Chapter 9

Network Security

Network layer security is a key aspect of the Internet-based security mechanisms. Originally, people were concentrating only on application level security. However, new security requirements demand that even the lower level data units should be protected. With this view in mind, network security mechanisms have emerged, and are being used quite extensively in real life.

This chapter discusses TCP/IP protocol suite in brief, with specific attention to the IP and TCP protocols. This is important for understanding the concepts in network security. We then examine the technology of firewalls. Firewalls are widely used by organizations to protect their internal networks from outside attacks. The chapter discusses the various types, organizations and architectures of firewalls. It also takes a look at the various issues involved therein.

The chapter then focuses on network level security, with a thorough description of the IPSec protocol. IPSec protocol has many sub-protocols. We discuss them in great detail. Finally, we study what a Virtual Private Network (VPN) means, and how it works.

9.1 BRIEF INTRODUCTION TO TCP/IP

9.1.1 Basic Concepts

As we know, the Internet is based on the Transmission Control Protocol/Internet protocol (TCP/IP) protocol suite. It is very important to know the basics of TCP/IP before we study how we can provide security at the network layer. We have already discussed this in brief earlier. However, in the context of network level security, a comprehensive understanding of TCP and IP is mandatory.

Let us first take a look at the various layers in the TCP/IP protocol suite. As we have studied before, the TCP/IP protocol suite contains five main layers: application, transport, network (or Internet), data link and physical. Unlike the OSI protocol suite, there are no presentation and session layers in TCP/IP. The various layers and protocols in the TCP/IP suite are shown in Fig. 9.1. Note that the presentation and session layers are not present in TCP/IP, but are shown for the comparison of TCP/IP versus the OSI model.

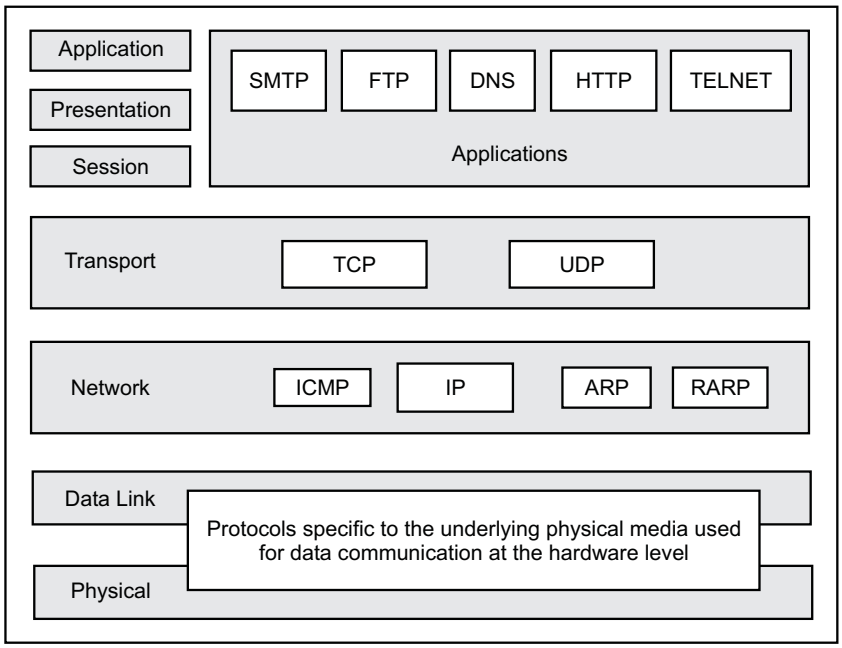

Fig. 9.1 *Layers in the TCP/IP protocol suite*

Let us now think about the significance of these layers. Figure 9.2 shows the idea. The data unit initially created at the application layer (i.e. by an application, such as email, Web browser, etc.) is called as a *message*. A message is actually broken down into *segments* by the transport layer, which is not shown here. Note that the transport layer of TCP/IP contains two protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). TCP is more often used, and therefore, we shall concentrate on it. For the scope of our current text, the same discussion would also apply to UDP. The transport layer then adds its own header to the segment and gives it to the network layer. The network layer adds the IP header to this block, and gives the result to the data link layer. The data link layer adds the frame header and gives it to the physical layer for transmission. At the physical layer, the actual bits are transmitted as voltage pulses. An opposite process happens at the destination end, where each layer removes the previous layer's header, and finally the application layer receives the original message.

9.1.2 TCP Segment Format

We must now examine what do we mean by saying that the transport (TCP), network (IP) and data link (frame) layers add headers to the received data block. For instance, when the transport layer adds the TCP header to the original message, it not only appends the header fields to the original message, but also performs some processing, such as calculating the checksum for error detection, etc. After the headers are added, a TCP segment looks as shown in Fig. 9.3. As we can see, a TCP segment consists of a header of size 20 to 60 bytes, followed by actual data. The header consists of 20 bytes if the TCP segment does not contain

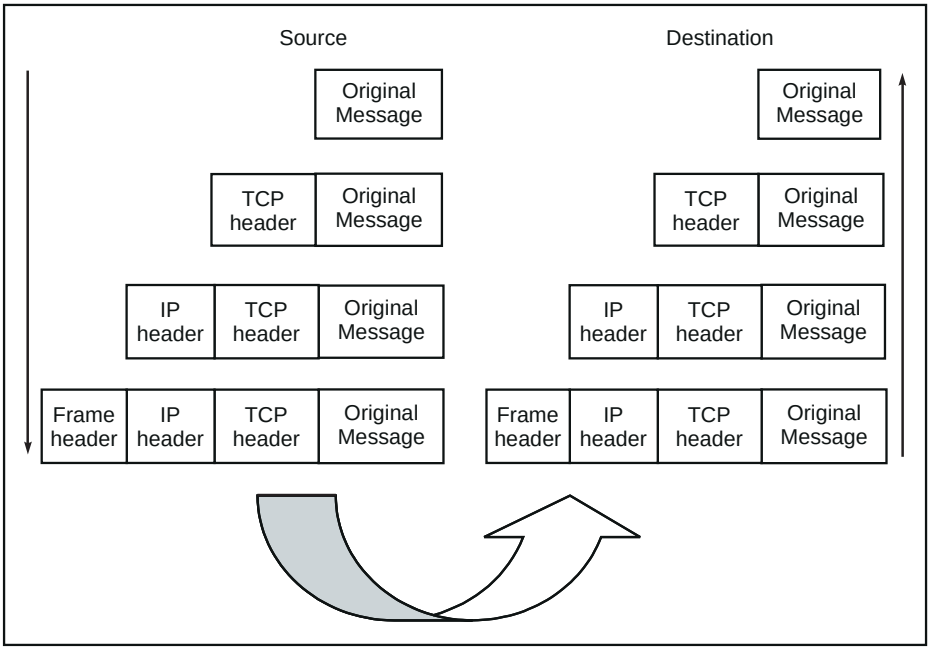

Fig. 9.2 *Message transfer from the source to the destination at different TCP/IP layers*

any options. Otherwise, the header consists of 60 bytes. That is, a maximum of 40 bytes are reserved for options. Options can be used to convey additional information to the destination. However, we shall ignore them, as they are not very frequently used.

Let us briefly discuss the header fields inside a TCP segment.

- **Source port number:** This 2-byte number signifies the port number of the source computer, corresponding to the application that is sending this TCP segment.
- **Destination port number:** This 2-byte number signifies the port number of the destination computer, corresponding to the application that is expected to receive this TCP segment.
- **Sequence number:** This 4-byte field defines the number assigned to the first byte of the data portion contained in this TCP segment. TCP is a connection-oriented protocol. For ensuring a correct delivery, each byte to be transmitted from the source to the destination is numbered in an increasing sequence. The sequence number field tells the destination host, which byte in this sequence comprises the first byte of the TCP segment. During the TCP connection establishment phase, both the source as well as the destination generate different unique random numbers. For instance, if this random number is 3130 and the first TCP packet is carrying 2000 bytes of data, then the sequence number field for that packet would contain 3132 (bytes 3130 and 3131 are used in connection establishment). The second segment would then have a sequence number of 5132 (3132 + 2000), and so on.
- **Acknowledgement number:** If the destination host receives a segment with sequence number X correctly, it sends X + 1 as the acknowledgement number back to the source.

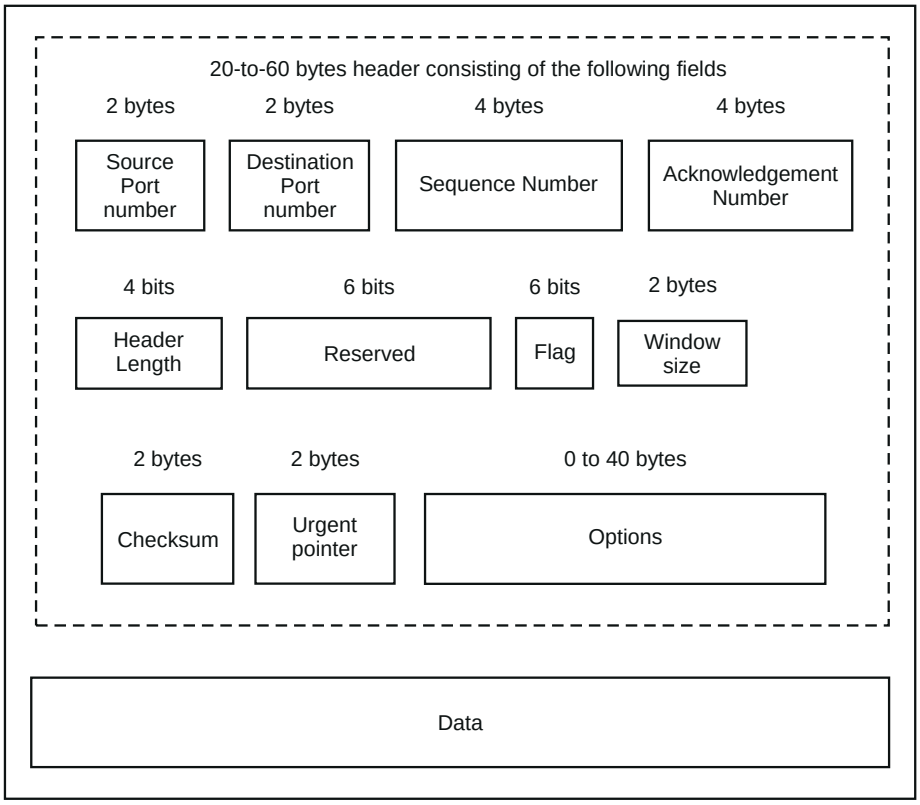

Fig. 9.3 *TCP segment format*

Thus, this 4-byte number defines the sequence number that the source is expecting from the destination as a receipt of the correct delivery.

- **Header length:** This 4-bit field specifies the number of four-byte words in the TCP header. As we know, the header length can be between 20 and 60 bytes. Therefore, the value of this field can be between 5 (because $5 \times 4 = 20$) and 15 (because $15 \times 4 = 60$).
- **Reserved:** This 6-byte field is reserved for future use and is currently unused.
- **Flag:** This 6-bit field defines six different control flags, each one of them occupying one bit. Out of the six flags, two are most important. The SYN flag indicates that the source wants to establish a connection with the destination. Therefore, this flag is used when a TCP connection is being established between two hosts. Similarly, the other flag of importance is the FIN flag. If the bit corresponding to this flag is set, then it means that the sender wants to terminate the current TCP connection.
- **Window size:** This field determines the size of the sliding window that the other party must maintain.
- **Checksum:** This 16-bit field contains the checksum for facilitating the error detection and correction.
- **Urgent pointer:** This field is used in situations where data in a TCP segment is more

important or urgent than other data in the same TCP connection. However, a discussion of such situations is beyond the scope of the current text.

9.1.3 IP Datagram Format

The TCP header plus the original message is now passed to the IP layer. The IP layer treats this whole package of TCP header + original message as its *original message*, and adds its own header to it. This results into the creation of an IP datagram. The format of an IP datagram is shown in Fig. 9.4.

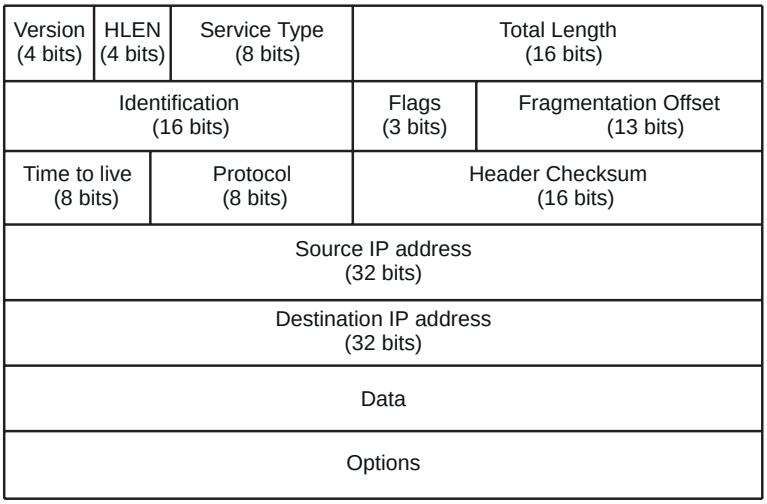

Fig. 9.4 *IP datagram*

An IP datagram is a variable-length datagram. A message can be broken down into multiple datagrams and a datagram in turn can be fragmented into different fragments, as we shall see. A datagram can contain a maximum of 65,536 bytes. A datagram is made up of two main parts: the header and the data. The header consists of anywhere between 20 and 60 bytes and essentially contains information about routing and delivery. The data portion contains the actual data to be sent to the recipient. The header is like an envelope: it contains information *about* the data. The data is analogous to the letter *inside* the envelope. Let us examine the fields of a datagram in brief.

- **Version:** This field currently contains a value 4, which indicates **IP version 4 (IPv4)**. In future, this field would contain 6 when **IP version 6 (IPv6)** becomes the standard.
- **Header Length (HLEN):** Indicates the size of the header in a multiple of four-byte words. When the header size is 20 bytes as shown in the figure, the value of this field is 5 (because $5 \times 4 = 20$), and when the option field is at the maximum size, the value of HLEN is 15 (because $15 \times 4 = 60$).
- **Service type:** This field is used to define service parameters such as the priority of the datagram and the level of reliability desired.
- **Total length:** This field contains the total length of the IP datagram. Because it is two bytes long, an IP datagram cannot be more than 65,536 bytes ($2^{16} = 65{,}536$).

- **Identification:** This field is used in the situations when a datagram is fragmented. As a datagram passes through different networks, it might be fragmented into smaller sub-datagrams to match the physical datagram size of the underlying network. In these situations, the sub-datagrams are sequenced using the identification field, so that the original datagram can be reconstructed from them.
- **Flags:** This field corresponds to the earlier field (identification). It indicates whether a datagram can be fragmented in the first place—and if it can be fragmented, whether it is the first or the last fragment, or it can be a middle fragment, etc.
- **Fragmentation offset:** If a datagram is fragmented, this field is useful. It is a pointer that indicates the offset of the data in the original datagram before fragmentation. This is useful when reconstructing a datagram from its fragments.
- **Time to live:** We know that a datagram travels through one or more routers before reaching its final destination. In the case of network problems, some of the routes to the final destination may not be available because of many reasons such as hardware failure, link failure, or congestion. In that case, the datagram may be sent through a different route. This can continue for a long time if the network problems are not resolved quickly. Soon, there could be many datagrams traveling in different directions through lengthy paths, trying to reach their destinations. This can create congestion and the routers may become too busy, thus bringing at least parts of the Internet to a virtual halt. In some cases, the datagrams can continue to travel in a loop in between, without reaching the final destination and in fact, coming back to the original sender. To avoid this, the datagram sender initializes this field (that is, *Time to live*) to some number. As the datagram travels through routers, this field is decremented each time. If the value in this field becomes zero or negative, it is immediately discarded. No attempt is made to forward it to the next hop. This avoids a datagram traveling for an infinite amount of time through various routers, and therefore, helps avoid network congestion. After all the other datagrams have reached the destination, the TCP protocol operating at the destination will find out this missing datagram and will have to request for its retransmission. Thus, IP is not responsible for the error-free, timely and in-sequence delivery of the entire message—it is done by TCP.
- **Protocol:** This field identifies the transport protocol running on top of IP. After the datagram is constructed from its fragments, it has to be passed on to the upper layer software piece. This could be TCP or UDP. This field specifies which piece of software at the destination node the datagram should be passed on to.
- **Source address:** This field contains the 32-bit IP address of the sender.
- **Destination address:** This field contains the 32-bit IP address of the final destination.
- **Options:** This field contains optional information such as routing details, timing, management, and alignment. For instance, it can store the information about the exact route that the datagram has taken. When it passes through a router, the router puts in its id, and optionally, also the time when it passed through that router, in one of the slots in this field. This helps tracing and fault detection of datagrams. However, most of the time, the space in this field is not sufficient for all these details, and therefore, it is not used very often.

This brief introduction to TCP/IP would suffice for the scope of the current text.

9.2 FIREWALLS

9.2.1 Introduction

The dramatic rise and progress of the Internet has opened possibilities that no one would have thought of. We can connect any computer in the world to any other computer, no matter how far the two are located from each other. This is undoubtedly a great advantage for individuals and corporate as well. However, this can be a nightmare for network support staff, which is left with a very difficult job of trying to protect the corporate networks from a variety of attacks. At a broad level, there are two kinds of attacks:

- Most corporations have large amounts of valuable and confidential data in their networks. Leaking of this critical information to competitors can be a great setback.
- Apart from the danger of the insider information leaking out, there is a great danger of the outside elements (such as viruses and worms) entering a corporate network to create havoc.

We can depict this situation as shown in Fig. 9.5.

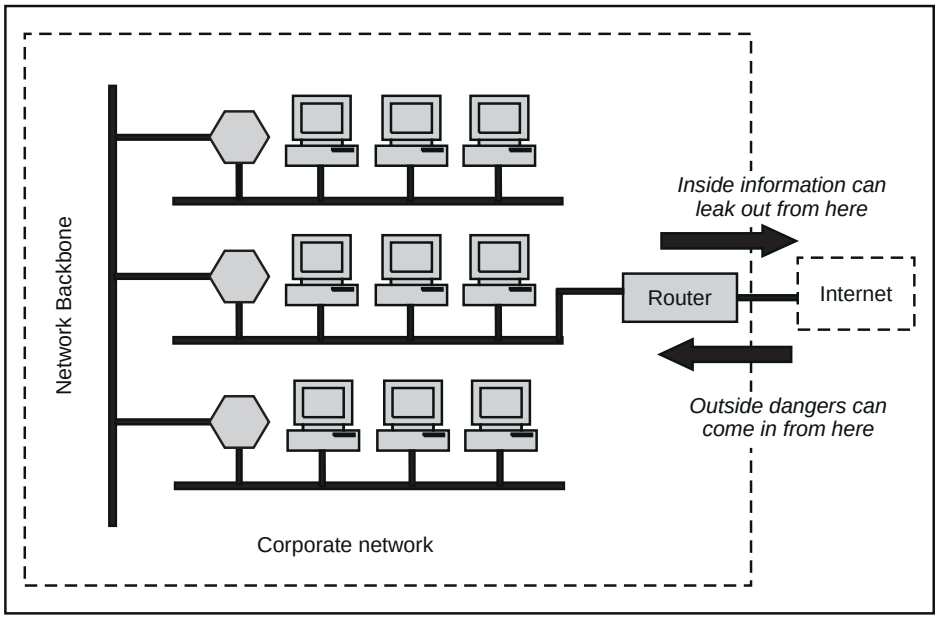

Fig. 9.5 *Threats from inside and outside a corporate network*

As a result of these dangers, we must have mechanisms which can ensure that the inside information remains inside, and also prevent the outsider attackers from entering inside a corporate network. As we know, encryption of information (if implemented properly) renders its transmission to the outside world redundant. That is, even if confidential information flows out of a corporate network, if it is in encrypted form, outsiders cannot make any sense of it. However, encryption does not work in the other direction. Outside attackers

can still try to break inside a corporate network. Consequently, better schemes are desired to achieve protection from outside attacks. This is where a **firewall** comes into picture.

Conceptually, a firewall can be compared with a sentry standing outside an important person's house (such as the nation's president). This sentry usually keeps an eye on and physically checks every person that enters into or comes out of the house. If the sentry senses that a person wishing to enter the president's house is carrying a knife, the sentry would not allow the person to enter. Similarly, even if the person does not possess any banned objects, but somehow looks suspicious, the sentry can still prevent that person's entry.

A firewall acts like a sentry. If implemented, it guards a corporate network by standing between the network and the outside world. All traffic between the network and the Internet in either direction must pass through the firewall. The firewall decides if the traffic can be allowed to flow, or whether it must be stopped from proceeding further. This is shown in Fig. 9.6.

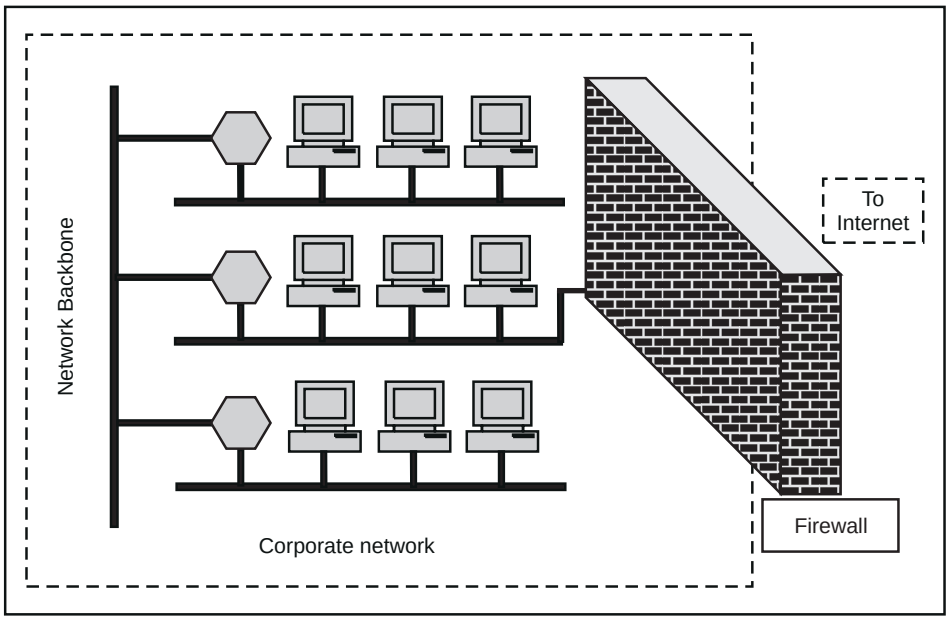

Fig. 9.6 *Firewall*

Of course, technically, a firewall is a specialized version of a router. Apart from the basic routing functions and rules, a router can be configured to perform the firewall functionality, with the help of additional software resources.

The characteristics of a good firewall implementation can be described as follows.

- All traffic from inside to outside, and vice versa, must pass through the firewall. To achieve this, all the access to the local network must first be physically blocked, and access only via the firewall should be permitted.
- Only the traffic authorized as per the local security policy should be allowed to pass through.
- The firewall itself must be strong enough, so as to render attacks on it useless.

9.2.2 Types of Firewalls

Based on the criteria that they use for filtering traffic, firewalls are generally classified into two types, as shown in Fig. 9.7.

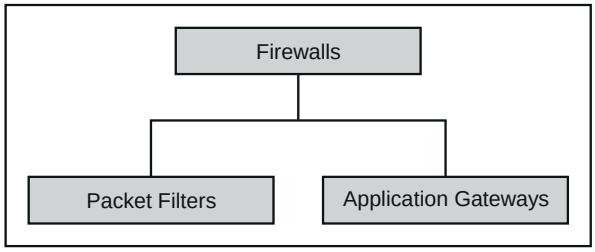

Fig. 9.7 *Types of firewalls*

Let us discuss these two types of firewalls one-by-one.

1. *Packet filters*

As the name suggests, a **packet filter** applies a set of rules to each packet, and based on the outcome, decides to either forward or discard the packet. It is also called as **screening router** or **screening filter**. Such a firewall implementation involves a router, which is configured to filter packets going in either direction (from the local network to the outside world, and vice versa). The filtering rules are based on a number of fields in the IP and TCP/UDP headers, such as source and destination IP addresses, IP protocol field (which identifies if the protocol in the upper transport layer is TCP or UDP), TCP/UDP port numbers (which identify the application which is using this packet, such as email, file transfer or World Wide Web).

The idea of a packet filter is shown in Fig. 9.8.

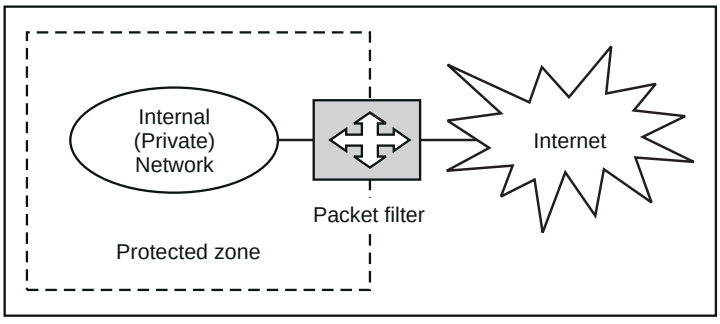

Fig. 9.8 *Packet filter*

Conceptually, a packet filter can be considered as a router that performs three main actions, as shown in Fig. 9.9.

A packet filter performs the following functions.

1. Receive each packet as it arrives.
2. Pass the packet through a set of rules, based on the contents of the IP and transport header fields of the packet. If there is a match with one of the set rules, decide whether to accept

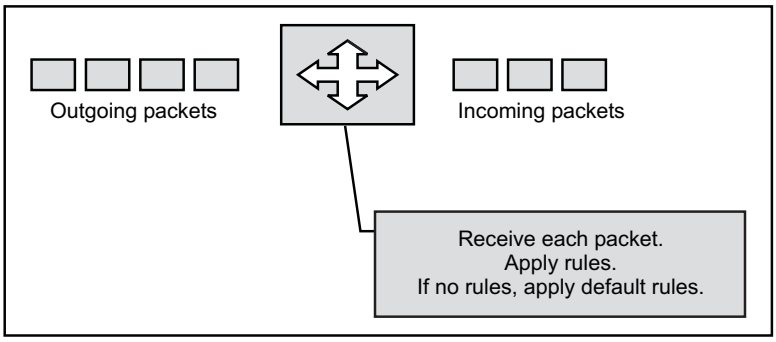

Fig. 9.9 *Packet filter operation*

or discard the packet based on that rule. For example, a rule could specify: disallow all incoming traffic from an IP address 157.29.19.10 (this IP address is taken just as an example), or disallow all traffic that uses UDP as the higher (transport) layer protocol.

3. If there is no match with any rule, take the default action. The default can be *discard all packets*, or *accept all packets*. The former policy is more conservative, whereas the latter is more open. Usually, the implementation of a firewall begins with the default *discard all packets* option, and then rules are applied one-by-one to enforce packet filtering.

The chief advantage of the packet filter is its simplicity. The users need not be aware of a packet filter at all. Packet filters are very fast in their operating speed. However, the two disadvantages of a packet filter are the difficulties in setting up the packet filter rules correctly, and lack of support for authentication.

Attackers can try and break the security of a packet filter by using the following techniques.

1. **IP address spoofing:** An intruder outside the corporate network can attempt to send a packet towards the internal corporate network, with the source IP address set equal to one of the IP addresses of the internal users. This is shown in Fig. 9.10. This attack can be defeated by discarding all the packets that arrive at the incoming side of the firewall, with the source address equal to one of the internal addresses.
2. **Source routing attacks:** An attacker can specify the route that a packet should take as it moves along the Internet. The attacker hopes that by specifying this option, the packet filter can be fooled to bypass its normal checks. Discarding all packets that use this option can thwart such an attack.
3. **Tiny fragment attacks:** IP packets pass through a variety of physical networks, such as Ethernet, Token Ring, X.25, Frame Relay, ATM, etc. All these networks have a predefined maximum frame size (called as the Maximum Transmission Unit or MTU). Many times, the size of the IP packet is greater than this maximum size allowed by the underlying network. In such cases, the IP packet needs to be fragmented, so that it can be accommodated inside the physical frame, and carried further. An attacker might attempt to use this characteristic of the TCP/IP protocol suite by intentionally creating fragments of the original IP packet and sending them. The attacker feels that the packet filter can be fooled, so that after fragmentation, it checks only the first fragment, and does not check the remaining fragments. This attack can be foiled by discarding all the packets where the (upper layer) protocol type is TCP and the packet

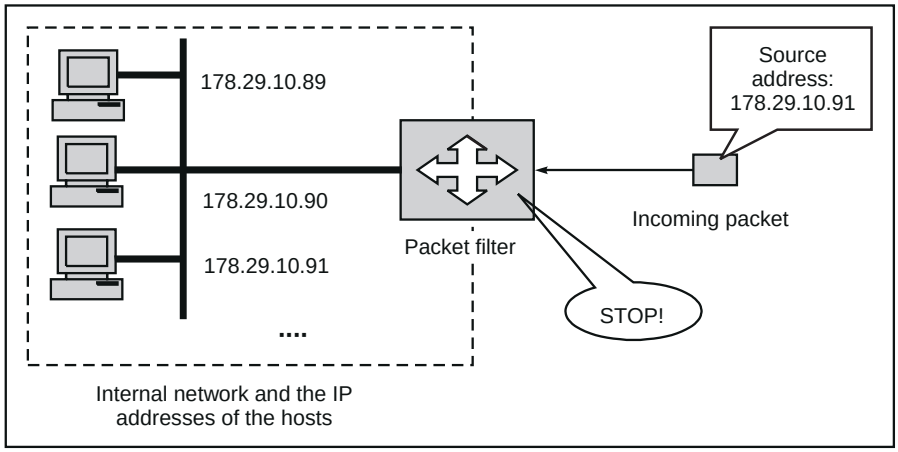

Fig. 9.10 *Packet filter defeating the IP address spoofing attack*

is fragmented (refer to *identification* and *protocol* fields of an IP packet discussed earlier to understand how we can implement this).

An advanced type of packet filter is called as **dynamic packet filter** or **stateful packet filter**. A dynamic packet filter allows the examination of packets based on the current state of the network. That is, it adapts itself to the current exchange of information, unlike the normal packet filters, which have routing rules hard coded. For instance, we can specify a rule with the help of a dynamic packet filter as follows:

Note Allow incoming TCP packets only if they are responses to the outgoing TCP packets that have gone through our network.

Note that the dynamic packet filter has to maintain a list of the currently open connections and outgoing packets in order to deal with this rule. Hence, it is called as *dynamic* or *stateful*. When such a rule is in effect, the logical view of the packet filtering can be illustrated as shown in Fig. 9.11.

As shown in the figure, firstly, an internal client sends a TCP packet to an external server, which the dynamic packet filter allows. In response, the server sends back a TCP packet, which the packet filter examines, and realizes that it is a response to the internal client's request. Therefore, it allows that packet in. However, next, the external server sends a new UDP packet, which the filter does not allow, because previously, the exchange of the client and the server packets happened using the TCP protocol. However, this packet is based on the UDP protocol. Since this is against the rule that was set up earlier, the filter drops the packet.

2. *Application gateways*

An **application gateway** is also called as a **proxy server**. This is because it acts like a proxy (i.e. deputy or substitute), and decides about the flow of application level traffic. The idea is shown in Fig. 9.12.

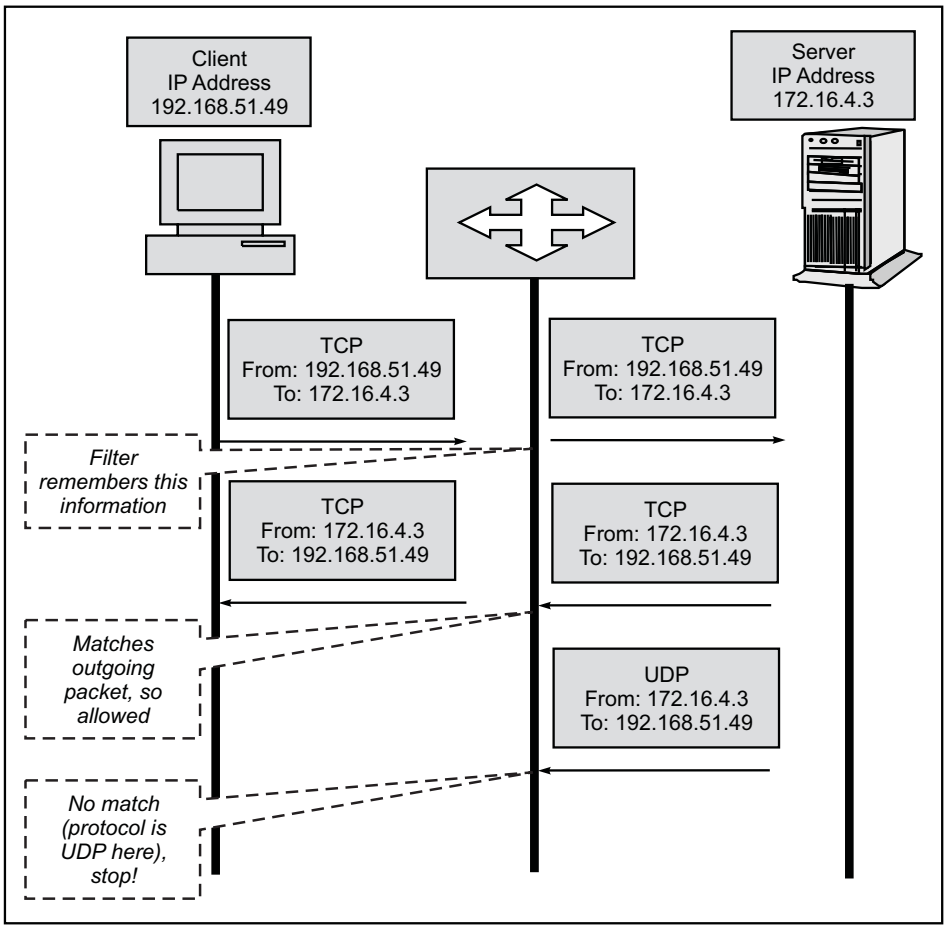

Fig. 9.11 *Dynamic packet filter technology*

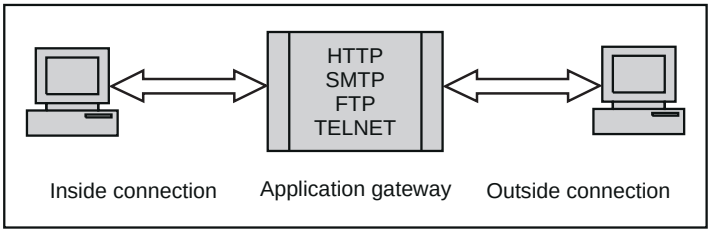

Fig. 9.12 *Application gateway*

Application gateways typically work as follows.

1. An internal user contacts the application gateway using a TCP/IP application, such as HTTP or TELNET.
2. The application gateway asks the user about the remote host with which the user wants to set up a connection for actual communication (i.e. its domain name or IP address, etc.).

The application gateway also asks for the user id and the password required to access the services of the application gateway.

3. The user provides this information to the application gateway.
4. The application gateway now accesses the remote host on behalf of the user, and passes the packets of the user to the remote host. Note that there is a variation of the application gateway, called as **circuit gateway**, which performs some additional functions as compared to those performed by an application gateway. A circuit gateway, in fact, creates a new connection between itself and the remote host. The user is not aware of this, and thinks that there is a direct connection between itself and the remote host. Also, the circuit gateway changes the source IP address in the packets from the end user's IP address to its own. This way, the IP addresses of the computers of the internal users are hidden from the outside world. This is shown in Fig. 9.13. Of course, both the connections are shown with a single arrow to stress on the concept, in reality, both the connections are two-ways.

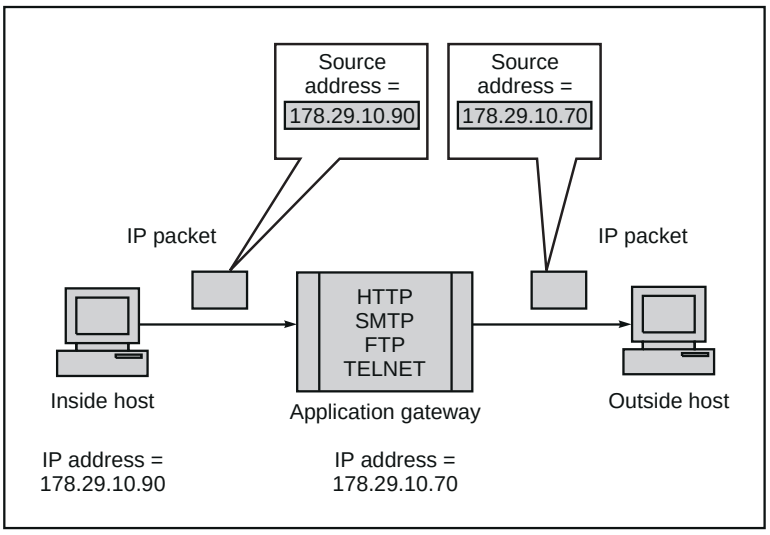

Fig. 9.13 *Circuit gateway operation*

The SOCKS server is an example of the real-life implementation of a circuit gateway. It is a client-server application. The SOCKS client runs on the internal hosts, and the SOCKS server runs on the firewall.

5. From here onwards, the application gateway acts like a proxy of the actual end user, and delivers packets from the user to the remote host and vice versa.

Application gateways are generally more secure than packet filters, because rather than examining every packet against a number of rules, here we simply detect whether a user is allowed to work with a TCP/IP application, or not. The disadvantage of application gateways is the overhead in terms of connections. As we noticed, there are actually two sets of connections now: one between the end user and the application gateway, and another between the application gateway and the remote host. The application gateway has to

manage these two sets of connections, and the traffic going between them. This means that the actual communicating internal host is under an illusion, as illustrated in Fig. 9.14.

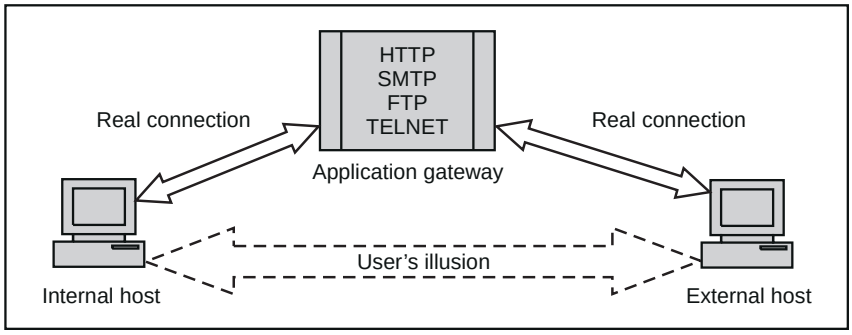

Fig. 9.14 *Application gateway creates an illusion*

An application gateway is also called as **bastion host**. Usually, a bastion host is a very key point in the security of a network.

9.2.3 Firewall Configurations

In practical implementations, a firewall is usually a combination of packet filters and application (or circuit) gateways. Based on this, there are three possible configurations of firewalls, as shown in Fig. 9.15.

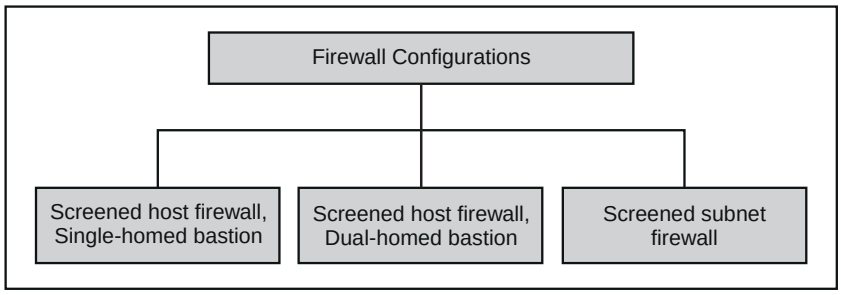

Fig. 9.15 *Firewall configurations*

Let us discuss these possible configurations now.

1. *Screened host firewall, Single-homed bastion*

In the **Screened host firewall, Single-homed bastion** configuration, a firewall set up consists of two parts: a packet-filtering router and an application gateway. Their purposes are as follows.

- The packet filter ensures that the incoming traffic (i.e. from the Internet to the corporate network) is allowed only if it is destined for the application gateway, by examining the *destination address* field of every incoming IP packet. Similarly, it also ensures that the outgoing traffic (i.e. from the corporate network to the Internet) is

allowed only if it is originating from the application gateway, by examining the *source address* field of every outgoing IP packet.

- The application gateway performs authentication and proxy functions, as explained earlier.

This configuration is illustrated in Fig. 9.16.

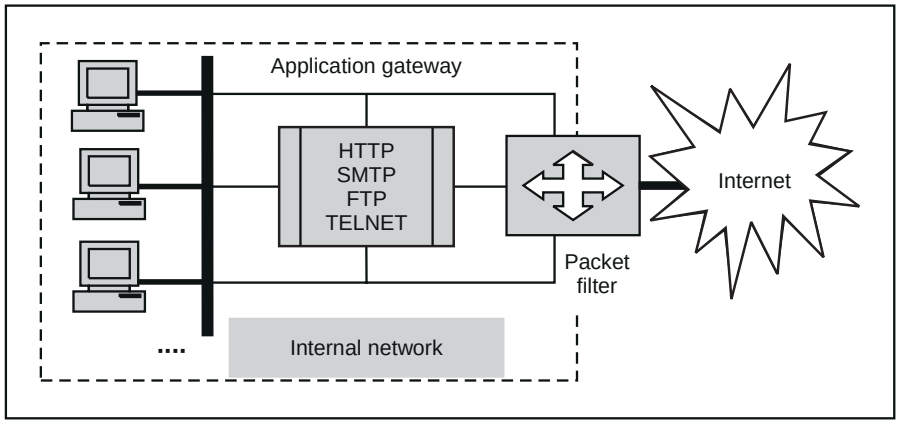

Fig. 9.16 *Screened host firewall, Single-homed bastion*

This configuration increases the security of the network by performing checks at both packet and application levels. This also gives more flexibility to the network administrators to define more granular security policies.

However, as we can see, one big disadvantage here is that the internal users are connected to the application gateway, as well as to the packet filter. Therefore, if the packet filter is somehow successfully attacked and its security compromised, then the whole internal network is exposed to the attacker.

2. *Screened host firewall, Dual-homed bastion*

To overcome the drawback of a *screened host firewall, single-homed bastion* configuration, another type of configuration, called as **Screened host firewall, Dual-homed bastion**, exists. This configuration is an improvement over the earlier scheme. Here, direct connections between the internal hosts and the packet filter are avoided. Instead, the packet filter connects only to the application gateway, which, in turn, has a separate connection with the internal hosts. Therefore, now even if the packet filter is successfully attacked, only the application gateway is visible to the attacker. The internal hosts are protected. This is shown in Fig. 9.17.

Can we think of a scheme, which is even better than this?

3. *Screened subnet firewall*

The **Screened subnet firewall** offers the highest security among the possible firewall configurations. It is an improvement over the previous scheme of *screened host firewall, Dual-homed bastion*. Here, two packet filters are used, one between the Internet and the application gateway, as previously, and another one between the application gateway and the internal network. This is shown in Fig. 9.18.

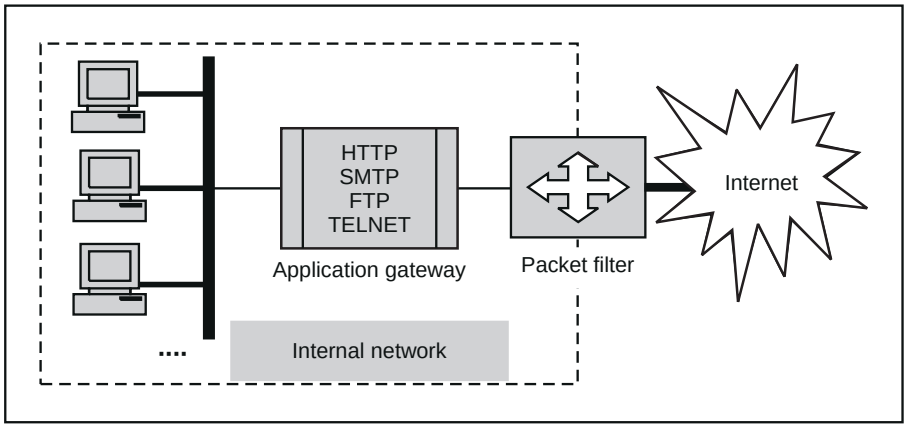

Fig. 9.17 *Screened host firewall, Dual-homed bastion*

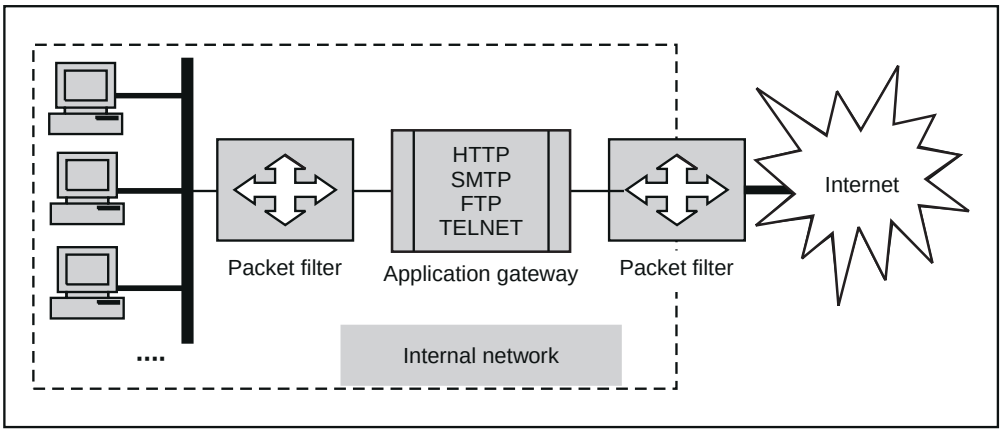

Fig. 9.18 *Screened subnet firewall*

Now, there are three levels of security for an attacker to break into. This makes the life of the attacker very difficult. The attacker does not come to know about the internal network, unless she breaks into both the packet filters and the single application gateway standing between them.

9.2.4 Demilitarized Zone (DMZ) Networks

The concept of a **Demilitarized Zone (DMZ)** networks is quite popular in firewall architectures. Firewalls can be arranged to form a DMZ. DMZ is required only if an organization has servers that it needs to make available to the outside world (e.g. Web servers or FTP servers). For this, a firewall has at least three network interfaces. One interface connects to the internal private network; the second connects to the external public network (i.e. the Internet), and the third connects to the public servers (which form the DMZ network). The idea is illustrated in Fig. 9.19.

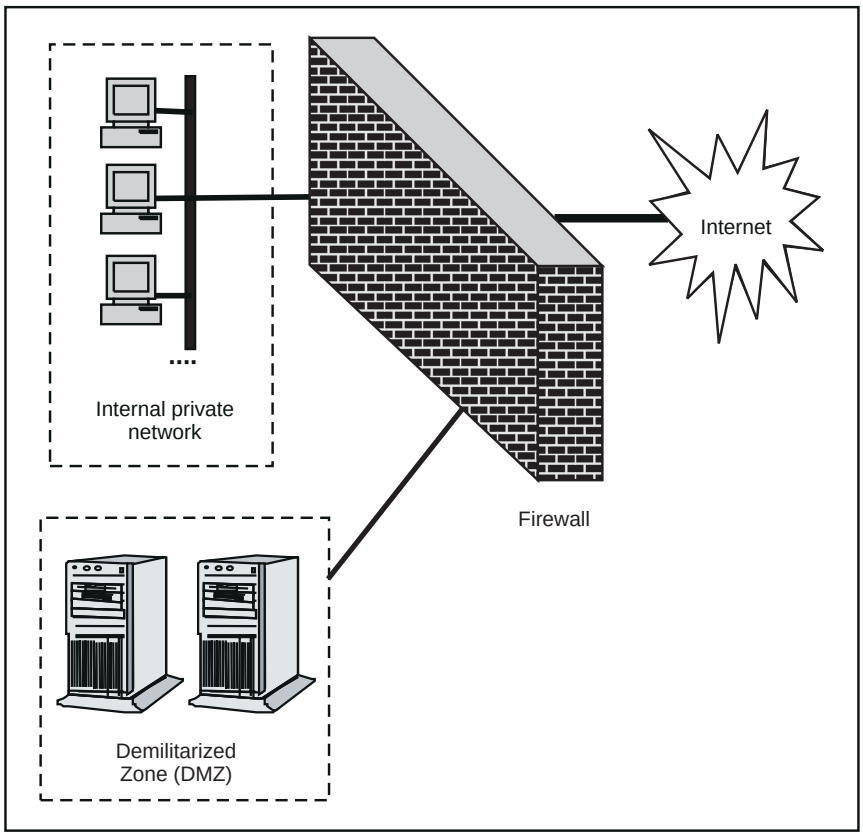

Fig. 9.19 *Demilitarized (DMZ) network*

The chief advantage of such a scheme is that the access to any service on the DMZ can be restricted. For instance, if the Web server is the only required service, we can limit the traffic in/out of the DMZ network to the HTTP and HTTPS protocols (i.e. ports 80 and 443, respectively). All other traffic can be filtered. More significantly, the internal private network is no way directly connected to the DMZ. So, even if an attacker can somehow manage to hack into the DMZ, the internal private network is safe, and out of the reach of the attacker.

9.2.5 Limitations of Firewall

We must note that although a firewall is an extremely useful security measure for an organization, it does not solve all the practical security problems. The main limitations of a firewall can be listed as follows.

1. **Insider's intrusion:** As we know, a firewall system is designed to thwart outside attacks. Therefore, if an inside user attacks the internal network in some way, the firewall cannot prevent such an attack.
2. **Direct Internet traffic:** A firewall must be configured very carefully. It is effective only if it is the only entry-exit point of an organization's network. If, instead, the

firewall is *one of the* entry-exit points, a user can bypass the firewall and exchange information with the Internet via the other entry-exit points. This can open up the possibilities of attacks on the internal network through those points. The firewall cannot, obviously, be expected to take care of such situations.

3. **Virus attacks:** A firewall cannot protect the internal network from virus threats. This is because a firewall cannot be expected to scan every incoming file or packet for possible virus contents. Therefore, a separate virus detection and removal mechanism is required for preventing virus attacks. Alternatively, some vendors bundle their firewall products with anti-virus software, to enable both the features *out of the box*.

9.3 IP SECURITY

9.3.1 Introduction

The IP packets contain data in plain text form. That is, anyone watching the IP packets pass by can actually access them, read their contents, and even change them. We have studied higher-level security mechanisms (such as SSL, SHTTP, PGP, PEM, S/MIME and SET) to prevent such kinds of attacks. Although these higher-level protocols enhance the protection mechanisms, there was a general feeling for a long time that why not secure IP packets themselves? If we can achieve this, then we need not rely only on the higher-level security mechanisms. The higher-level security mechanisms can then serve as additional security measures. Thus, we will have two levels of security in this scheme:

- First offer security at the IP packet level itself.
- Continue implementing higher-level security mechanisms, depending on the requirements.

This is shown in Fig. 9.20.

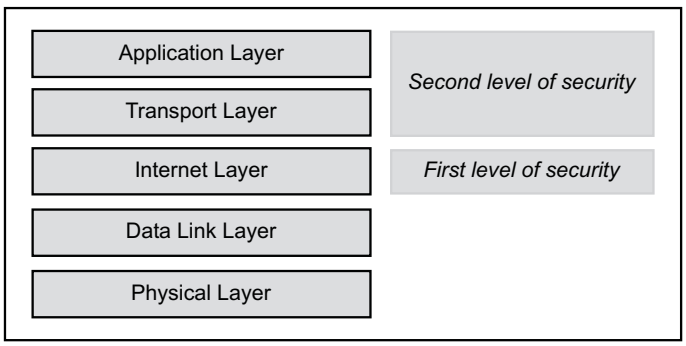

Fig. 9.20 *Security at the Internet layer as well as the above layers*

We have already discussed the higher-level security protocols. Our focus of discussion in this chapter is the first level of security (at the Internet layer).

In 1994, the Internet Architecture Board (IAB) prepared a report, called as *Security in the Internet Architecture* (RFC 1636). This report stated that the Internet was a very open network, which was unprotected from hostile attacks. Therefore, said the report, the Internet

needs better security measures, in terms of authentication, integrity and confidentiality. Just in 1997, about 150,000 Websites were attacked in various ways, proving that the Internet was quite an unsafe place at times. Consequently, the IAB decided that authentication, integrity and encryption must be a part of the next version of the IP protocol, called as *IP version 6 (Ipv6)* or *IP new generation (IPng)*. However, since the new version of IP was to take some years to be released and implemented, the designers devised ways to incorporate these security measures in the current version of IP, called as *IP version 4 (IPv4)* as well.

The outcome of the study and IAB's report is the protocol for providing security at the IP level, called as **IP Security (IPSec)**. In 1995, the Internet Engineering Task Force (IETF) published five security-based standards related to IPSec, as shown in Table 9.1.

Table 9.1 *RFC documents related to IPSec*

RFC Number	*Description*
1825	An overview of the security architecture
1826	Description of a packet authentication extension to IP
1827	Description of a packet encryption extension to IP
1828	A specific authentication mechanism
1829	A specific encryption mechanism

IPv4 *may* support these features, but IPv6 *must* support them. The overall idea of IPSec is to encrypt and seal the transport and application layer data during transmission. It also offers integrity protection for the Internet layer. However, the Internet header itself is not encrypted, because of which the intermediate routers can deliver encrypted IPSec messages to the intended recipient. The logical format of a message after IPSec processing is shown in Fig. 9.21.

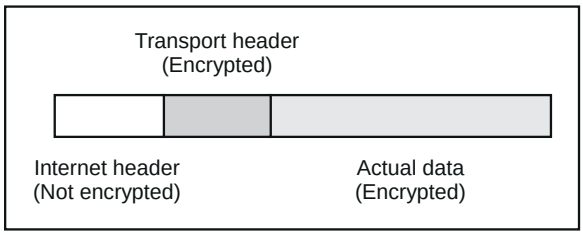

Fig. 9.21 *Result of IPSec processing*

Thus, the sender and the receiver look at IPSec as shown in Fig. 9.22 as another layer in the TCP/IP protocol stack. This layer sits in-between the transport and the Internet layers of the conventional TCP/IP protocol stack.

9.3.2 IPSec Overview

1. *Introduction*

We must learn a few terms and concepts in order to understand the IPSec protocol. All these concepts are inter-related. However, rather than looking at these individual concepts straightaway, we shall start with the big picture. We will first take a look at the basic concepts

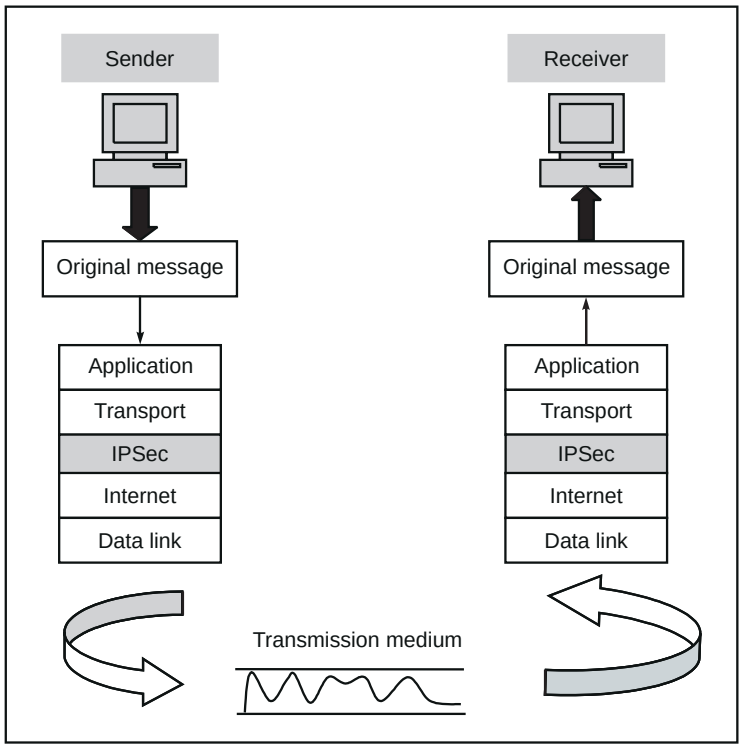

Fig. 9.22 *Conceptual IPSec positioning in the TCP/IP protocol stack*

in IPSec, and then elaborate each of the concepts. In this section, we shall restrict ourselves to the broad overview of the basic concepts in IPSec.

2. *IPSec protocols*

As we know, an IP packet consists of two portions: IP header and the actual data. IPSec features are implemented in the form of additional IP headers (called as **extension headers**) to the standard, default IP headers. These extension IP headers follow the standard IP headers. IPSec offers two main services: authentication and confidentiality. Each of these requires its own extension header. Therefore, to support these two main services, IPSec defines two IP extension headers: one for authentication, and another for confidentiality.

IPSec actually consists of two main protocols, as shown in Fig. 9.23.

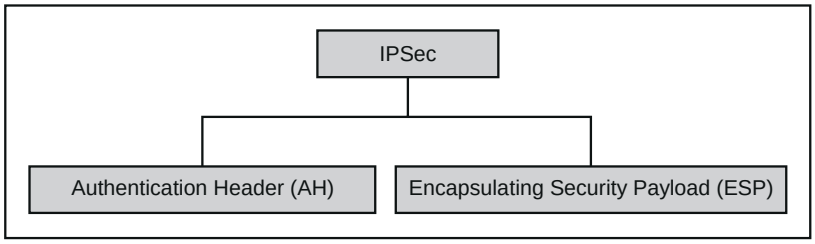

Fig. 9.23 *IPSec protocols*

These two protocols are required for the following purposes.

- The **Authentication Header (AH)** protocol provides authentication, integrity and an optional anti-replay service. The IPSec AH is a header in an IP packet, which contains a cryptographic checksum (similar to a message digest or hash) for the contents of the packet. The AH is simply inserted between the IP header and any subsequent packet contents. No changes are required to the data contents of the packet. Thus, security resides completely in the contents of the AH.
- The **Encapsulating Security Payload (ESP)** protocol provides data confidentiality. The ESP protocol also defines a new header to be inserted into the IP packet. ESP processing also includes the transformation of the protected data into an unreadable, encrypted forma. Under normal circumstances, the ESP will be *inside* the AH. That is, encryption happens first, and then authentication.

On receipt of an IP packet that was processed by IPSec, the receiver processes the AH first, if present. The outcome of this tells the receiver if the contents of the packet are all right, or whether they have been tampered with, while in transit. If the receiver finds the contents acceptable, it extracts the key and algorithms associated with the ESP, and decrypt the contents.

There are some more details that we should know. Both AH and ESP can be used in one of the two *modes*, as shown in Fig. 9.24.

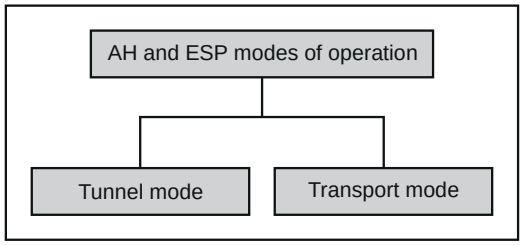

Fig. 9.24 *AH and ESP modes of operation*

We shall later study more about these modes. However, a quick overview would help.

In the **tunnel mode**, an encrypted *tunnel* is established between two hosts. Suppose X and Y are two hosts, wanting to communicate with each other using the IPSec tunnel mode. What happens here is that they identify their respective proxies, say P1 and P2, and a *logical encrypted tunnel* is established between P1 and P2. X sends its transmission to P1. The tunnel carries the transmission to P2. P2 forwards it to Y. This is shown in Fig. 9.25.

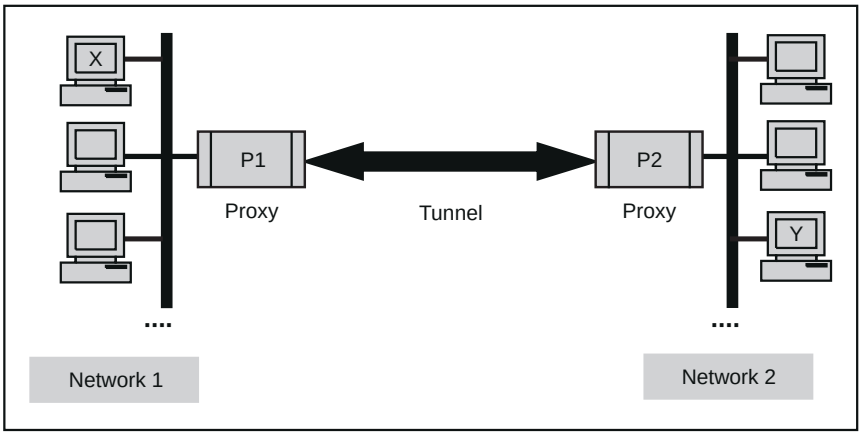

Fig. 9.25 *Concept of tunnel mode*

How do we implement this technically? As we shall see, we will have two sets of IP headers: internal and external. The internal IP header (which is encrypted) contains the source and destination addresses as X and Y, whereas the external IP header contains the source and destination addresses as P1 and P2. That way, X and Y are protected from potential attackers. This is shown in Fig. 9.26.

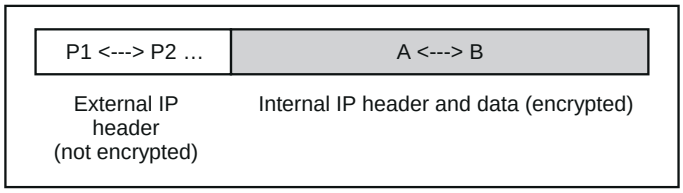

Fig. 9.26 *Implementation of tunnel mode*

In contrast, the transport mode does not hide the actual source and destination addresses. They are visible in plain text, while in transit.

3. *The Internet Key Exchange (IKE) Protocol*

Another supporting protocol is also used in IPSec. This protocol is used for the key management procedures, and is called as **Internet Key Exchange (IKE)** protocol.

Note IKE is used to negotiate the cryptographic algorithms to be later used by AH and ESP in the actual cryptographic operations.

The IPSec protocols are designed to be independent of the actual lower-level cryptographic algorithms. Thus, IKE is the initial phase of IPSec, where the algorithms and keys are decided. After the IKE phase, the AH and ESP protocols take over. This process is shown in Fig. 9.27.

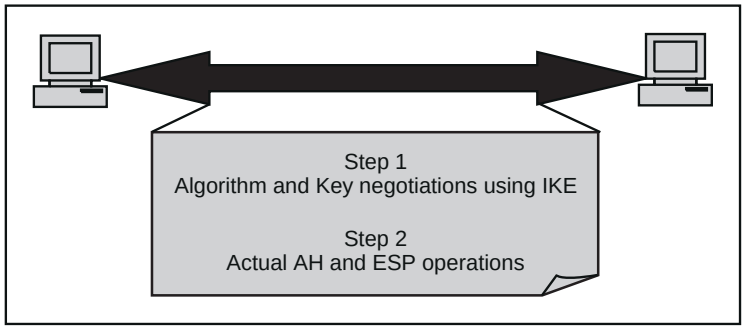

Fig. 9.27 *Steps in IPSec operation*

4. *Security Association (SA)*

The output of the IKE phase is a **Security Association (SA)**.

Note SA is an agreement between the communicating parties about factors such as the IPSec protocol version in use, mode of operation (transport mode or tunnel mode), cryptographic algorithms, cryptographic keys, lifetime of keys, etc.

By now, we would have guessed that the principal objective of the IKE protocol is to establish an SA between the communicating parties. Once this is done, both major protocols of IPSec (i.e. AH and ESP) make use of SA for their actual operation.

Note that if both AH and ESP are used, each communicating party requires two sets of SA: one for AH and one for ESP. Moreover, an SA is simplex, i.e. unidirectional. Therefore, at a second level, we need two sets of SA per communicating party: one for incoming transmission, and another for outgoing transmission. Thus, if the two communicating parties use both AH and ESP, each of them would require four sets of SA, as shown in Fig. 9.28.

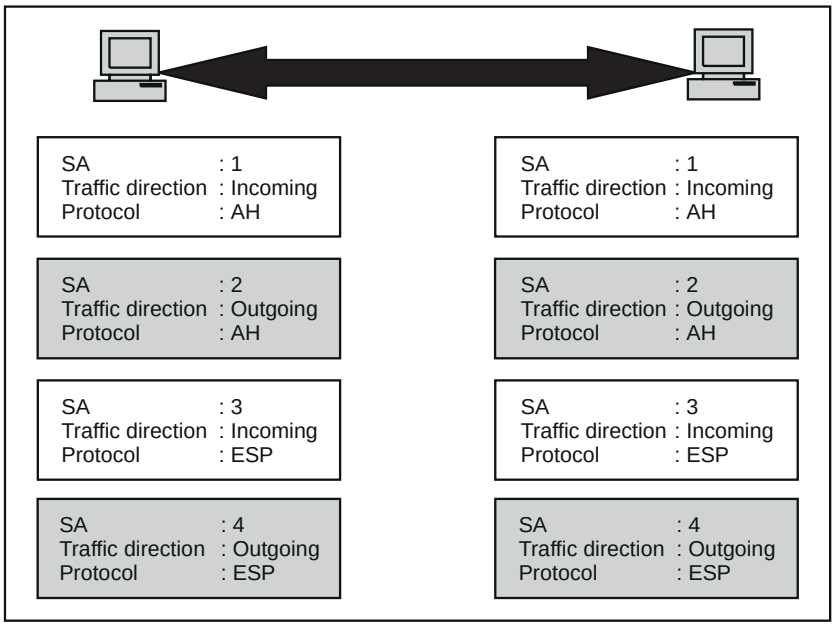

Fig. 9.28 *Security association types and classifications*

Obviously, both the communicating parties must allocate some storage area for storing the SA information at their end. For this purpose, a standard storage area called as **Security Association Database (SAD)** is predefined and used by IPSec. Thus, each communicating party requires maintaining its own SAD. The SAD contains active SA entries. The contents of a SAD are shown in Table 9.2.

Having discussed the background of IPSec, let us now discuss the two main protocols in IPSec: AH and ESP.

Table 9.2 *SAD fields*

Field	*Description*
Sequence number counter	This 32-bit field is used to generate the sequence number field, which is used in the AH or ESP headers.
Sequence counter overflow	This flag indicates whether the overflow of the sequence number counter should generate an audible event and prevent further transmission of packets on this SA.
Anti-replay window	A 32-bit counter field and a bit map, which are used to detect if an incoming AH or ESP packet is a replay.
AH authentication	AH authentication cryptographic algorithm and the required key.
ESP authentication	ESP authentication cryptographic algorithm and the required key.
ESP encryption	ESP encryption algorithm, key, Initial Vector (IV) and IV mode.
IPSec protocol mode	Indicates which IPSec protocol mode (e.g. transport or tunnel) should be applied to the AH and ESP traffic.
Path Maximum Transfer Unit (PMTU)	The maximum size of an IP datagram that will be allowed to pass through a given network path without fragmentation.
Lifetime	Specifies the life of the SA. After this time interval, the SA must be replaced with a new one.

9.3.3 Authentication Header (AH)

1. *AH format*

The Authentication Header (AH) provides support for data integrity and authentication of IP packets. The data integrity service ensures that data inside IP packets is not altered during the transit. The authentication service enables an end user or a computer system to authenticate the user or the application at the other end, and decide to accept or reject packets, accordingly. This also prevents the IP spoofing attacks. Internally, AH is based on the MAC protocol, which means that the two communicating parties must share a secret key in order to use AH. The AH structure is shown in Fig. 9.29.

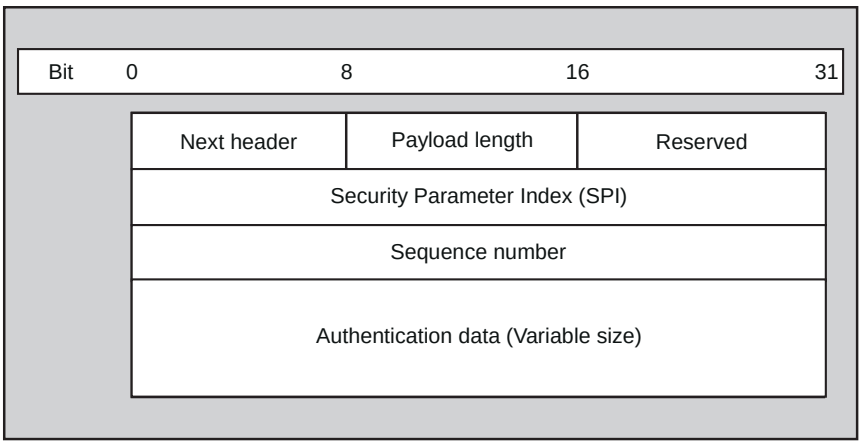

Fig. 9.29 *Authentication Header (AH) format*

Let us discuss the fields in the AH now, as shown in Table 9.3.

Table 9.3 *Authentication Header field descriptions*

Field	*Description*
Next header	This 8-bit field identifies the type of header that immediately follows the AH. For example, if an ESP header follows the AH, this field contains a value 50, whereas if another AH follows this AH, this field contains a value 51.
Payload length	This 8-bit field contains the length of the AH in 32-bit words minus 2. Suppose that the length of the authentication data field is 96 bits (or three 32-bit words). With a three-word fixed header, we have a total of 6 words in the header. Therefore, this field will contain a value of 4.
Reserved	This 16-bit field is reserved for future use.
Security Parameter Index (SPI)	This 32-bit field is used in combination with the source and destination addresses as well as the IPSec protocol used (AH or ESP) to uniquely identify the Security Association (SA) for the traffic to which a datagram belongs.
Sequence number	This 32-bit field is used to prevent replay attacks, as discussed later.
Authentication data	This variable-length field contains the authentication data, called as the Integrity Check Value (ICV), for the datagram. This value is the MAC, used for authentication and integrity purposes. For IPv4 datagrams, the value of this field must be an integral multiple of 32. For IPv6 datagrams, the value of this field must be an integral multiple of 64. For this, additional padding bits may be required. The ICV is calculated generating a MAC using the HMAC digest algorithm.

2. *Dealing with replay attacks*

Let us now study how AH deals with and prevents the replay attacks. To reiterate, in a replay attack, the attacker obtains a copy of an authenticated packet and later sends it to the intended destination. Since the same packet is received twice, the destination could face some problems because of this. To prevent this, as we know, the AH contains a field called as *sequence number*.

Initially, the value of this field is set to 0. Every time the sender sends a packet to the same sender over the same SA, it increments the value of this field by 1. The sender must not allow this value to circle back from $2^{32} - 1$ to 0. If the number of packets over the same increases this number, the sender must establish a new SA with the recipient.

On the receiver's side, there is some more processing involved. The receiver maintains a sliding window of size W, with the default value of W = 64. The right edge of the window represents the highest sequence number N received so far, for a valid packet. For simplicity, let us depict a sliding window with W = 8, as shown in Fig. 9.30.

Let us understand the significance of the receiver's sliding window, and also see how the receiver operates on it.

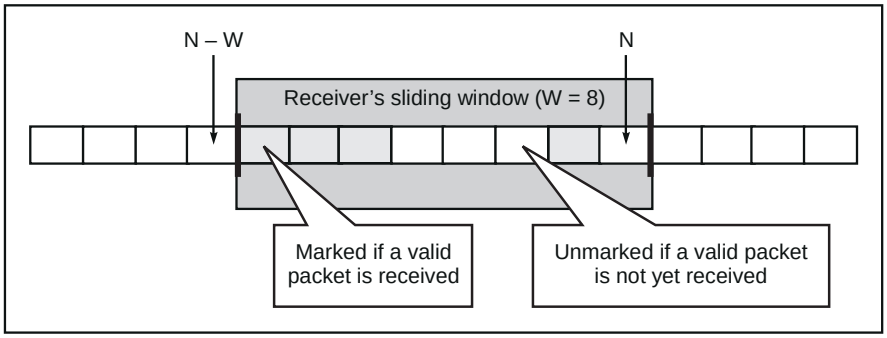

Fig. 9.30 *Receiver's sliding window*

As we can see, the following values are used:

- W: Specifies the size of the window. In our example, it is 8.
- N: Specifies the maximum highest sequence number so far received for a valid packet. N is always at the right edge of the window.

For any packet with a sequence number in the range from (N – W + 1) to N that has been correctly received (i.e. successfully authenticated), the corresponding slot in the window is marked (see figure). On the other hand, any packet in this range, which is not correctly received (i.e. not successfully authenticated), the slot is unmarked (see figure).

Now, when a receiver receives a packet, it performs the following action depending on the sequence number of the packet, as shown in Fig. 9.31.

1. If the sequence number of the received packet falls within the window, and if the packet is new, its MAC is checked. If the MAC is successfully validated, the corresponding slot in the window is marked. The window itself does not move to the right hand side.

2. If the received packet is to the right of the window [i.e. the sequence number of the packet is > N], and if the packet is new, the MAC is checked. If the packet is authenticated successfully, the window is advanced to the right in such a way that the right edge of the window now matches with the sequence number of this packet. That is, this sequence number now becomes the new N.

3. If the received packet is to the left of the window [i.e. the sequence number of the packet is < (N – W)], or if the MAC check fails, the packet is rejected, and an audible event is triggered.

Fig. 9.31 *Sliding window logic used by the receiver for each incoming packet*

Note that the third action thwarts replay attacks. This is because if the receiver receives a packet whose sequence number is less than (N – W), it concludes that someone posing as the sender is attempting to resend a packet sent by the sender earlier.

We must also realize that in extreme conditions, this kind of technique can make the receiver believe that a transmission is in error, even though it is not the case. For example, suppose that the value of W is 64 and that of N is 100. Now suppose that the sender sends

a burst of packets, numbered 101 to 500. Because of network congestions and other issues, suppose that the receiver somehow receives a packet with sequence number 300 first. It would immediately move the right edge of the window to 300 (i.e. N = 300 now). Now suppose that the receiver next receives packet number 102. From our calculations, N – W = 300 – 64 = 236. Therefore, the sequence number of the packet just received (102) is less than (N – W = 236). Thus, our third condition in the earlier list would get triggered, and the receiver would reject this valid packet, and raise an alarm.

However, such situations are rare, and with an optimized value of W, such situations can be avoided.

3. *Modes of operation*

As we know, both AH and ESP can work in two modes: the transport mode and the tunnel mode. Let us now discuss AH in the context of these two modes.

- **AH transport mode:** In the transport mode, the position of the Authentication Header (AH) is between the original IP header and the original TCP header of the IP packet. This is shown in Fig. 9.32.

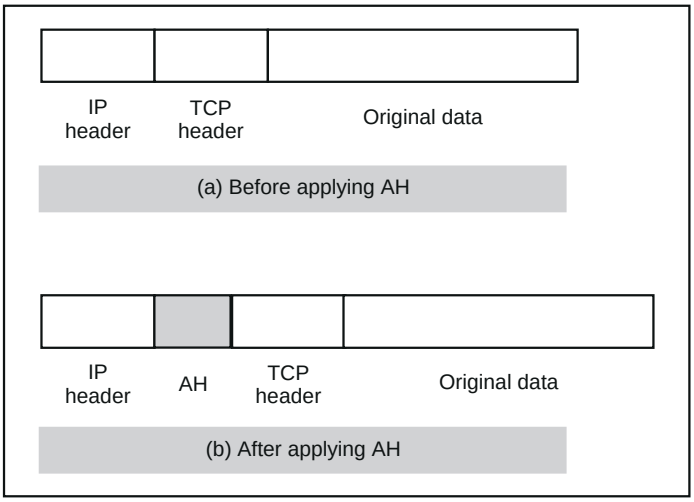

Fig. 9.32 *AH transport mode*

- **AH tunnel mode:** In the tunnel mode, the entire original IP packet is authenticated and the AH is inserted between the original IP header and a new outer IP header. The inner IP header contains the ultimate source and destination IP addresses, whereas the outer IP header possibly contains different IP addresses (e.g. IP addresses of the firewalls or other security gateways). This is shown in Fig. 9.33.

9.3.4 Encapsulating Security Payload (ESP)

1. *ESP format*

The Encapsulating Security Payload (ESP) protocol provides confidentiality and integrity of messages. ESP is based on symmetric key cryptography techniques. ESP can be used in isolation, or it can be combined with AH.

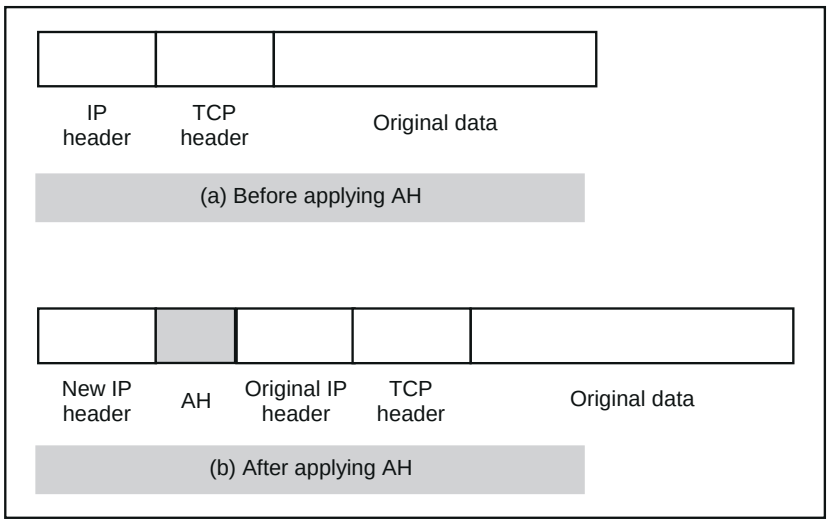

Fig. 9.33 *AH tunnel mode*

The ESP packet contains four fixed-length fields, and three variable-length fields. Figure 9.34 shows the ESP format.

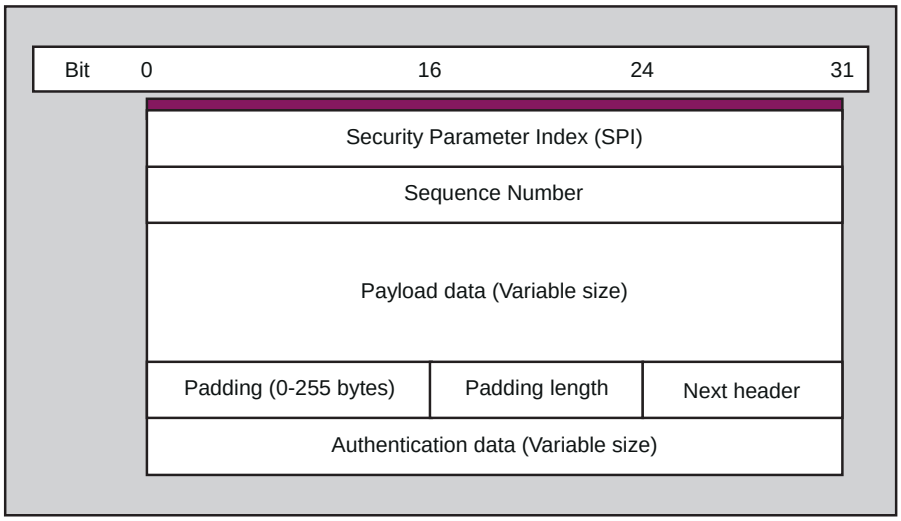

Fig. 9.34 *Encapsulating Security Payload (ESP) format*

Let us discuss the fields in the ESP now, as shown in Table 9.4.

2. *Modes of operation*

ESP, like AH, can operate in the transport mode or the tunnel mode. Let us discuss these two possibilities now.

Table 9.4 *ESP field descriptions*

Field	*Description*
Security Parameter Index (SPI)	This 32-bit field is used in combination with the source and destination addresses as well as the IPSec protocol used (AH or ESP) to uniquely identify the Security Association (SA) for the traffic to which a datagram belongs.
Sequence number	This 32-bit field is used to prevent replay attacks, as discussed earlier.
Payload data	This variable-length field contains the transport layer segment (transport mode) or IP packet (tunnel mode), which is protected by encryption.
Padding	This field contains the padding bits, if any. These are used by the encryption algorithm, or for aligning the padding length field, so that it begins at the third byte within the 4-byte word.
Padding length	This 8-bit field specifies the number of padding bytes in the immediately preceding field.
Next header	This 8-bit field identifies the type of encapsulated data in the payload. For example, a value 6 in this field indicates that the payload contains TCP data.
Authentication data	This variable-length field contains the authentication data, called as the Integrity Check Value (ICV), for the datagram. This is calculated over the length of the ESP packet minus the Authentication Data field.

- **ESP transport mode:** Transport mode ESP is used to encrypt, and optionally authenticate the data carried by IP (for example, a TCP segment). Here, the ESP is inserted into the IP packet immediately before the transport layer header (i.e. TCP or UDP), and an ESP trailer (containing the fields Padding, Padding length and Next header) is added after the IP packet. If authentication is also used, the ESP Authentication Data field is added after the ESP trailer. The entire transport layer segment and the ESP trailer are encrypted. The entire cipher text, along with the ESP header is authenticated. This is shown in Fig. 9.35.

  We can summarize the operation of the ESP transport mode as follows.

  1. At the sender's end, the block of data containing the ESP trailer and the entire transport layer segment is encrypted and the plain text of this block is replaced with its corresponding cipher text to form the IP packet. Authentication is appended, if selected. This packet is now ready for transmission.
  2. The packet is routed to the destination. The intermediate routers need to take a look at the IP header as well as any IP extension headers, but not at the cipher text.
  3. At the receiver's end, the IP header plus any plain text IP extension headers are examined. The remaining portion of the packet is then decrypted to retrieve the original plain text transport layer segment.

- **ESP tunnel mode:** The tunnel mode ESP encrypts an entire IP packet. Here, the ESP header is prefixed to the packet, and then the packet along with the ESP trailer is encrypted. As we know, the IP header contains the destination address as well as inter-

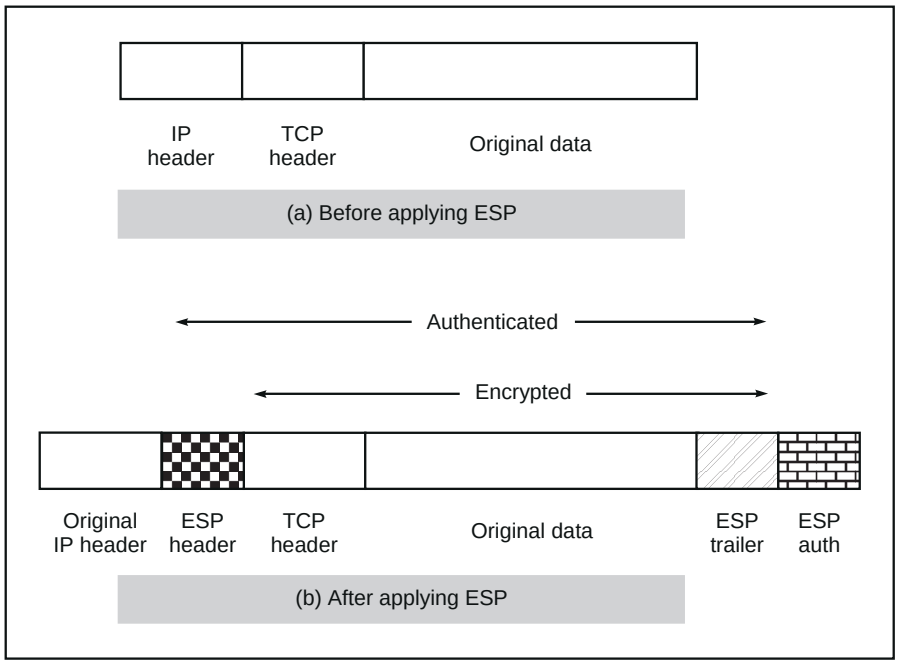

Fig. 9.35 *ESP transport mode*

mediate routing information. Therefore, this packet cannot be transmitted as it is. Otherwise, the delivery of the packet would be impossible. Therefore, a new IP header is added, which contains sufficient information for routing. This is shown in Fig. 9.36. We can summarize the operation of the ESP tunnel mode as follows.

1. At the sender's end, the sender prepares an inner IP packet with the destination address as the internal destination. This packed is prefixed with an ESP header, and then the packet and ESP trailer are encrypted and Authentication Data is (optionally) added. A new IP header is added to the start of this block. This forms the outer IP packet.
2. The outer packet is routed to the destination firewall. Each intermediate router needs to check and process the outer IP header, along with any other outer IP extension headers. It need not know about the cipher text.
3. At the receiver's end, the destination firewall processes the outer IP header plus any extension headers, and recovers the plain text from the cipher text. The packet is then sent to the actual destination host.

9.3.5 IPSec Key Management

1. *Introduction*

Apart from the two core protocols (AH and ESP), the third most significant aspect of IPSec is key management. Without a proper key management set up, IPSec cannot exist. This key management in IPSec consists of two aspects: key agreement and distribution. As we know,

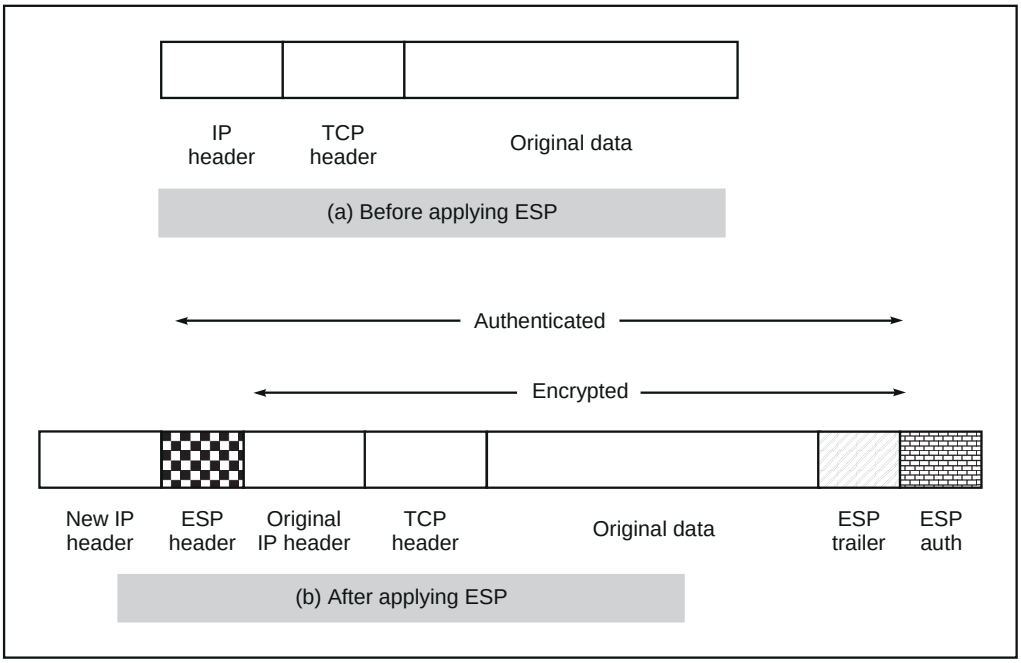

Fig. 9.36 *ESP tunnel mode*

we require four keys if we want to make use of both AH and ESP: two keys for AH (one for message transmissions, one for message receiving), and two keys for ESP (one for message transmissions, one for message receiving).

The protocol used in IPSec for key management is called as **ISAKMP/Oakley**. The **Internet Security Association and Key Management Protocol (ISAKMP)** protocol is a platform for key management. It defines the procedures and packet formats for negotiating, establishing, modifying and deleting SAs. ISAKMP messages can be transmitted via the TCP or UDP transport protocol. TCP and UDP port number 500 is reserved for ISAKMP.

The initial version of ISAKMP mandated the use of the Oakley protocol. Oakley is based on the Diffie-Hellman key exchange protocol, with a few variations. We will first take a look at Oakley, and then examine ISAKMP.

2. *Oakley key determination protocol*

The Oakley protocol is a refined version of the Diffie-Hellman key exchange protocol. We will not repeat the concepts of Diffie-Hellman, as we have already studied it in great detail. However, we will note here that Diffie-Hellman offers two desirable features:

- Creation of secret keys as and when required
- No requirement for any preexisting infrastructure

However, Diffie-Hellman also suffers from a few problems, as follows.

- No mechanism for authentication of the parties.
- Vulnerability to man-in-the-middle-attack.

- Involves a lot of mathematical processing. An attacker can take undue advantage of this by sending a number of hoax Diffie-Hellman requests to a host. The host can unnecessarily spend a large amount of time in trying to compute the keys, rather than doing any actual work. This is called as **congestion attack** or **clogging attack.**

The Oakley protocol is designed to retain the advantages of Diffie-Hellman, and to remove its drawbacks. The features of Oakley are as follows.

- It has features to defeat replay attacks.
- It implements a mechanism called as cookies to defeat congestion attacks.
- It enables the exchange of Diffie-Hellman public key values.
- It provides authentication mechanisms to thwart man-in-the-middle attacks.

We have already discussed the Diffie-Hellman key exchange protocol in detail. Here, we shall simply discuss the approaches taken by Oakley to tackle the issues with Diffie-Hellman.

- **Authentication:** Oakley supports three authentication mechanisms: digital signatures (generation of a message digest, and its encryption with the sender's private key), public key encryption (encrypting some information such as the sender's user id with the recipient's public key), and secret key encryption (a key derived by using some out-of-band mechanisms).
- **Dealing with congestion attacks:** Oakley uses the concept of cookies to thwart congestion attacks. As we know, in this kind of attack, an attacker forges the source address of a legitimate user and sends a public Diffie-Hellman key to another legitimate user. The receiver performs modular exponentiation to calculate the secret key. A number of such calculations performed rapidly one after the other can cause congestion or clogging of the victim's computer. To tackle this, each side in Oakley must send a pseudo-random number, called as cookie, in the initial message, which the other side must acknowledge. This acknowledgement must be repeated in the first message of Diffie-Hellman key exchange. If an attacker forges the source address, she does not get the acknowledgement cookie from the victim, and her attack fails. Note that at the most the attacker can force the victim to generate and send a cookie, but not to perform the actual Diffie-Hellman calculations.

The Oakley protocol provides for a number of message types. For simplicity, we shall consider only one of them, called as *aggressive key exchange*. It consists of three message exchanges between the two parties, say X and Y. Let us examine these three messages.

- ***Message 1*:** To begin with, X sends a cookie and the public Diffie-Hellman key of X for this exchange, along with some other information. X signs this block with its private key.
- ***Message 2*:** When Y receives *message 1*, it verifies the signature of X using the public key of X. When Y is satisfied that the message indeed came from X, it prepares an acknowledgement message for X, containing the cookie sent by X. Y also prepares its own cookie and Diffie-Hellman public key, and along with some other information, it signs the whole package with its private key.
- ***Message 3*:** Upon receipt of *message 2*, X verifies it using the public key of Y. When X is satisfied about it, it sends a message back to Y to inform that it has received Y's public key.

3. *ISAKMP*

Note The ISAKMP protocol defines procedures and formats for establishing, maintaining and deleting SA information. An ISAKMP message contains an ISAKMP header followed by one or more payloads.

The entire block is encapsulated inside a transport segment (such as TCP or UDP segment). The header format for ISAKMP messages is shown in Fig. 9.37.

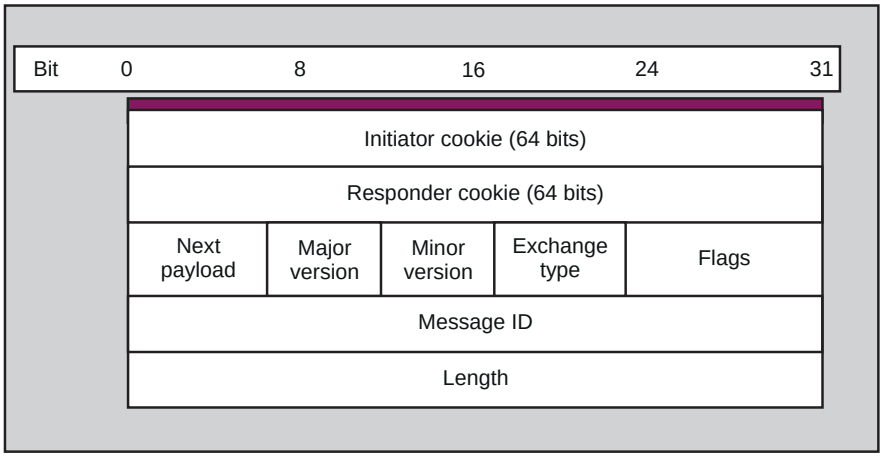

Fig. 9.37 *ISAKMP header format*

The fields in the ISAKMP header are shown in Table 9.5.

Table 9.5 *ISAKMP header field descriptions*

Field	*Description*
Initiator cookie	This 64-bit field contains the cookie of the entity that initiates the SA establishment or deletion.
Responder cookie	This 64-bit field contains the cookie of the responding entity. Initially, this field contains null when the initiator sends the very first ISAKMP message to the responder.
Next payload	This 8-bit field indicates the type of the first payload of the message (discussed later).
Major version	This 4-bit field identifies the major ISAKMP protocol version as used in the current exchange.
Minor version	This 4-bit field identifies the minor ISAKMP protocol version as used in the current exchange.
Exchange type	This 8-bit field indicates the type of exchange (discussed later).
Flags	This 8-bit field indicates the specific set of options for this ISAKMP exchange.
Message ID	This 32-bit field identifies the unique id for this message.
Length	This 32-bit field specifies the total length of the message, including the header and all the payloads in octets.

Let us quickly discuss the fields not explained yet.

- **Payload types:** ISAKMP specifies different *payload types*. For example, an *SA payload* is used to start establishment of an SA. The *proposal payload* contains information used during the SA establishment. The *key exchange payload* indicates for exchanging keys using mechanisms such as Oakley, Diffie-Hellman, RSA, etc. There are many other *payload types*.
- **Exchange types:** There are five *exchange types* defined in ISAKMP. The *base exchange* allows the transmission of the key and authentication material. The *identity protection exchange* expands the *base exchange* to protect the identities of the user. The *authentication only exchange* is used to perform mutual authentication. The *aggressive exchange* attempts to minimize the number of exchanges at the cost of hiding the user's identities. The *information exchange* is used for one-way transmission of information for SA management.

9.4 VIRTUAL PRIVATE NETWORKS (VPN)

9.4.1 Introduction

Until very recently, there has been a very clear demarcation between public and private networks. A public network, such as the public telephone system and the Internet, is a large collection of communicators who are generally unrelated with each other. In contrast, a private network is made up of computers owned by a single organization, which share information with each other. Local Area Networks (LAN), Metropolitan Area Networks (MAN) and Wide Area Networks (WAN) are examples of private networks. A firewall usually separates a private network from a public network.

Let us assume that an organization wants to connect two of its branch networks to each other. The trouble is that these branches are located quite a distance apart. One branch is in Delhi, and the other branch is in Mumbai. Two solutions out of all the available ones seem logical:

- Connect the two branches using a personal network, i.e. lay cables between the two offices yourself, or obtain a leased line between the two branches.
- Connect the two branches with the help of a public network, such as the Internet.

The first solution gives far more control and offers a sense of security, as compared to the second solution. However, it is also quite complicated. Laying cables between two cities is not easy, and is usually not permitted either. The second solution seems easier to implement, as there is no special infrastructure set up required. However, it also seems to be vulnerable to possible attacks. How nice it would be, if we could combine the two solutions!

Virtual Private Networks (VPN) offer such a solution. A VPN is a mechanism of employing encryption, authentication and integrity protection so that we can use a public network (such as the Internet) as if it is a private network (such as a physical network created and controlled by you). VPN offers high amount of security, and yet does not require any special cabling on behalf of the organization that wants to use it. Thus, a VPN combines the advantages of a public network (cheap and easily available) with those of a private network (secure and reliable).

A VPN can connect distant networks of an organization, or it can be used to allow traveling users to remotely access a private network (e.g. the organization's intranet) securely over the Internet.

A VPN is thus a mechanism to simulate a private network over a public network, such as the Internet. The term *virtual* signifies that it depends on the use of virtual connections. These connections are temporary, and do not have any physical presence. They are made up of packets.

9.4.2 VPN Architecture

The idea of a VPN is actually quite simple to understand. Suppose an organization has two networks, *Network 1* and *Network 2*, which are physically apart from each other and we want to connect them using the VPN approach. In such a case, we set up two firewalls, *Firewall 1* and *Firewall 2*. The encryption and decryption are performed by the firewalls. The architectural overview is shown in Fig. 9.38.

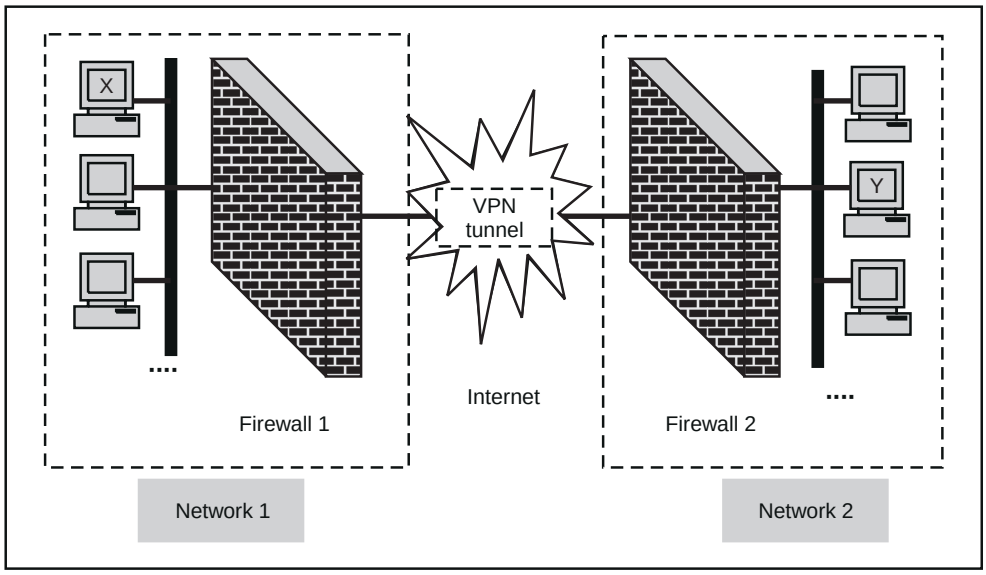

Fig. 9.38 *VPN between two private networks*

We have shown two networks, *Network 1* and *Network 2*. Network 1 connects to the Internet via a firewall named *Firewall 1*. Similarly, *Network 2* connects to the Internet with its own firewall, *Firewall 2*. We shall not worry about the configuration of the firewall here, and shall assume that the best possible configuration is selected by the organization. However, the key point here is that the two firewalls are *virtually* connected to each other via the Internet. We have shown this with the help of a *VPN tunnel* between the two firewalls.

With this configuration in mind, let us understand how the VPN protects the traffic passing between any two hosts on the two different networks. For this, let us assume that host X on *Network 1* wants to send a data packet to host Y on *Network 2*. This transmission would work as follows.

1. Host X creates the packet, inserts its own IP address as the source address, and the IP address of host Y as the destination address. This is shown in Fig. 9.39. It sends the packet using the appropriate mechanism.

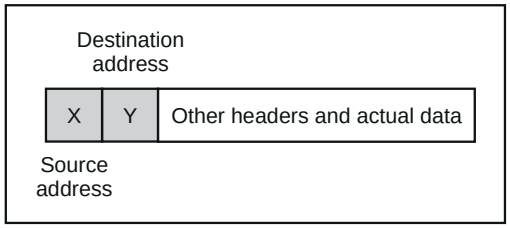

Fig. 9.39 *Original packet*

2. The packet reaches *Firewall 1*. As we know, *Firewall 1* now adds new headers to the packet. In these new headers, it changes the source IP address of the packet from that of host X to its own address (i.e. the IP address of *Firewall 1*, say F1). It also changes the destination IP address of the packet from that of host Y to the IP address of *Firewall 2*, say F2). This is shown in Fig. 9.40. It also performs the packet encryption and authentication, depending on the settings, and sends the modified packet over the Internet.

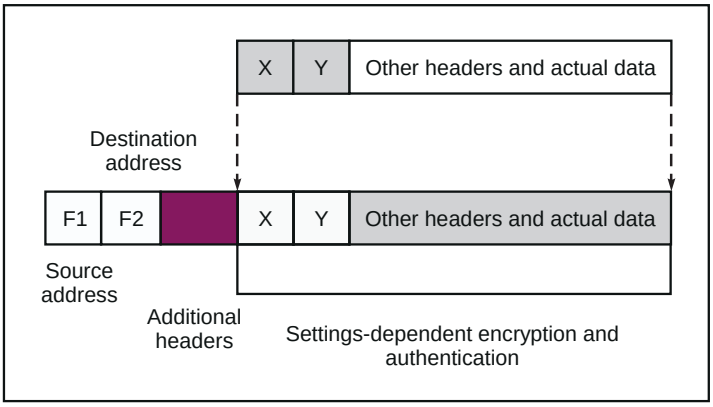

Fig. 9.40 *Firewall I changes the packet contents*

3. The packet reaches *Firewall 2* over the Internet, via one or more routers, as usual. *Firewall 2* discards the outer header and performs the appropriate decryption and other cryptographic functions as necessary. This yields the original packet, as was created by host X in step 1. This is shown in Fig. 9.41. It then takes a look at the plain text contents of the packet, and realizes that the packet is meant for host Y (because the destination address inside the packet specifies host Y). Therefore, it delivers the packet to host Y.

There are three main VPN protocols. A detailed study of these protocols is beyond the scope of the current text. However, we shall briefly discuss them for the sake of completeness.

- The **Point to Point Tunneling Protocol (PPTP)** is used on Windows NT systems. It mainly supports the VPN connectivity between a single user and a LAN, rather than between two LANs.
- Developed by IETF, the **Layer 2 Tunneling Protocol (L2TP)** is an improvement over PPTP. L2TP is considered as the secure open standard for VPN connections. It works

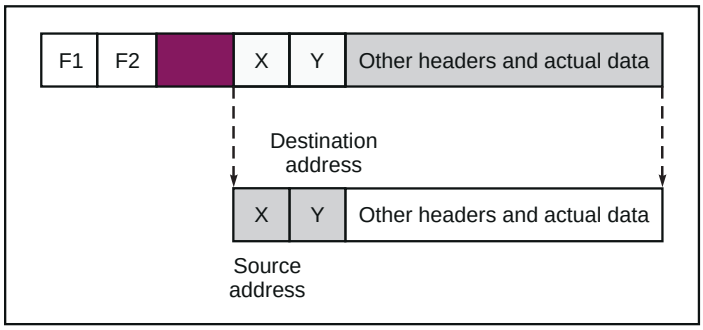

Fig. 9.41 *Firewall 2 retrieves the original packet contents*

for both combinations: user-to-LAN and LAN-to-LAN. It can include the IPSec functionality as well.

- Finally, IPSec can be used in isolation. We have discussed IPSec in detail earlier.

Chapter Summary

- *Corporate networks can be attacked from outside, or internal information can be leaked out.*
- *Encryption cannot prevent outside attackers from attacking a network.*
- *A firewall should be placed between a corporate network and the outside world.*
- *A firewall can be application gateway or packet filter.*
- *Dynamic packet filter (also called as stateful packet filter) adapts itself to the changing conditions.*
- *A circuit gateway creates a new connection between itself and the remote host.*
- *Firewall architectures combine the various types of firewalls in some combination.*
- *Screened subnet firewall is the strongest firewall architecture.*
- *IPSec provides security between the transport and the Internet layers.*
- *IPSec provides authentication and confidentiality services.*

Key Terms and Concepts

- Application gateway
- Bastion host
- Demilitarized Zone (DMZ)
- Encapsulating Security Payload (ESP)
- Internet Security Association and Key Management Protocol (ISAKMP)
- ISAKMP/Oakley
- Proxy server
- Authentication Header (AH)
- Circuit gateway
- Dynamic packet filter
- Firewall
- IP Security (IPSec)
- Packet filter
- Screened host firewall, Dual-homed bastion

- Screened host firewall, Single-homed bastion
- Screened subnet firewall
- Screening router
- Stateful packet filter
- Virtual Private Network (VPN)

Multiple-choice Questions

1. Firewall should be situated _____.
   (a) inside a corporate network
   (b) outside a corporate network
   (c) between a corporate network and the ourside world
   (d) none of these
2. Firewall is a specialized form of a _____.
   (a) bridge (b) disk
   (c) printer (d) router
3. A packet filter examines _____ packets.
   (a) all (b) no
   (c) some (d) alternate
4. _____ adapts itself to the changing conditions.
   (a) Stateless static filter (b) Static packet filter
   (c) Adaptive packet filter (d) Stateful packet filter
5. Application gateways are _____ packet filters.
   (a) less secure than (b) more secure than
   (c) equally secure to (d) slower
6. In _____, direct connections between the internal hosts and the packet filter are avoided.
   (a) Screened host firewall, Triple-homed bastion
   (b) Screened host firewall, Single-homed bastion
   (c) Screened host firewall, Null-homed bastion
   (d) none of the above
7. IPSec provides security at the _____ layer.
   (a) application (b) transport
   (c) network (d) data link
8. ISAKMP/Oakley is related to _____.
   (a) SSL (b) SET
   (c) SHTTP (d) IPSec

Review Questions

1. What can be the two main attacks on corporate networks?
2. List the characteristics of a good firewall implementation.
3. What are the three main actions of a packet filter?
4. How is a circuit gateway different from an application gateway?
5. What is the disadvantage of a Screened host firewall, Single-homed bastion?

6. How is Screened host firewall, Dual-homed bastion different from Screened host firewall, Single-homed bastion?
7. When is a Demilitarized Zone (DMZ) required? How is it implemented?
8. What are the limitations of a firewall?
9. What is the significance of tunnel mode?
10. What is a VPN?