
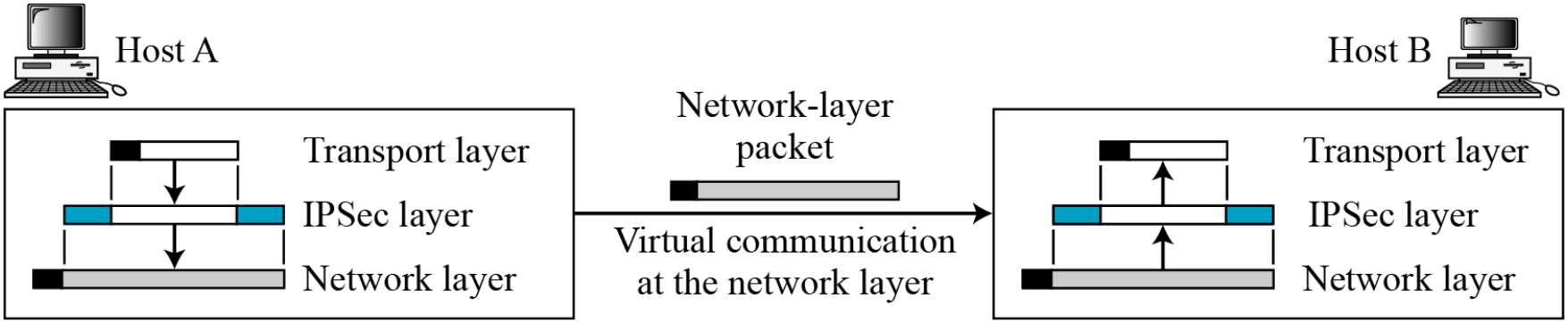


figure 119: IPSec Transport Mode Operation

When IPSec operates in transport mode, it is integrated with IP and used to transport the upper layer (TCP/UDP) message directly. After processing, the datagram has just one IP header that contains the AH and/or ESP IPSec headers.

Figure 18.3 *Transport mode in action*



18.1.2 Tunnel Mode

In tunnel mode, IPSec protects the entire IP packet. It takes an IP packet, including the header, applies IPSec security methods to the entire packet, and then adds a new IP header.

Note

IPSec in tunnel mode protects the original IP header.

Tunnel Mode

In this mode, IPSec is used to protect a complete ***encapsulated*** IP datagram after the IP header has already been applied to it. The IPSec headers appear in front of the original IP header, and then a ***new*** IP header is added in front of the IPSec header. That is to say, the entire original IP datagram is secured and then encapsulated within another IP datagram. This is shown in Figure 120.

Figure 18.4 *IPSec in tunnel mode*

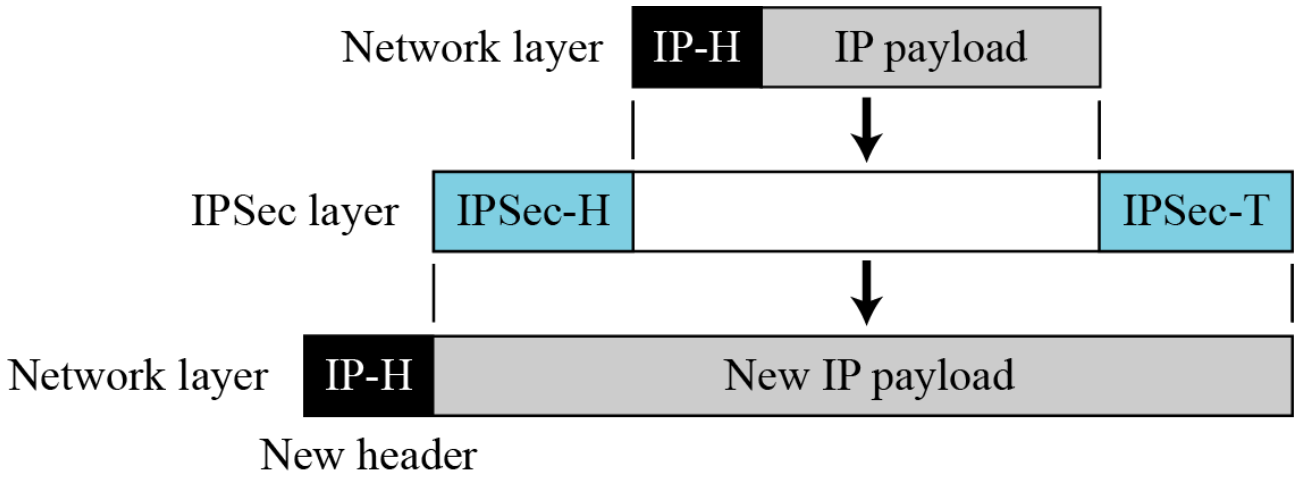
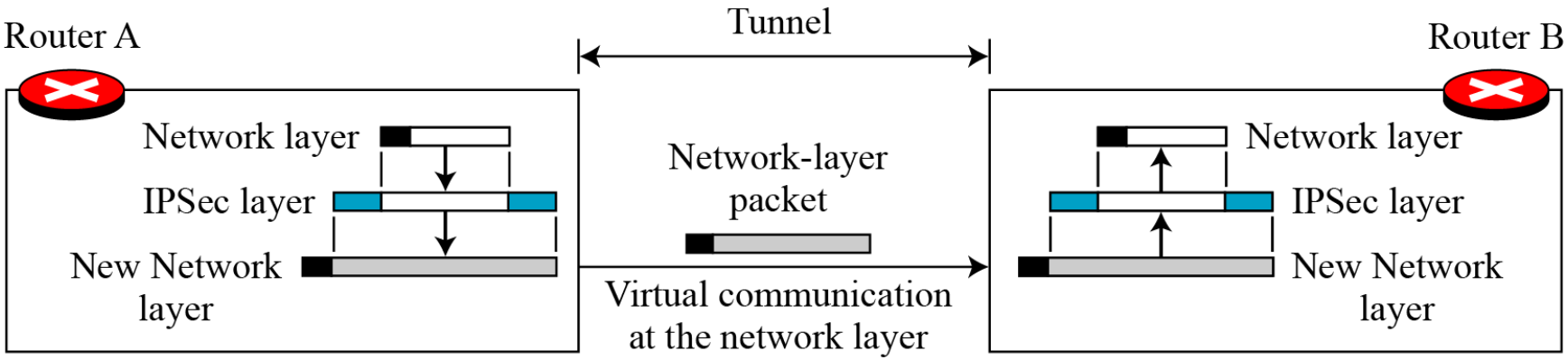


Figure 18.5 *Tunnel mode in action*



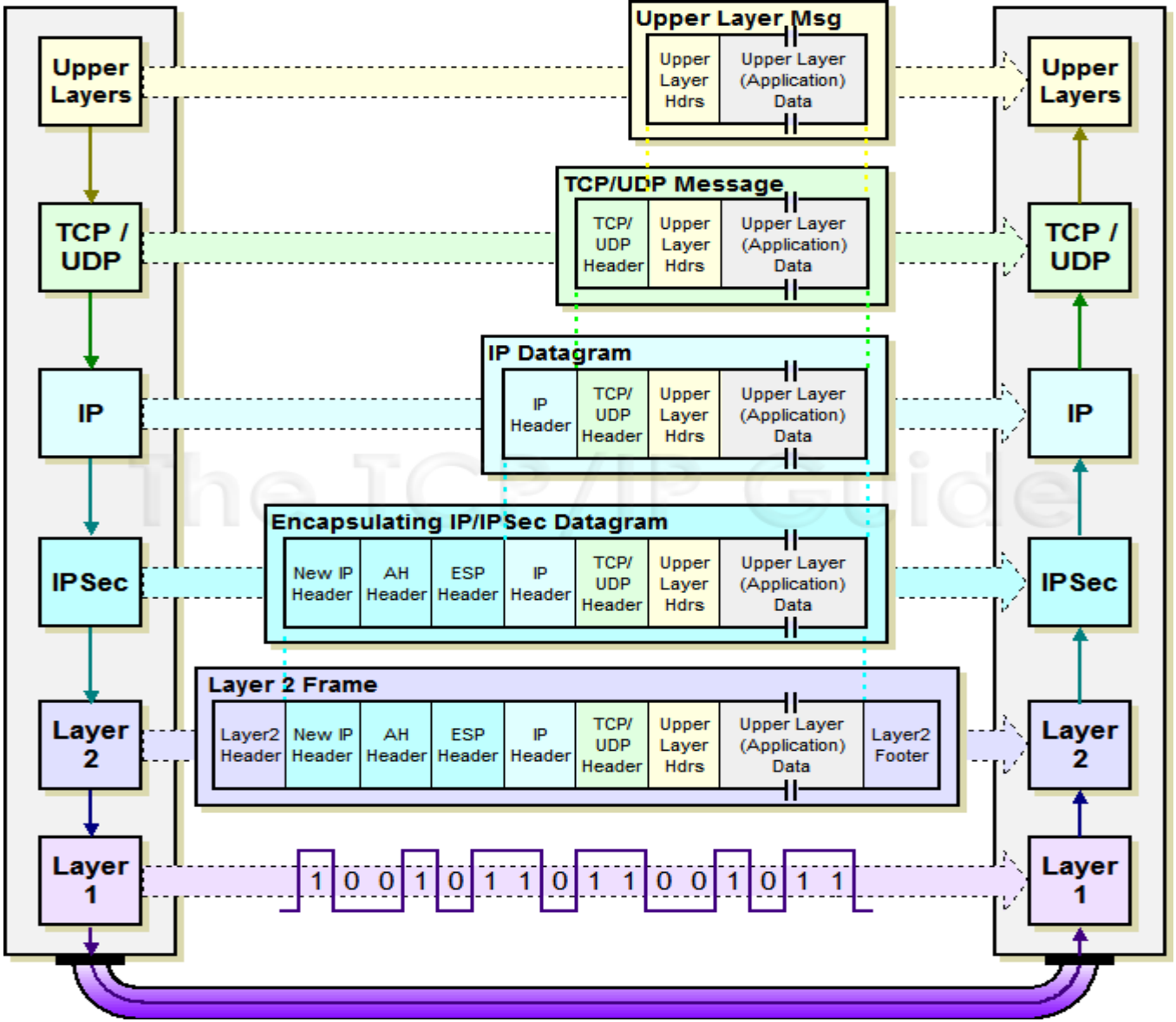
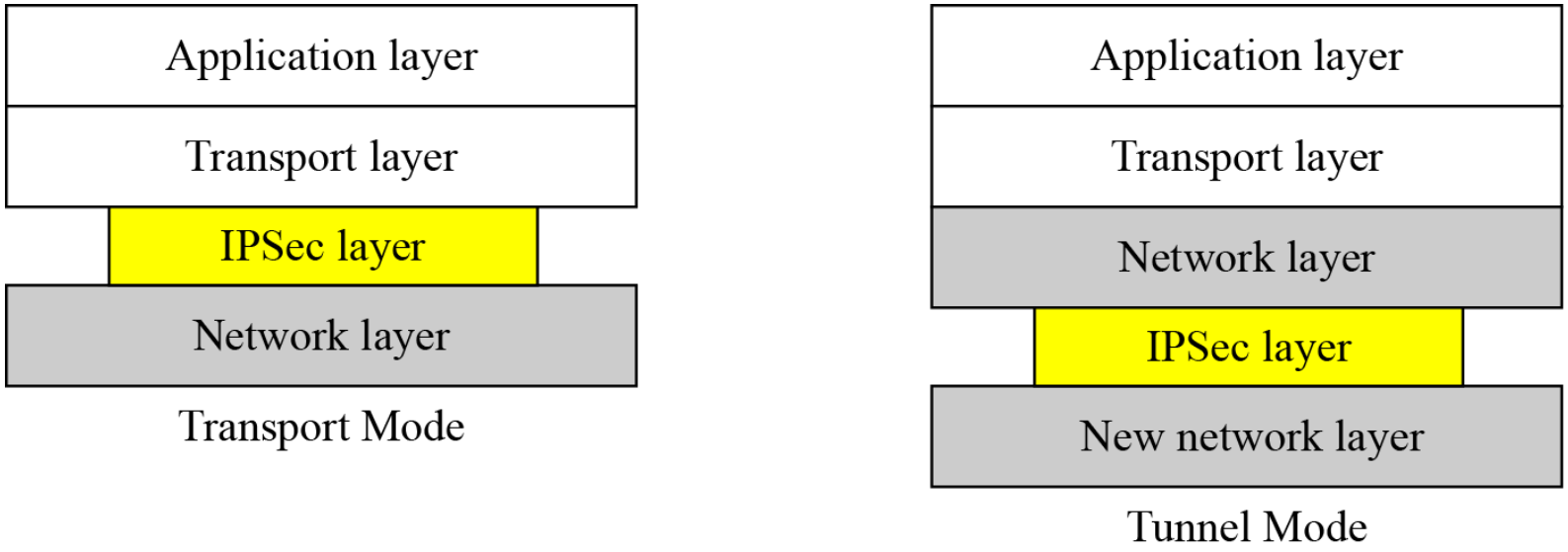


Figure 120: IPSec Tunnel Mode Operation

IPSec *tunnel mode* is so named because it represents an encapsulation of a complete IP datagram, forming a virtual tunnel between IPSec-capable devices. The IP datagram is passed to IPSec, where a new IP header is created with the AH and/or ESP IPSec headers.added. Contrast to Figure 119.

18.1.3 Comparison

Figure 18.6 *Transport mode versus tunnel mode*



18-3 SECURITY ASSOCIATION

Security Association is a very important aspect of IPSec. IPSec requires a logical relationship, called a Security Association (SA), between two hosts. This section first discusses the idea and then shows how it is used in IPSec.

Topics discussed in this section:

IPSec Security Associations and the Security Association Database (SAD); Security Policies and the Security Policy Database (SPD); Selectors; the Security Parameter Index (SPI)

These constructs are used to guide the operation of IPSec in a general way and also in particular exchanges between devices.

They control how IPSec works and ensure that each datagram coming into or leaving an IPSec-capable device is properly treated.

Where to start… where to start. ☺ Let's begin by considering the problem of how to apply security in a device that may be handling many different exchanges of datagrams with others.

There is overhead involved in providing security, so we do not want to do it for every message that comes in or out.

Some types of messages may need more security, others less. Also, exchanges with certain devices may require different processing than others.

Security Policies, Security Associations and Associated Databases:

To manage all of this complexity, IPSec is equipped with a flexible, powerful way of specifying how different types of datagrams should be handled.

To understand how this works, we must first define two important logical concepts:

- **Security Policies:** A *security policy* is a rule that is programmed into the IPSec implementation that tells it how to process different datagrams received by the device.

For example, security policies are used to decide if a particular packet needs to be processed by IPSec or not; those that do not bypass AH and ESP entirely.

If security is required, the security policy provides general guidelines for how it should be provided, and if necessary, links to more specific detail.

Security policies for a device are stored in the device's *Security Policy Database (SPD)*.

- **Security Associations:** A *Security Association (SA)* is a set of security information that describes a particular kind of secure connection between one device and another.

You can consider it a "contract", if you will, that specifies the particular security mechanisms that are used for secure communications between the two.

- A device's security associations are contained in its *Security Association Database (SAD)*.

It's often hard to distinguish the SPD and the SAD, since they are similar in concept.

The main difference between them is that security policies are general while security associations are more specific.

To determine what to do with a particular datagram, a device first checks the SPD.

The security policies in the SPD may reference a particular security association in the SAD.

If so, the device will look up that security association and use it for processing the datagram.

Selectors

One issue we haven't covered yet is how a device determines what policies or SAs to use for a specific datagram. Again here, IPSec defines a very flexible system that lets each security association define a set of rules for choosing datagrams that the SA applies to.

Each of these rule sets is called a *selector*. For example, a selector might be defined that says that a particular range of values in the *Source Address* of a datagram, combined with another value in the *Destination Address*, means a specific SA must be used for the datagram.

Let's now come back to security associations, which are a very important concept in IPSec. Each secure communication that a device makes to another requires that an SA be established.

SAs are unidirectional, so each one only handles either inbound or outbound traffic for a particular device. This allows different levels of security to be implemented for a flow from device *A* to device *B*, than for traffic coming from device *B* to device *A*. In a bidirectional communication of this sort, both *A* and *B* would have two SAs; *A* would have SAs we could call "SAdeviceBin" and "SAdeviceBout". Device *B* would have SAs "SAdeviceAin" and "SAdeviceAout".

Security Association Triples and the Security Parameter Index (SPI)Security associations don't actually have names, however. They are instead defined by a set of three parameters, called a *triple*:

- **Security Parameter Index (SPI):** A 32-bit number that is chosen to uniquely identify a particular SA for any connected device. The SPI is placed in AH or ESP datagrams and thus links each secure datagram to the security association. It is used by the recipient of a transmission so it knows what SA governs the datagram.

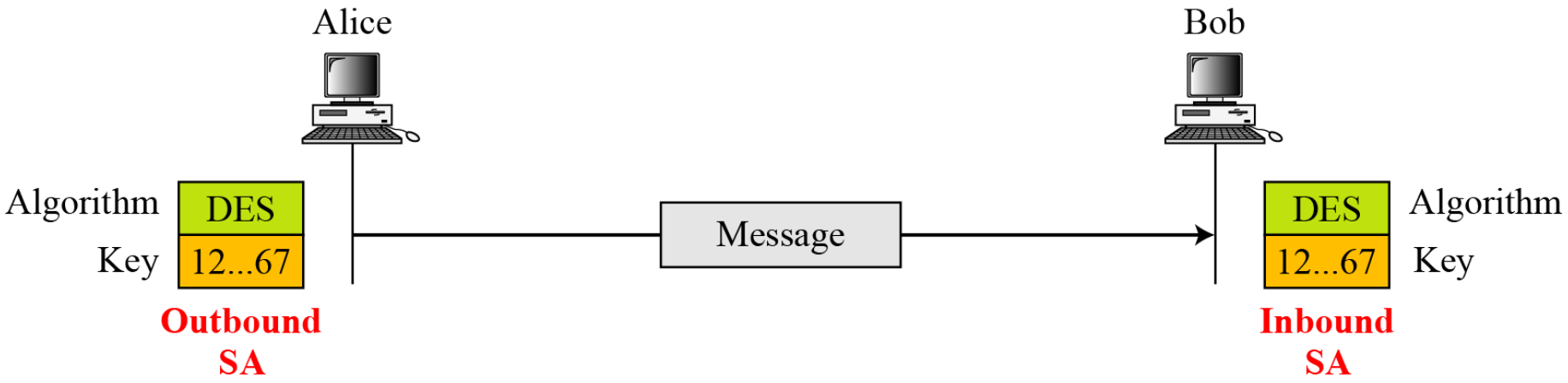
- **IP Destination Address:** The address of the device for whom the SA is established.

- **Security Protocol Identifier:** Specifies whether this association is for AH or ESP. If both are in use with this device they have separate SAs.

As you can see, the two security protocols AH and ESP are dependent on security associations and policies and the various databases that control their operation. Management of these databases is important, but another whole complex subject. Generally, SAs can either be set up manually (which is of course extra work) or an automated system can be deployed using a protocol like IKE.

18.3.1 Idea of Security Association

Figure 18.10 *Simple SA*



18.3.2 Security Association Database (SAD)

Figure 18.11 *SAD*

Index	SN	OF	ARW	AH/ESP	LT	Mode	MTU
< SPI, DA, P >							
< SPI, DA, P >							
< SPI, DA, P >							
< SPI, DA, P >							

Security Association Database

Legend:

SPI: Security Parameter Index

DA: Destination Address

AH/ESP: Information for either one

P: Protocol

Mode: IPSec Mode Flag

SN: Sequence Number

OF: Overflow Flag

ARW: Anti-Replay Window

LT: Lifetime

MTU: Path MTU (Maximum Transfer Unit)

18.3.2 (Continued)

Table 18.2 *Typical SA Parameters*

Parameters	*Description*
Sequence Number Counter	This is a 32-bit value that is used to generate sequence numbers for the AH or ESP header.
Sequence Number Overflow	This is a flag that defines a station's options in the event of a sequence number overflow.
Anti-Replay Window	This detects an inbound replayed AH or ESP packet.
AH Information	This section contains information for the AH protocol: 1. Authentication algorithm 2. Keys 3. Key lifetime 4. Other related parameters
ESP Information	This section contains information for the ESP protocol: 1. Encryption algorithm 2. Authentication algorithm 3. Keys 4. Key lifetime 5. Initiator vectors 6. Other related parameters
SA Lifetime	This defines the lifetime for the SA.
IPSec Mode	This defines the mode, transport or tunnel.
Path MTU	This defines the path MTU (fragmentation).

18-4 SECURITY POLICY

Another import aspect of IPSec is the Security Policy (SP), which defines the type of security applied to a packet when it is to be sent or when it has arrived. Before using the SAD, discussed in the previous section, a host must determine the predefined policy for the packet.

Topics discussed in this section:

18.4.1 Security Policy Database

Figure 18.12 *Connection identifiers*

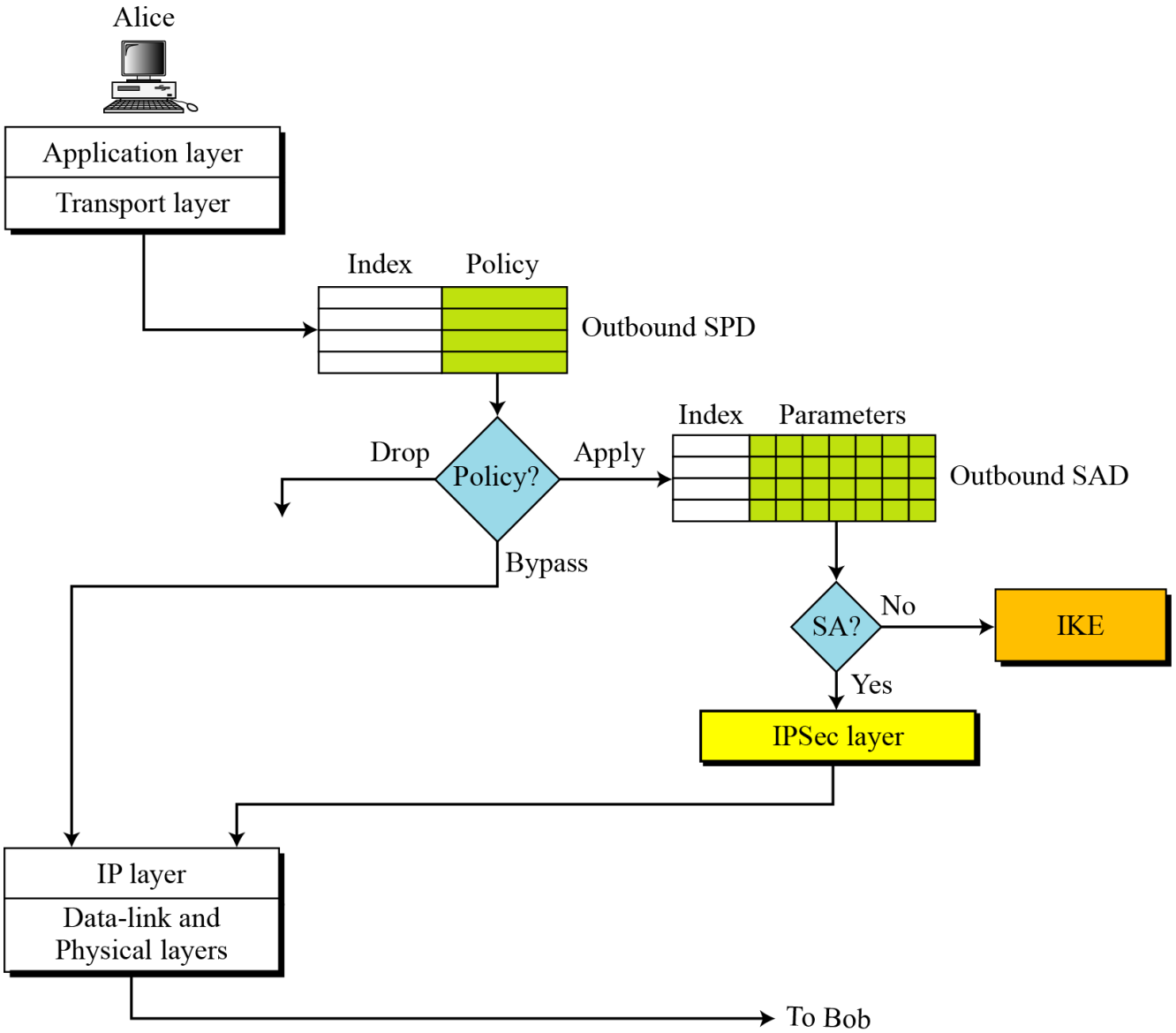
Index	Policy
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	

Legend:

SA: Source Address
SPort: Source Port
DA: Destination Address
DPort: Destination Port
P: Protocol

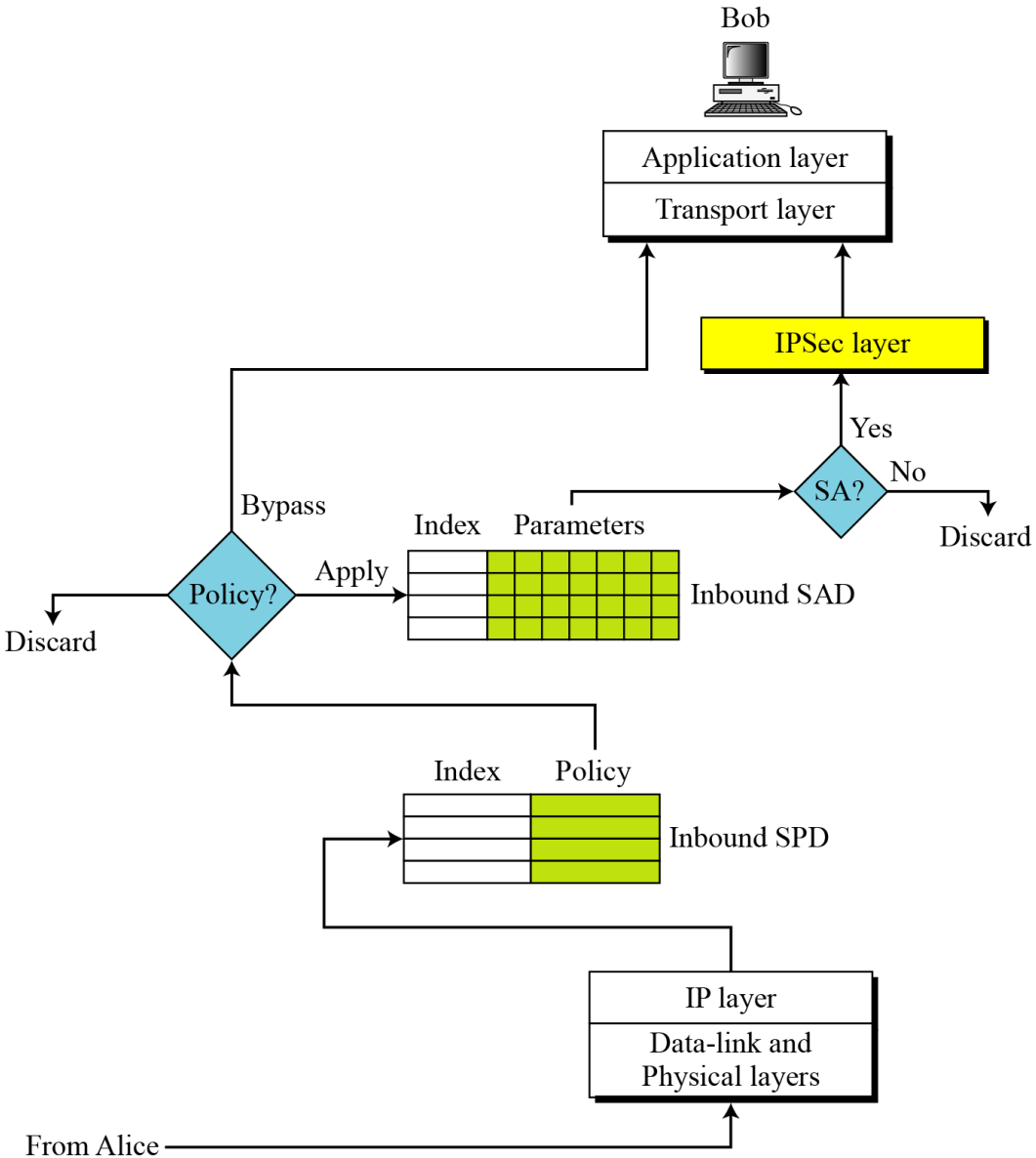
18.4.1 (Continued)

Figure 18.13 *Outbound processing*



18.4.1 (Continued)

Figure 18.14 *Inbound processing*



18-2 TWO SECURITY PROTOCOL

IPSec defines two protocols—the Authentication Header (AH) Protocol and the Encapsulating Security Payload (ESP) Protocol—to provide authentication and/or encryption for packets at the IP level.

Topics discussed in this section:

18.2.1 Authentication Header (AH)
18.2.2 Encapsulating Security Payload (ESP)
18.2.3 IPv4 and IPv6
18.2.4 AH versus ESP
18.2.5 Services Provided by IPSec

18.2.1 Authentication Header (AH)

Note

The AH protocol provides source authentication and data integrity, but not privacy.

IPSec Authentication Header (AH)

One of the two core security protocols in IPSec is the *Authentication Header (AH)*.

AH is a protocol that provides *authentication* of either all or part of the contents of a datagram through the addition of a *header* that is calculated based on the values in the datagram. What parts of the datagram are used for the calculation, and the placement of the header, depends on the mode (tunnel or transport) and the version of IP (IPv4 or IPv6).

The operation of the AH protocol is surprisingly simple—especially for any protocol that has anything to do with network security.

It can be considered analogous to the algorithms used to calculate checksums or perform CRC checks for error detection. In those cases, a standard algorithm is used by the sender to compute a checksum or CRC code based on the contents of a message. This computed result is transmitted along with the original data to the destination, which repeats the calculation and discards the message if any discrepancy is found between its calculation and the one done by the source.

This is the same idea behind AH, except that instead of using a simple algorithm known to everyone, we use a special hashing algorithm and a specific key known only to the source and the destination.

A security association between two devices is set up that specifies these particulars so that the source and destination know how to perform the computation but nobody else can.

On the source device, AH performs the computation and puts the result (called the *Integrity Check Value* or *ICV*) into a special header with other fields for transmission.

The destination device does the same calculation using the key the two devices share, which enables it to see immediately if any of the fields in the original datagram were modified (either due to error or malice).

It's important that I point out explicitly that just as a checksum doesn't change the original data, neither does the ICV calculation change the original data. The presence of the AH header allows us to verify the integrity of the message, but doesn't encrypt it. Thus, AH provides ***authentication*** but not ***privacy*** (that's what ESP is for. No, I don't mean using a psychic, I mean the other IPSec core protocol!)

IPSec Authentication Header (AH)

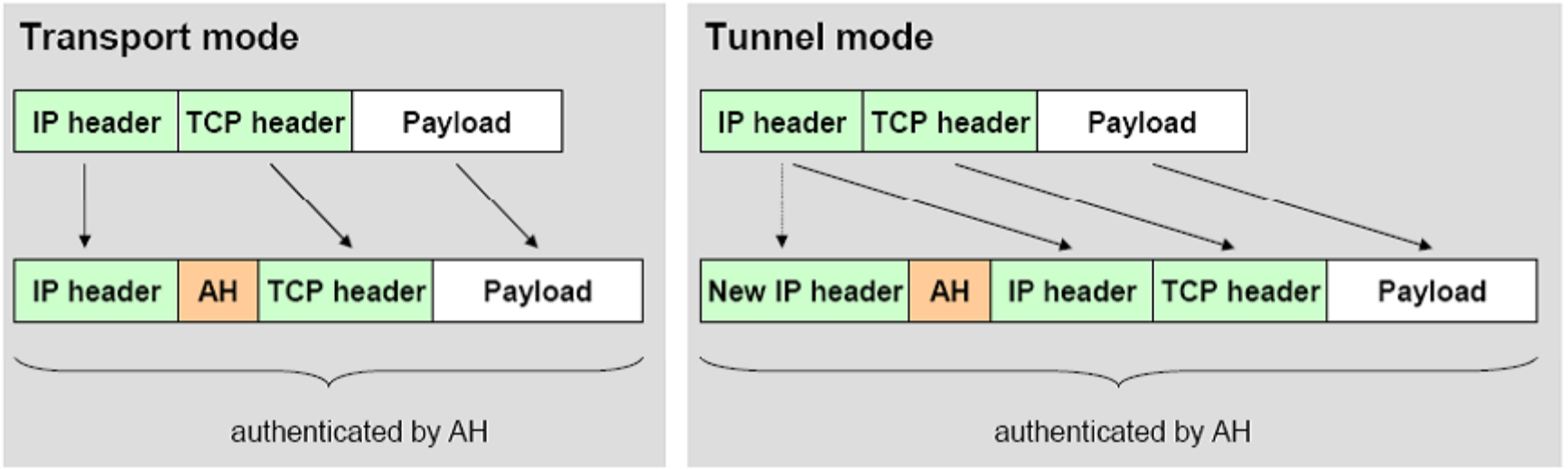
Authentication Header Datagram Placement and Linking:

The calculation of the authentication header is similar for both IPv4 and IPv6. One difference is in the exact mechanism used for placing the header into the datagram and for linking the headers together. I'll describe IPv6 first since it is simpler, as AH was really designed to fit into IPv6's mechanism for this.

IPv6 Authentication Header Placement and Linking: The AH is inserted into the IP datagram as an extension header, following the normal IPv6 rules for extension header linking.
It is linked by the previous header (extension or main) putting into its *Next Header* field the assigned value for the AH header (51). The AH header then links to the next extension header or the transport layer header using its *Next Header* field.
In transport mode, the AH is placed into the main IP header and appears before any *Destination Options* header containing options intended for the final destination, and before an *ESP* header if present, but after any other extension headers. In tunnel mode, it appears as an extension header of the new IP datagram that encapsulates the original one being tunneled. This is shown graphically in Figure 121

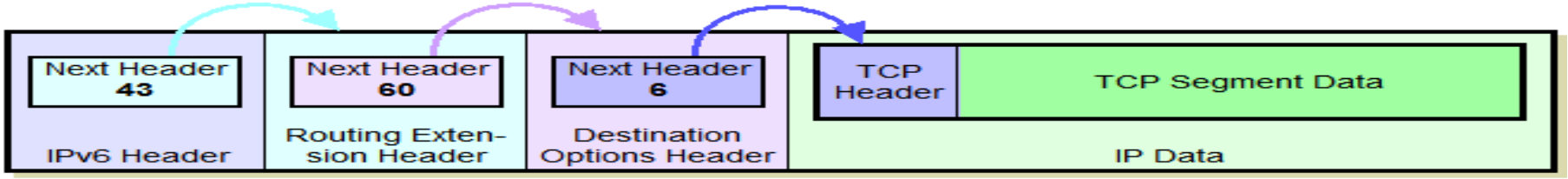
Use of the Authentication Header

AH can be used in two ways:

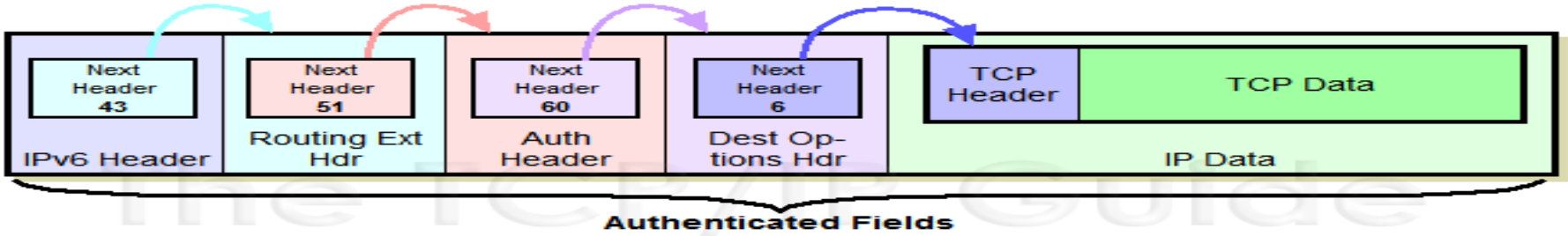


In general, AH has the following characteristics

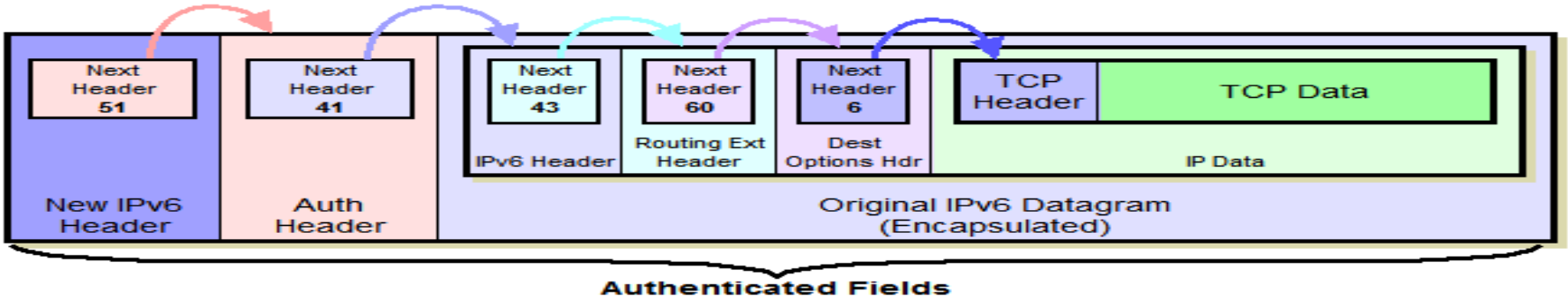
- Advantage: AH mechanism does not significantly increase implementation costs
- Disadvantage: AH increases IP processing costs and communication latency in participating systems



Original IPv6 Datagram Format (Including Routing Extension Header and Destination-Specific Destination Options Extension Header)



IPv6 AH Datagram Format - IPSec Transport Mode



IPv6 AH Datagram Format - IPSec Tunnel Mode

Figure 121: IPv6 Datagram Format With IPSec Authentication Header (AH):

At top is an example IPv6 datagram with two extension headers linked using the standard IPv6 mechanism (see Figure 106.) When AH is applied in transport mode, it is simply added as a new extension header (shown in pink) that goes between the *Routing* extension header and the *Destination Options* header.

In tunnel mode, the entire original datagram is encapsulated into a new IPv6 datagram that contains the Authentication Header.

In both cases the *Next Header* fields are used to link each header one to the next.

Note the use of Next Header value 41 in tunnel mode, which is the value for the encapsulated IPv6 datagram.

IPSec Authentication Header (AH)

IPv4 Authentication Header Placement and Linking

A method that is similar to the IPv6 header linking technique is employed.

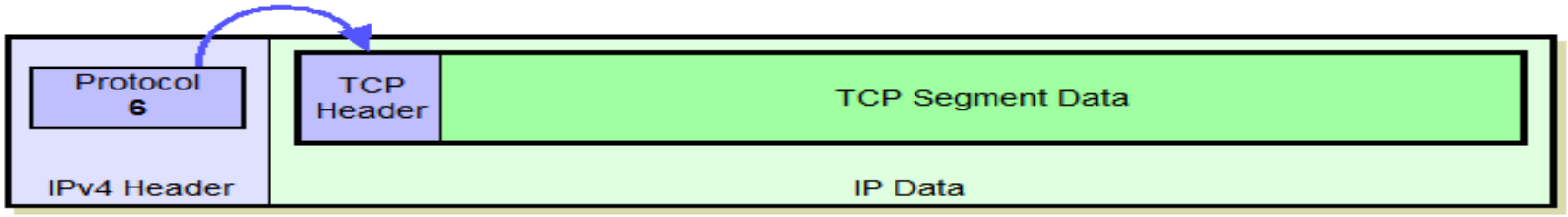
In an IPv4 datagram, the *Protocol* field indicates the identity of the higher layer protocol (typically TCP or UDP) carried in the datagram.

As such, this field “points” to the next header, which is at the front of the IP payload.

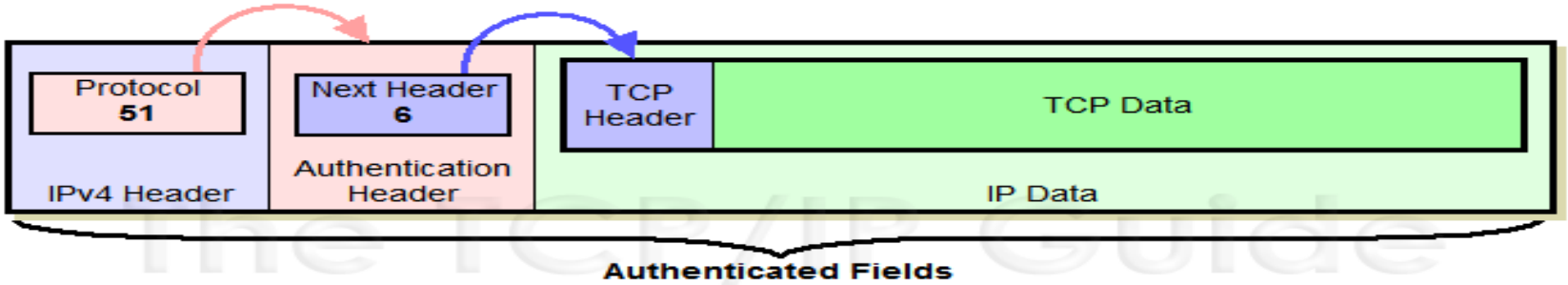
AH takes this value and puts it into its *Next Header* field, and then places the protocol value for AH itself (51 decimal) into the IP *Protocol* field.

This makes the IP header “point” to the AH, which then “points” to whatever the IP datagram pointed to before.

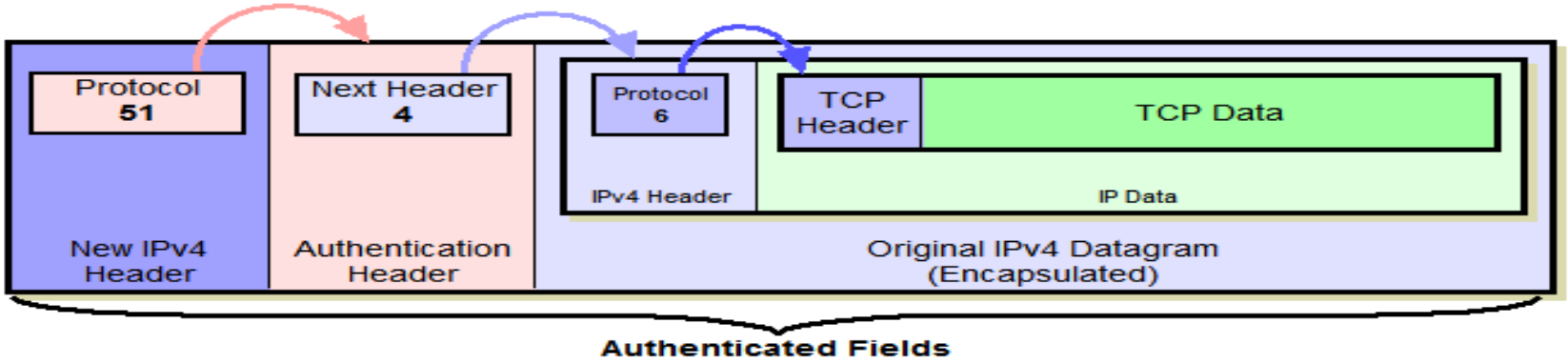
Again, in transport mode, the authentication header is added after the main IP header of the original datagram; in tunnel mode it is added after the new IP header that encapsulates the original datagram being tunneled. This is shown in Figure 122.



Original IPv4 Datagram Format



IPv4 AH Datagram Format - IPSec Transport Mode



IPv4 AH Datagram Format - IPSec Tunnel Mode

Figure 122: IPv4 Datagram Format With IPSec Authentication Header (AH)

At top is an example IPv4 datagram; it may or may not contain IPv4 options (which are not distinct entities as they are in IPv6).

In transport mode, the authentication header is added between the IP header and the IP data; the *Protocol* field of the IP header points to it, while its *Next Header* field contains the IP header's prior protocol value (in this case 6, for TCP.)

In tunnel mode the IPv4 datagram is encapsulated into a new IPv4 datagram that includes the AH header. Note in tunnel mode, the AH header's use of the value 4 (which means IPv4) in its *Next Header*.

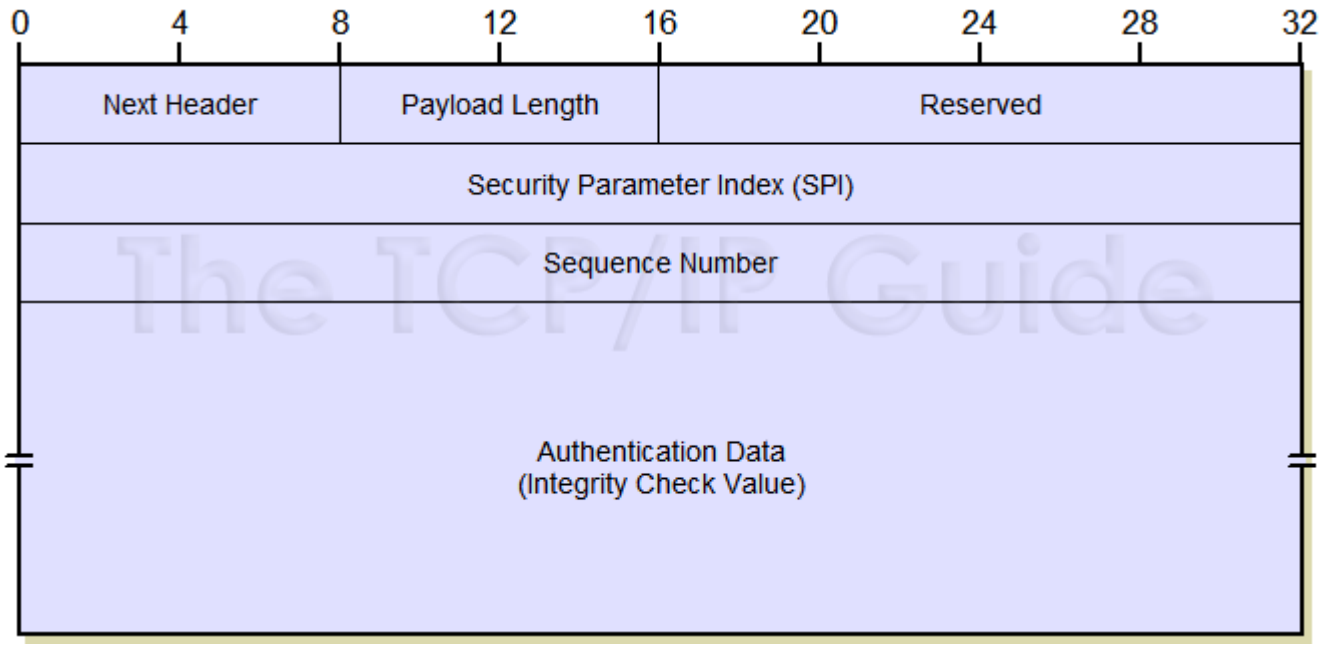
IPSec Authentication Header (AH)

Authentication Header Format

The format of the Authentication Header itself is described in Table 79 and shown in Figure 123.

Table 79: IPSec Authentication Header (AH) Format

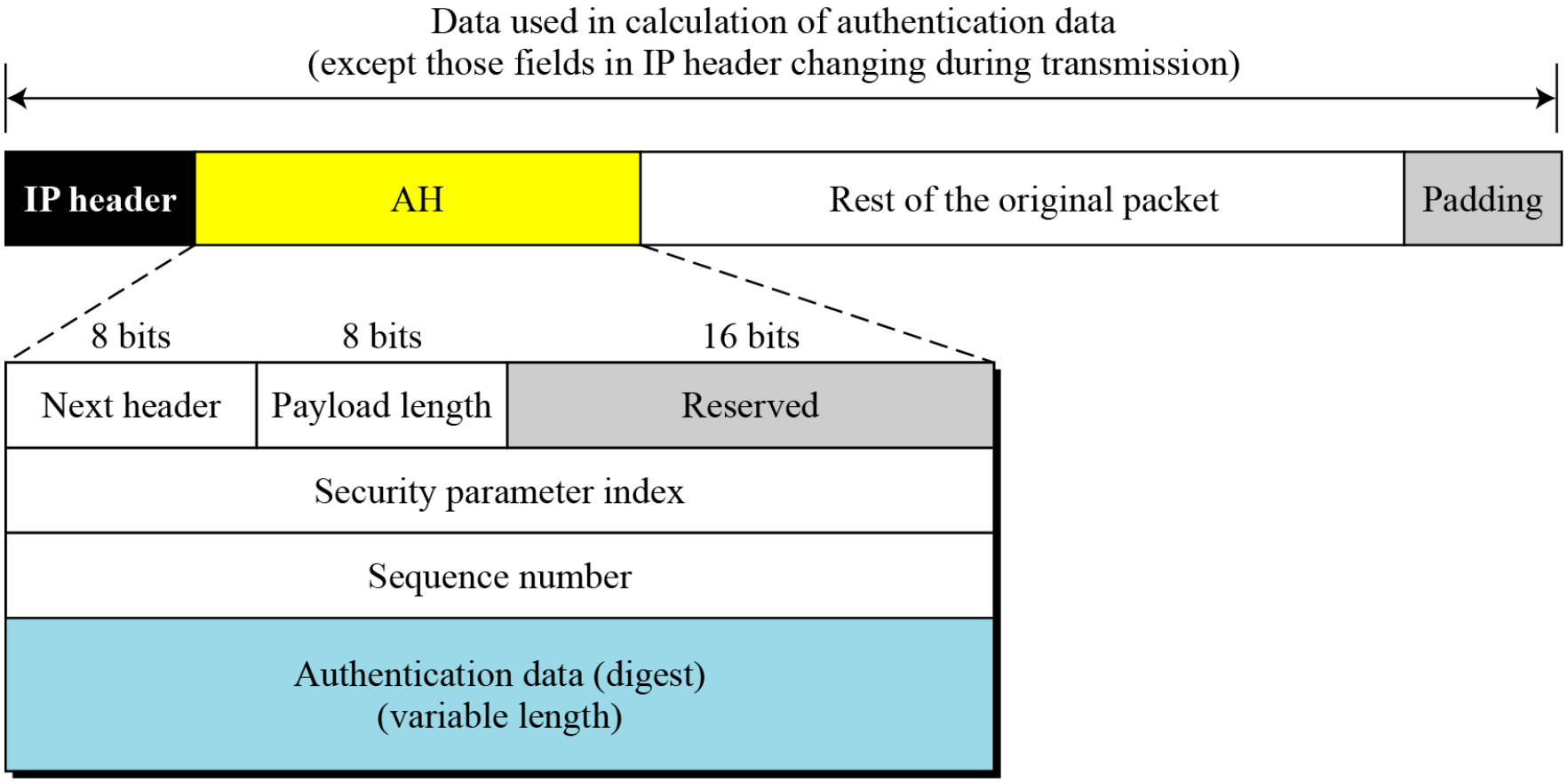
Field Name	Size (bytes)	Description
Next Header	1	***Next Header:*** Contains the protocol number of the next header after the AH. Used to link headers together.
Payload Len	1	***Payload Length:*** Despite its name, this field measures the length of the authentication header itself, not the payload. (I wonder what the history is behind that!) It is measured in 32 bit units, with 2 subtracted for consistency with how header lengths are normally calculated in IPv6.
Reserved	2	***Reserved:*** Not used; set to zeroes.
SPI	4	***Security Parameter Index (SPI):*** A 32-bit value that when combined with the destination address and security protocol type (which here is obviously the one for AH) identifies the security association to be used for this datagram. See the topic on security associations for more details.
Sequence Number	4	***Sequence Number:*** This is a counter field that is initialized to zero when a security association is formed between two devices, and then incremented for each datagram sent using that SA. This uniquely identifies each datagram on an SA and is used to provide protection against replay attacks by preventing the retransmission of captured datagrams.
Authentication Data	Variable	***Authentication Data:*** This field contains the result of the hashing algorithm performed by the AH protocol, the Integrity Check Value (ICV).



The size of the *Authentication Data* field is variable to support different datagram lengths and hashing algorithms. Its total length must be a multiple of 32 bits. Also, the entire header must be a multiple of either 32 bits (for IPv4) or 64 bits (for IPv6), so additional padding may be added to the *Authentication Data* field if necessary.

You may also notice that no IP addresses appear in the header, which is a prerequisite for it being the same for both IPv4 and IPv6.

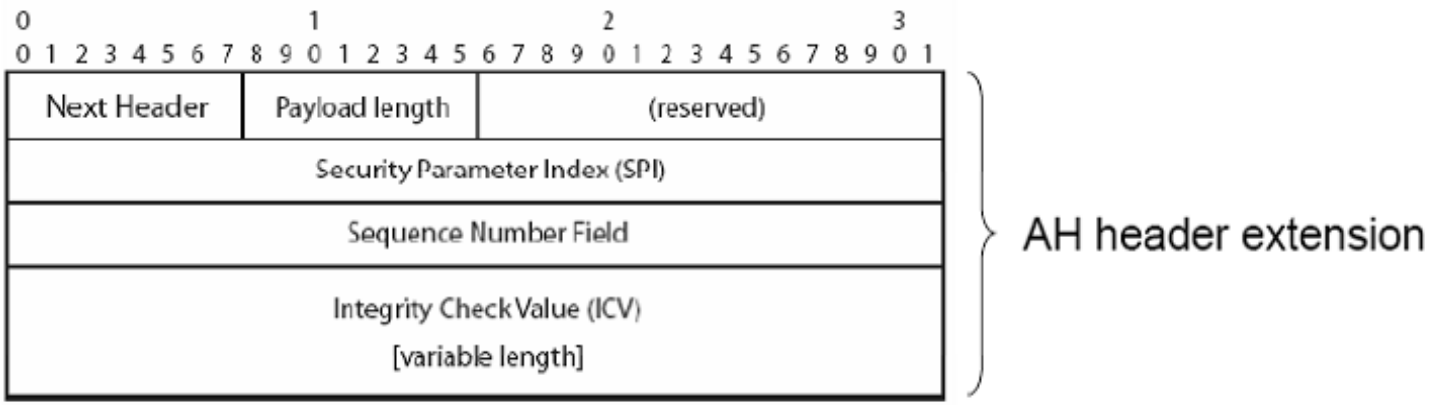
Figure 18.7 *Authentication Header (AH) protocol*



Summary of AH

Authentication Header (AH)

- AH provides *sender authentication* and *data integrity* for IP packets by enhancing the IP header with some additional fields:



- Next Header: type of the following header after the AH (as the protocol field in IPv4)
- Payload Length: length of the integrity check value (ICV) in 32-bit words
- Reserved: for future use, currently set to zero
- SPI: identifies the SA for the IP packet on receiver side
- Sequence number: assign each packet an unique identifier to protect against replay attacks
- ICV: the assigned authentication/integrity value

Summary of AH

Integrity Check Value

- The authentication data mostly is computed by using a cryptographic authentication algorithm and a corresponding secret key
  - Use a one-way hash function, MD5 is default in IPSec
  - The ICV in case of MD 5 is a 128-bit hash value from the concatenation of the key, the message, and the key again
  - Alternative: use digital signatures from public key cryptography
  - In general, the authentication algorithm is negotiated as part of the SA
  - The receiver uses the same algorithm to test the ICV

- Problem: some fields of the IP packet header change in transit, e.g. the TTL field in IPv4
  - The ICV is computed for a whole IP packet
  - With another TTL, the receiver comes to another ICV as the sender
  - Thus, the receiver has to set such information like TTL back to the initial value before computing and testing the ICV value

Note: AH does not encrypt the payload!

18.2.2 Encapsulating Security Payload (ESP)

Note

ESP provides source authentication, data integrity, and privacy.

IPSec Encapsulating Security Payload (ESP)

The IPSec Authentication Header (AH) provides integrity authentication services to IPSec-capable devices, so they can verify that messages are received intact from other devices.

For many applications, however, this is only one piece of the puzzle. We want to not only protect against intermediate devices changing our datagrams, we want to protect against them examining their contents as well.

For this level of private communication, AH is not enough; we need to use the *Encapsulating Security Payload (ESP)* protocol.

The main job of ESP is to provide the privacy we seek for IP datagrams by *encrypting* them. An encryption algorithm combines the data in the datagram with a key to transform it into an encrypted form. This is then repackaged using a special format that we will see shortly, and transmitted to the destination, which decrypts it using the same algorithm. ESP also supports its own authentication scheme like that used in AH, or can be used in conjunction with AH.

Encapsulating Security Payload Fields: ESP has several fields that are the same as those used in AH, **but packages its fields in a very different way.**

Instead of having just a header, it divides its fields into three components:

- ***ESP Header:*** This contains two fields, the *SPI* and *Sequence Number*, and comes before the encrypted data. Its placement depends on whether ESP is used in transport mode or tunnel mode, as explained in the topic on IPSec modes.

- ***ESP Trailer:*** This section is placed after the encrypted data. It contains padding that is used to align the encrypted data, through a *Padding* and *Pad Length* field. Interestingly, it also contains the *Next Header* field for ESP.

- ***ESP Authentication Data:*** This field contains an *Integrity Check Value (ICV)*, computed in a manner similar to how the AH protocol works, for when ESP's optional authentication feature is used.

There are two reasons why these fields are broken into pieces like this.

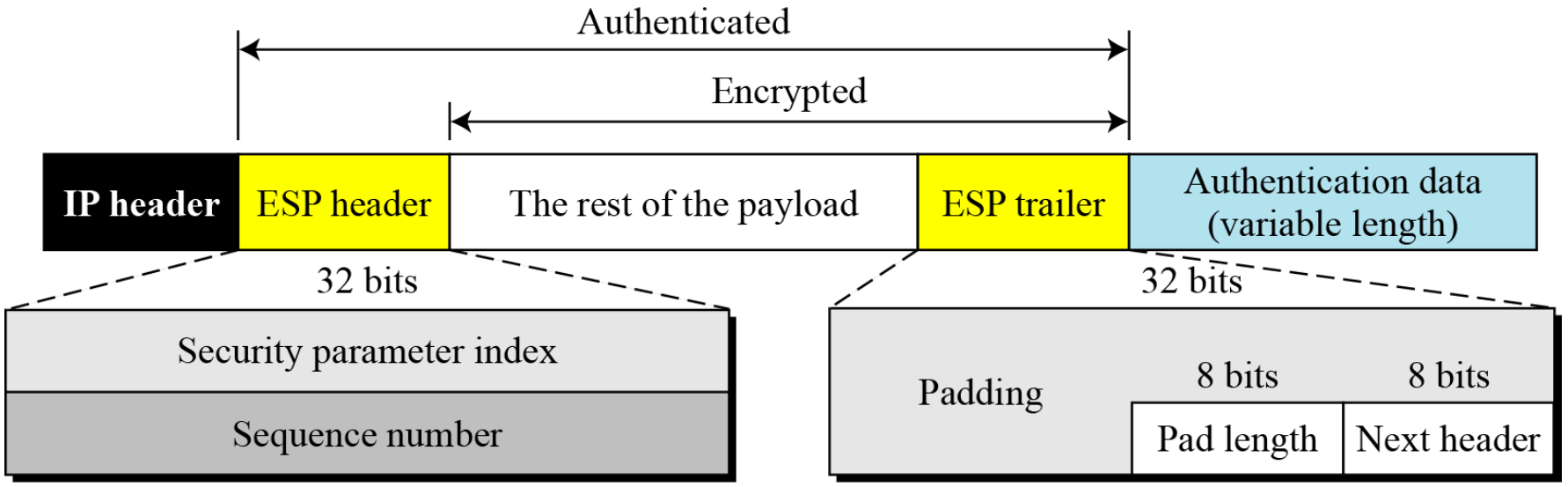
The first is that some encryption algorithms require the data to be encrypted to have a certain block size, and so padding must appear after the data and not before it.

That's why padding appears in the ESP Trailer.

The second is that the *ESP Authentication Data* appears separately because it is used to authenticate the rest of the encrypted datagram *after encryption*.

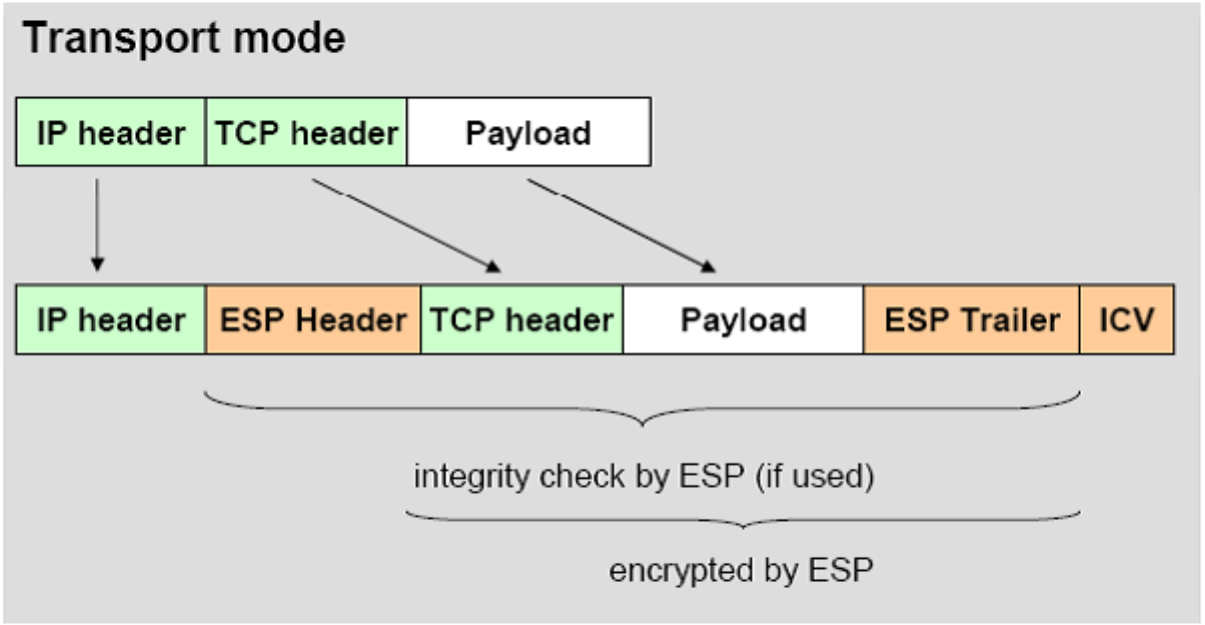
This means it cannot appear in the ESP Header or ESP Trailer.

Figure 18.8 *ESP*



Use of the Encapsulating Security Payload

ESP can be used in two ways, as AH:



- The receiver processes the IP header and plaintext part of the ESP to obtain the SPI value
- This value is used as an index for a local SPI table to find the negotiated SA parameters and cryptographic keys to decrypt the rest of the packet

IPSec Encapsulating Security Payload (ESP)

Encapsulating Security Payload Operations and Field Use

Let's try to explain this procedurally, by considering three basic steps performed by ESP.

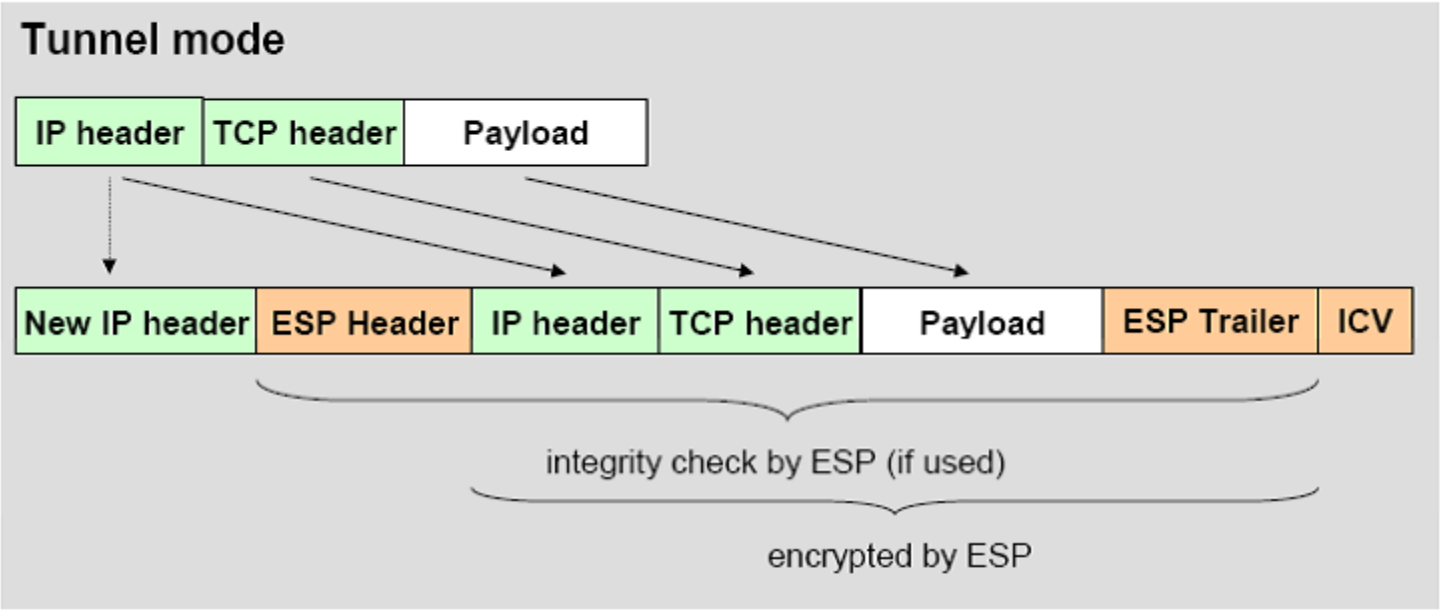
1. Header Calculation and Placement: The first thing to consider is how the ESP header is placed; this is similar to how AH works:

IPv6: The *ESP Header* is inserted into the IP datagram as an extension header, following the normal IPv6 rules for extension header linking.

In transport mode, it appears before a *Destination Options* header containing options intended for the final destination, but after any other extension headers, if present.

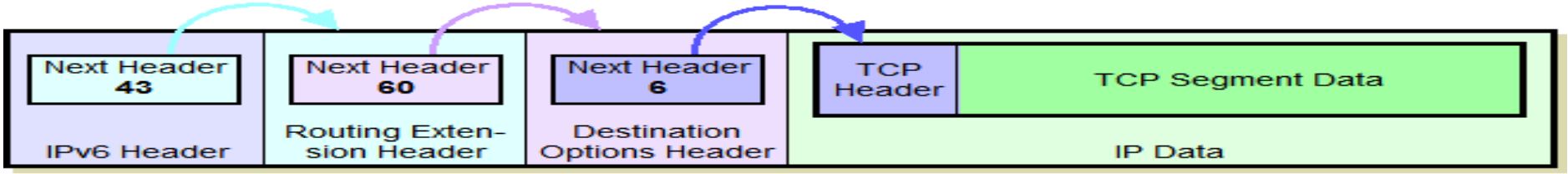
In tunnel mode, it appears as an extension header of the new IP datagram that encapsulates the original one being tunneled. This can be seen in Figure 124.

Use of the Encapsulating Security Payload

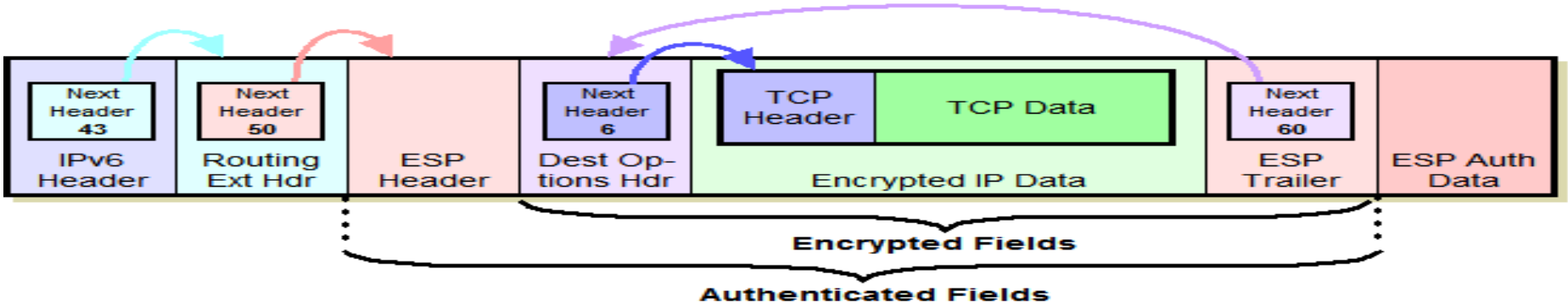


- In tunnel mode, an encrypted tunnel between two security gateways is installed
- Used e.g. between two LANs which should communicate in a secure manner (as in VPN, Virtual private Network)

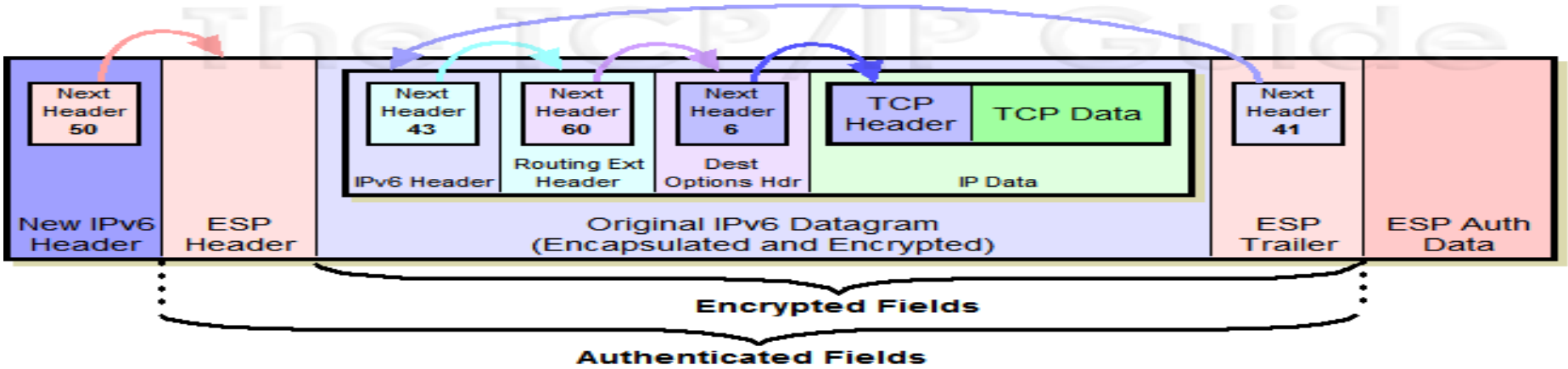
Note: AH and ESP mechanism have been designed independently and can be applied separately or together



Original IPv6 Datagram Format (Including Routing Extension Header and Destination-Specific Destination Options Extension Header)



IPv6 ESP Datagram Format - IPSec Transport Mode



IPv6 ESP Datagram Format - IPSec Tunnel Mode

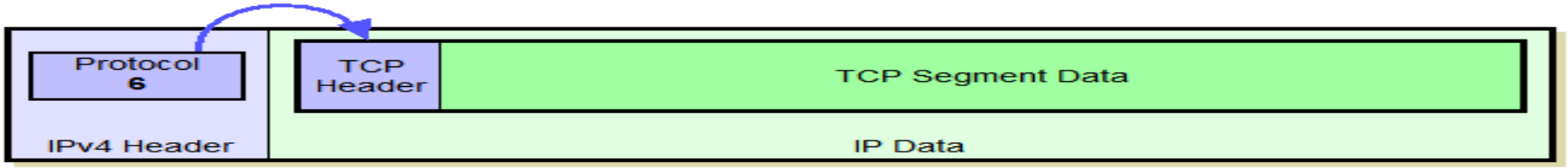
Figure 124: IPv6 Datagram Format With IPSec Encapsulating Security Payload (ESP)

At top is the same example IPv6 datagram with two extension headers shown in Figure 121. When ESP is applied in transport mode, the *ESP Header* is added to the existing datagram as in AH, and the *ESP Trailer* and *ESP Authentication Data* are placed at the end.

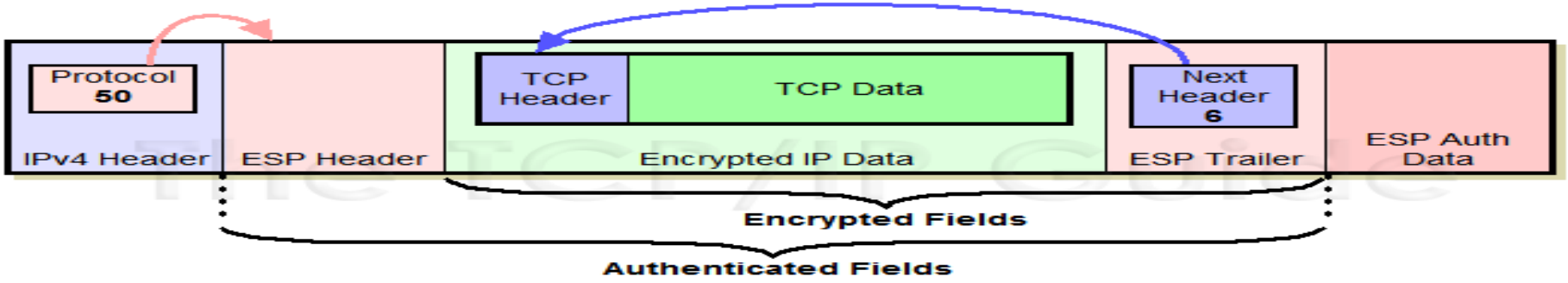
In tunnel mode, the *ESP Header* and *Trailer* bracket the entire encapsulated IPv6 datagram.

Note the encryption and authentication coverage in each case, and also how the *Next Header* field “points back” into the datagram since it appears in the ESP Trailer.

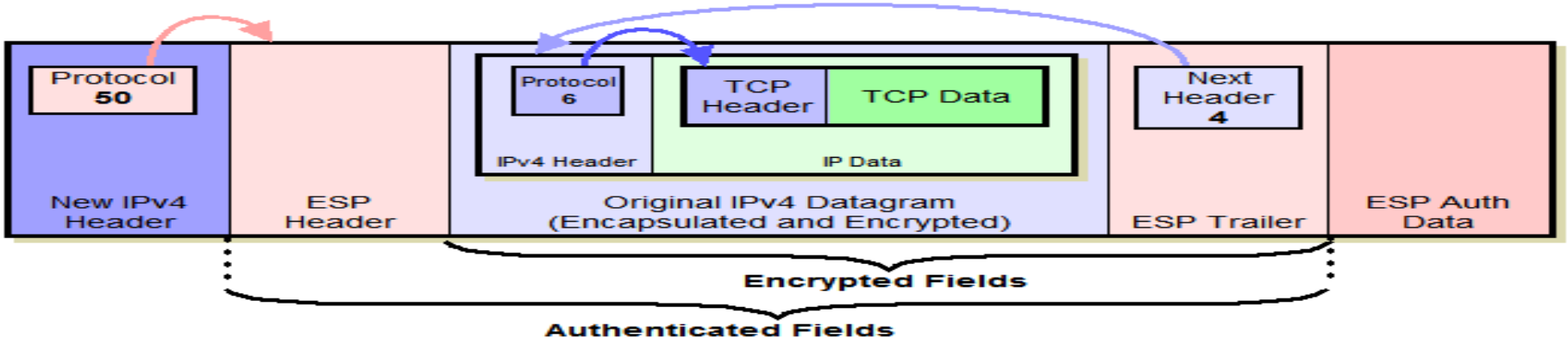
IPv4: As with the AH, the *ESP Header* is placed after the normal IPv4 header. In transport mode, it appears after the original IP header; in tunnel mode, after the IP header of the new datagram encapsulating the original. This is shown in Figure 125.



Original IPv4 Datagram Format



IPv4 ESP Datagram Format - IPSec Transport Mode



IPv4 ESP Datagram Format - IPSec Tunnel Mode

Figure 125: IPv4 Datagram Format With IPSec Encapsulating Security Payload (ESP)

At top is the same sample IPv4 datagram shown in Figure 122.

When this datagram is processed by ESP in transport mode, the ESP Header is placed between the IPv4 header and data, with the ESP Trailer and ESP Authentication Data following.

In tunnel mode, the entire original IPv4 datagram is surrounded by these ESP components, rather than just the IPv4 data. Again, as in Figure 124, note the encryption and authentication coverage, and how the Next Header field "points back" to specify the identity of the encrypted data/datagram.

IPSec Encapsulating Security Payload (ESP)

2. Trailer Calculation and Placement

The *ESP Trailer* is appended to the data to be encrypted. ESP then performs the encryption. The payload (TCP/UDP message or encapsulated IP datagram) and the ESP trailer are both encrypted, but the *ESP Header* is not.

Note again that any other IP headers that appear between the ESP header and the payload are also encrypted. In IPv6 this can include a *Destination Options* extension header.

Normally, the *Next Header* field would appear in the ESP header and would be used to link the ESP header to the header that comes after it. However, the *Next Header* field in ESP appears in the trailer and not the header, which makes the linking seem a bit strange in ESP. The method is the same as that used in AH and in IPv6 in general, with the *Next Header* and/or *Protocol* fields used to tie everything together. However, in ESP the *Next Header* field appears *after* the encrypted data, and so "points back" to one of the following: a *Destination Options* extension header (if present), a TCP/UDP header (in transport mode) or an IPv4/IPv6 header (in tunnel mode). This too is shown in Figure 124 and Figure 125.

3. ESP Authentication Field Calculation and Placement: If the optional ESP authentication feature is used, the authentication field is computed over the entire ESP datagram (except the *Authentication Data* field itself, of course). This includes the ESP header, payload and trailer.

Key Concept: The IPSec *Encapsulating Security Payload* protocol allows the contents of a datagram to be encrypted, to ensure that only the intended recipient is able to see the data.

It is implemented using three components: an *ESP Header* added to the front of a protected datagram, an *ESP Trailer* that follows the protected data, and an optional *ESP Authentication Data* field that provides authentication services similar to those provided by the Authentication Header (AH).

IPSec Encapsulating Security Payload (ESP)

Encapsulating Security Payload Format

The format of the ESP sections and fields is described in Table 80 and shown in Figure 126. I have shown explicitly in each the encryption and authentication coverage of the fields, which will hopefully cause all that stuff I just wrote to make at least a bit more sense.

Table 80: IPSec Encapsulating Security Payload (ESP) Format

Section	Field Name	Size (bytes)	Description	Encryption Coverage	Authentication Coverage
ESP Header	*SPI*	4	**Security Parameter Index (SPI):** A 32-bit value that is combined with the destination address and security protocol type to identify the security association to be used for this datagram. See the topic on security associations for more details.		
	Sequence Number	4	**Sequence Number:** A counter field initialized to zero when a security association is formed between two devices, and then incremented for each datagram sent using that SA. This is used to provide protection against replay attacks.		
Payload	*Payload Data*	Variable	**Payload Data:** The encrypted payload data, consisting of a higher layer message or encapsulated IP datagram. May also include support information such as an initialization vector, required by certain encryption methods.		
ESP Trailer	*Padding*	Variable (0 to 255)	**Padding:** Additional padding bytes included as needed for encryption or for alignment.		
	Pad Length	1	**Pad Length:** The number of bytes in the preceding Padding field.		
	Next Header	1	**Next Header:** Contains the protocol number of the next header in the datagram. Used to chain together headers.		
ESP Authentication Data		Variable	**ESP Authentication Data:** This field contains the *Integrity Check Value (ICV)* resulting from the application of the optional ESP authentication algorithm.		

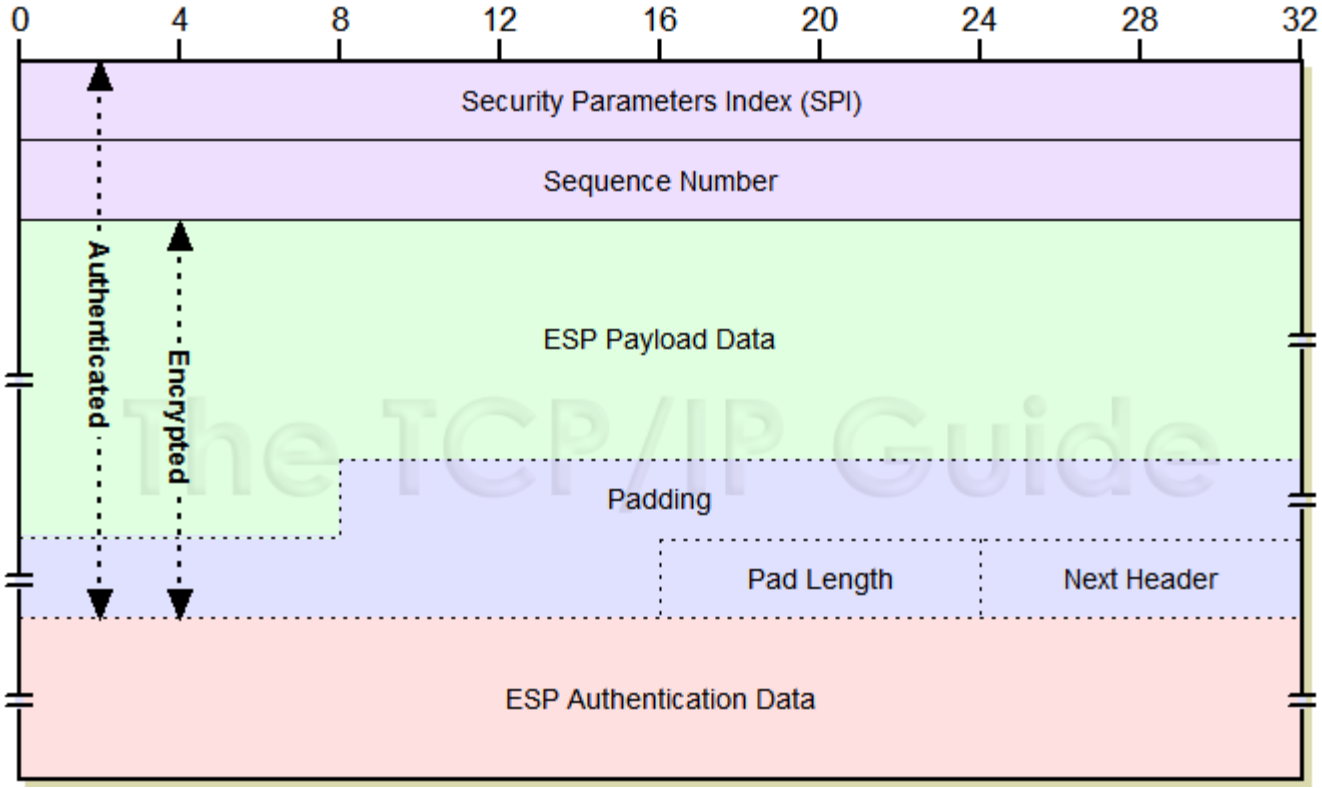


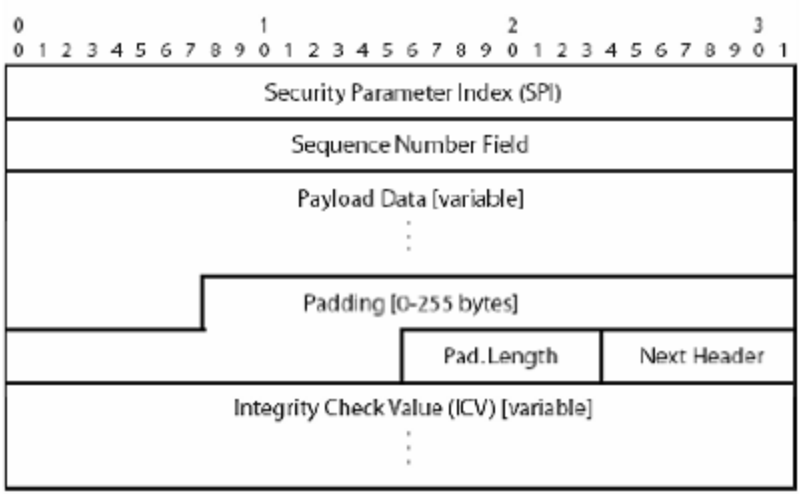
Figure 126: IPSec Encapsulating Security Payload (ESP) Format
Note that most of the fields and sections in this format are variable length. The exceptions are the *SPI* and *Sequence Number* fields, which are 4 bytes long, and the *Pad Length* and *Next Header* fields, 1 byte each.

The *Padding* field is used when encryption algorithms require it. *Padding* is also used to make sure that the *ESP Trailer* ends on a 32-bit boundary. That is, the size of the *ESP Header* plus *Payload* plus *ESP Trailer* must be a multiple of 32 bits. The *ESP Authentication Data* must also be a multiple of 32 bits.

Summary of ESP

Encapsulating Security Payload (ESP)

ESP provides data confidentiality by using encryption and encapsulation and adding an ESP header/trailer to an IP packet:



- SPI, Sequence Number, and ICV (only if integrity is also checked): as in AH
- Payload data: encrypted with an algorithm defined in the SA (default: DES in CBC mode)
- Padding: filled with random bits
- Padding Length: indicates the total length of the Padding field
- Next header: identifies the following header

18.2.3 IPv4 and IPv6

IPSec supports both IPv4 and IPv6. In IPv6, however, AH and ESP are part of the extension header.

18.2.4 AH versus ESP

The ESP protocol was designed after the AH protocol was already in use. ESP does whatever AH does with additional functionality (privacy).

18.2.5 Services Provided by IPSec

Table 18.1 *IPSec services*

Services	*AH*	*ESP*
Access control	yes	yes
Message authentication (message integrity)	yes	yes
Entity authentication (data source authentication)	yes	yes
Confidentiality	**no**	yes
Replay attack protection	yes	yes

IPsec Packet Flow

In next slide figure shows how an IP packet proceeds when IPsec has been invoked on an outbound packet.

The flow diagram illustrates where authentication header (AH) and encapsulating security payload (ESP) entities can be applied to the packet.

Subsequent sections describe how to apply these entities, as well as how to choose the algorithms.

Figure shows the IPsec inbound process.

Figure: IPsec Applied to Outbound Packet Process

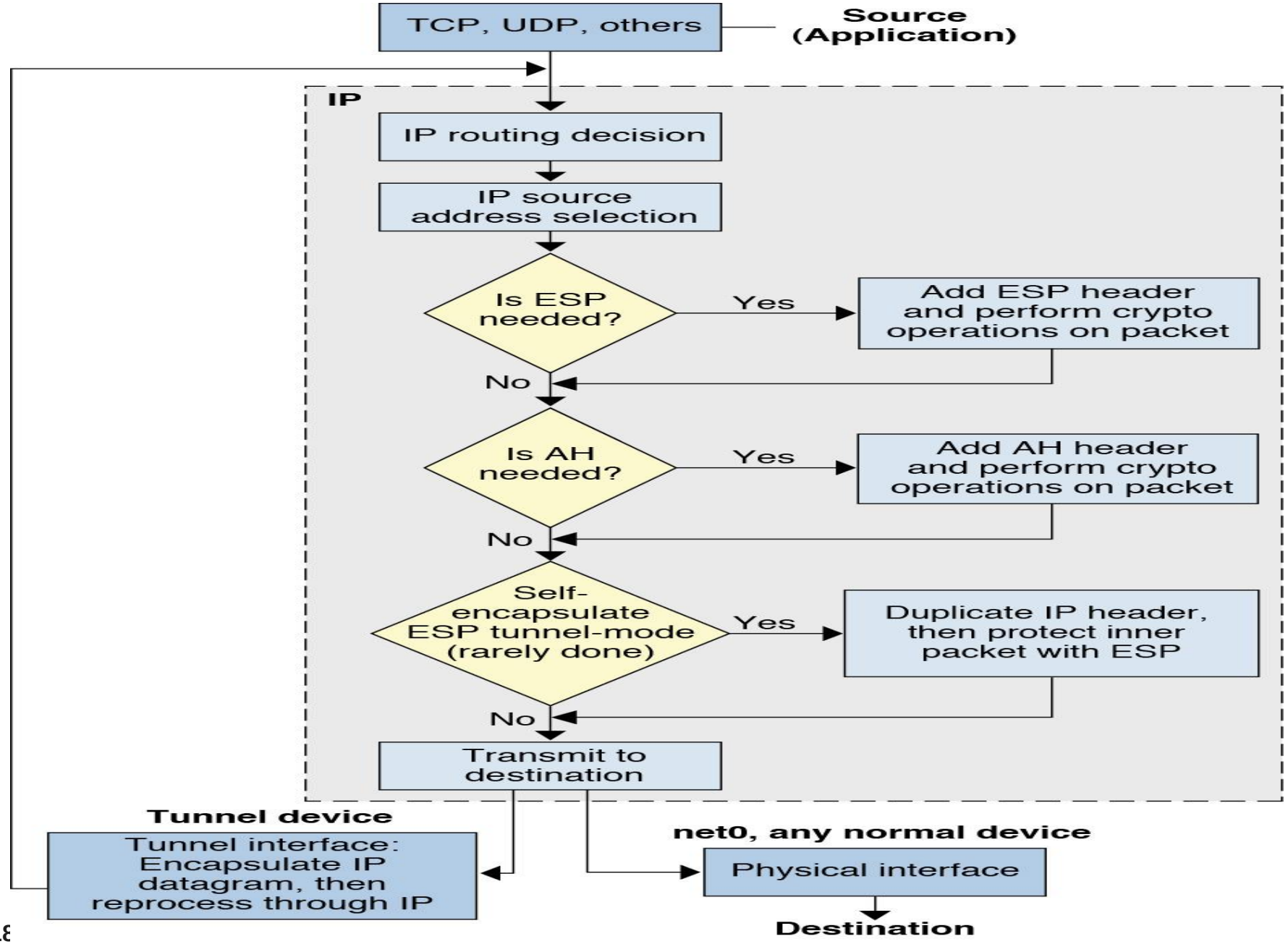
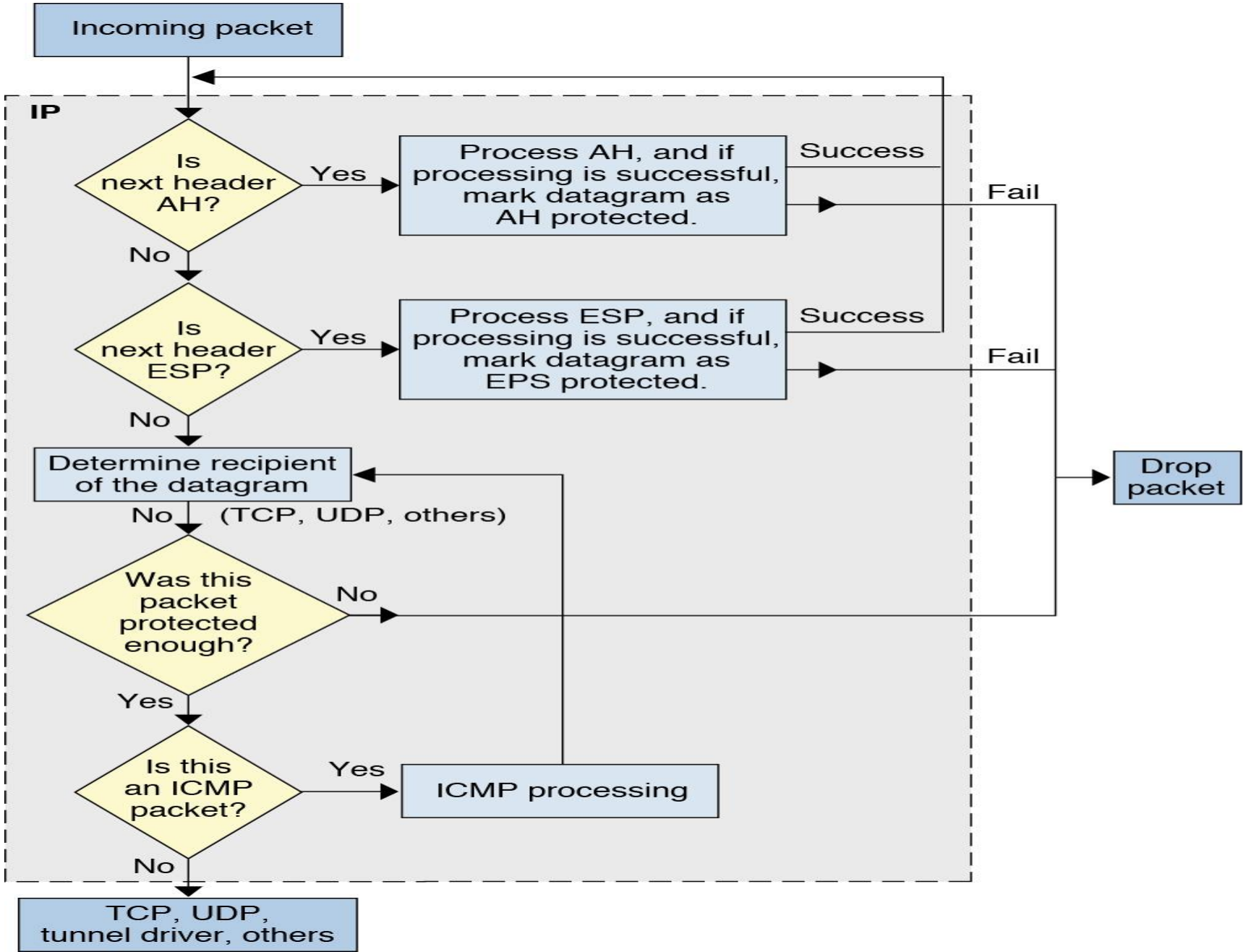


Figure IPsec Applied to Inbound Packet Process



18-5 INTERNET KEY EXCHANGE (IKE)

The Internet Key Exchange (IKE) is a protocol designed to create both inbound and outbound Security Associations.

Topics discussed in this section:

Note

IKE creates SAs for IPSec.

IPSec Key Exchange (IKE)

IPSec, like many secure networking protocol sets, is based on the concept of a “shared secret”. Two devices that want to send information securely encode and decode it using a piece of information that only they know.

Anyone who isn't “in” on the secret is able to intercept the information but is prevented either from reading it (if ESP is used to encrypt the payload) or from tampering with it undetected (if AH is used).

Before either AH or ESP can be used, however, it is necessary for the two devices to exchange the “secret” that the security protocols themselves will use.

The primary support protocol used for this purpose in IPSec is called *Internet Key Exchange (IKE)*.

IKE Overview and Relationship to Other Key Exchange Methods:

The purpose of IKE is to allow devices to exchange information required for secure communication.

As the title suggests, this includes cryptographic keys used for encoding authentication information and performing payload encryption.

IKE works by allowing IPSec-capable devices to exchange security associations (SAs), to populate their security association databases (SADs).

These are then used for the actual exchange of secured datagrams with the AH and ESP protocols.

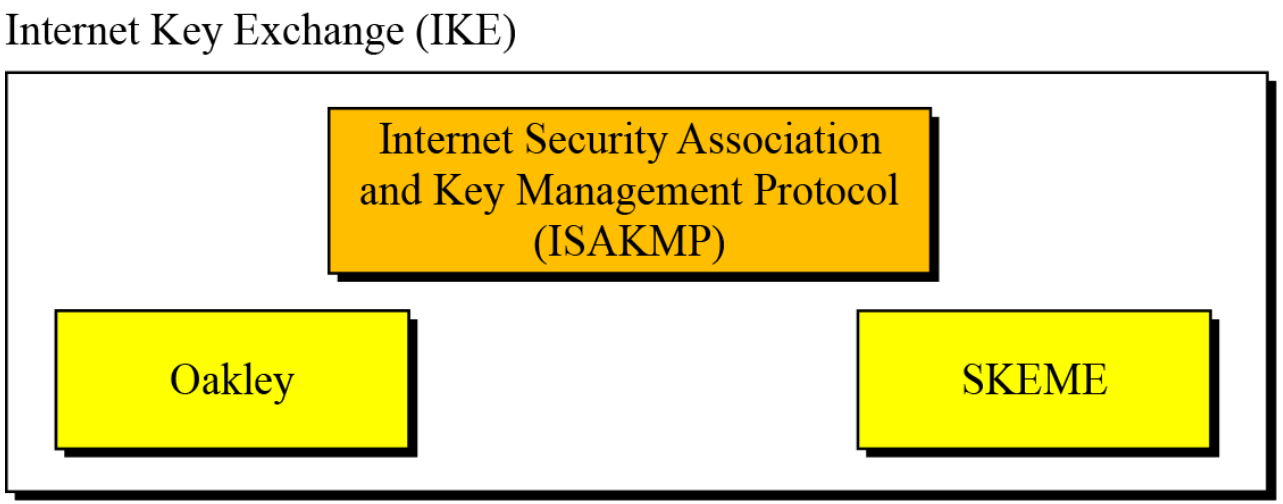
IKE is considered a “hybrid” protocol because it combines (and supplements) the functions of three other protocols. The first of these is the *Internet Security Association and Key Management Protocol (ISAKMP)*. This protocol provides a framework for exchanging encryption keys and security association information. It operates by allowing security associations to be negotiated through a series of phases.

- **OAKLEY:** Describes a specific mechanism for exchanging keys through the definition of various key exchange “modes”. Most of the IKE key exchange process is based on OAKLEY.

- ISAKMP is a generic protocol that supports many different key exchange methods. In IKE, the ISAKMP framework is used as the basis for a specific key exchange method that combines features from two key exchange protocols.

- **SKEME:** Describes a different key exchange mechanism than OAKLEY. IKE uses some features from SKEME, including its method of public key encryption and its fast re-keying feature

Figure 18.15 *IKE components*



18.5.2 IKE Phases

Note

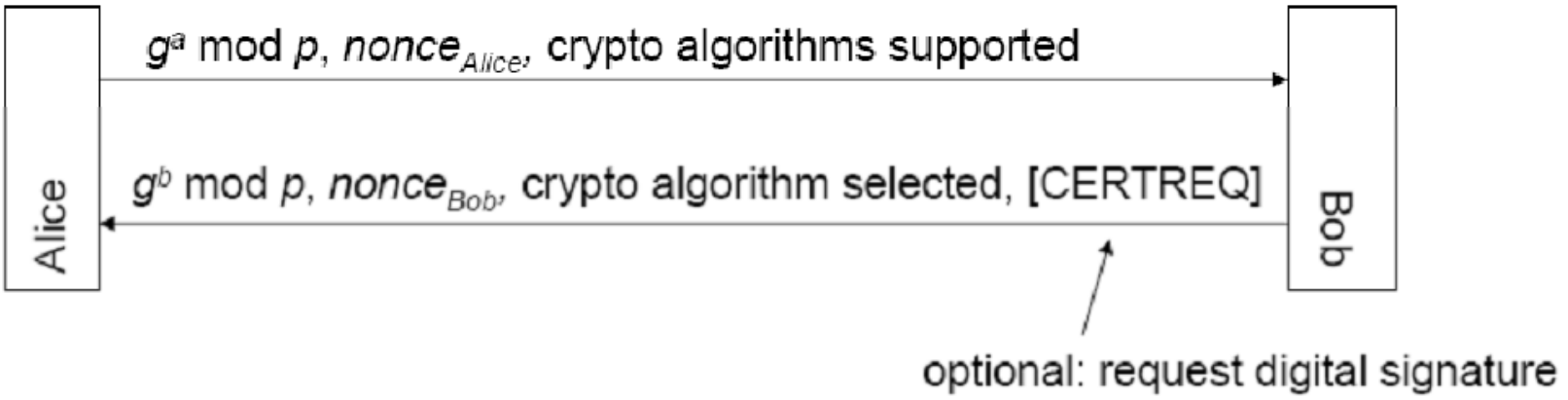
IKE is divided into two phases: phase I and phase II. Phase I creates SAs for phase II; phase II creates SAs for a data exchange protocol such as IPSec..

Internet Key Exchange (IKE)

- Before applying AH and/or ESP, a SA has to be established – this is done using the
- Internet Key Exchange (IKEv2) in two phases:
  - Phase 1: negotiate a SA in two steps
    - Purpose: mutual authentication, establish secret keys for phase 2
  - Phase 2: create multiple SAs used for one communication each
    - Purpose: from the results of phase 1, several SAs can be generated, which gives a speedup (if several SAs are needed)

IKE Phases

- First half of phase 1: establish a common secret using Diffie-Hellman key exchange:



- Second half of phase 1: mutual authentication
  - Using the key negotiated by Diffie-Hellman, now encrypted mutual authentication is done – optionally by using certificates, if requested before
  - In the authentication messages, also IP addresses and TCP ports of communication partners are included
- IKE phase 2: applied after mutual authentication.
  - Use negotiated key to propose/accept SAs – maybe also by doing a new key exchange with every SA proposal to generate new session keys

18.5.1 Improved Diffie-Hellman

Figure 18.16 *Diffie-Hellman key exchange*

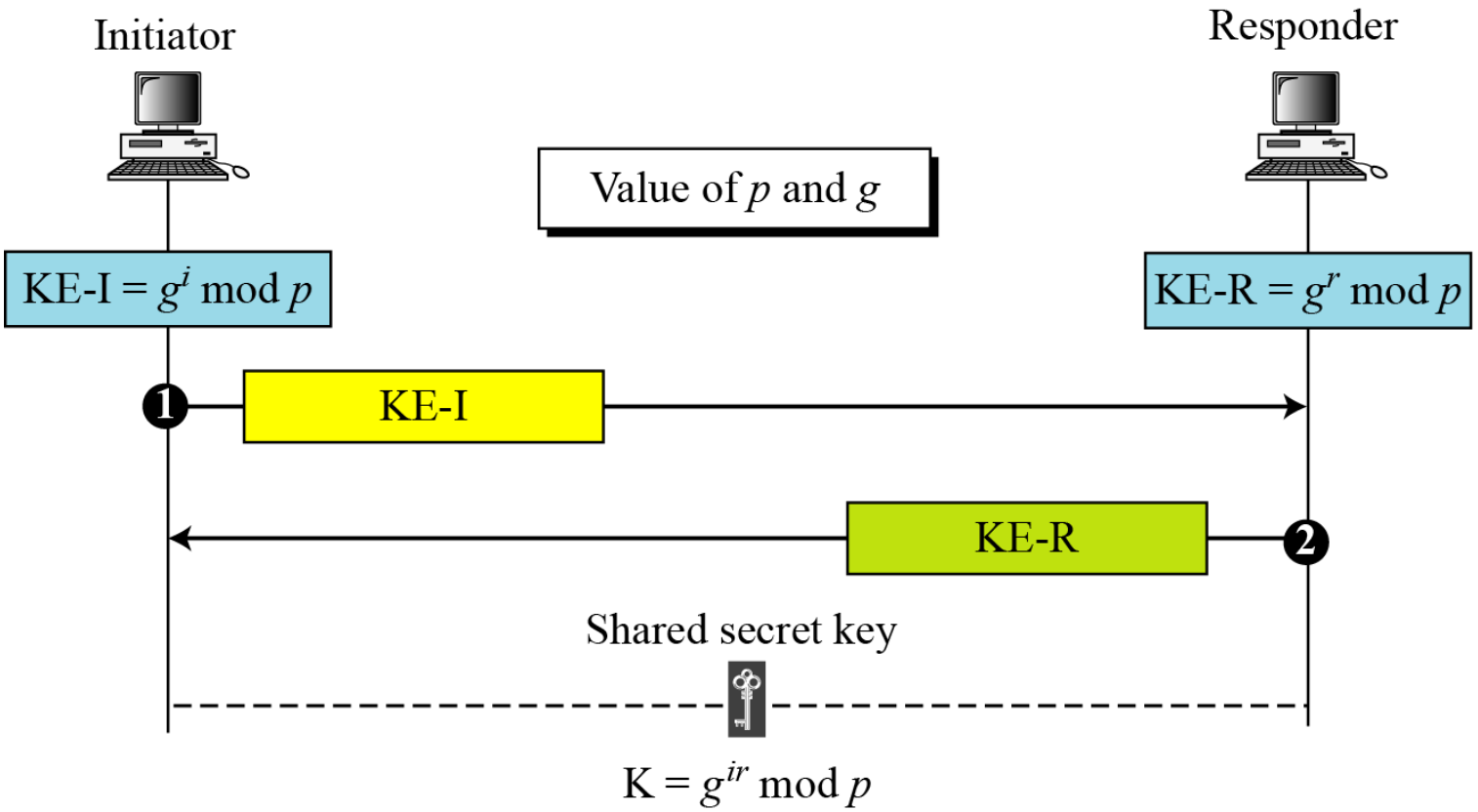


Table 18.4 *Hash Algorithms*

Value	*Description*
1	MD5
2	SHA
3	Tiger
4	SHA2-256
5	SHA2-384
6	SHA2-512

Table 18.5 *Encryption algorithms*

Value	*Description*
1	DES
2	IDEA
3	Blowfish
4	RC5
5	3DES
6	CAST
7	AES

Table 18.7 *Certification types*

Value	*Type*
0	None
1	Wrapped X.509 Certificate
2	PGP Certificate
3	DNS Signed Key
4	X.509 Certificate —Signature
5	X.509 Certificate—Key Exchange
6	Kerberos Tokens
7	Certification Revocation List
8	Authority Revocation List
9	SPKI Certificate
10	X.509 Certificate—Attribute