

INTRODUCTION *to* NETWORK SECURITY

- Provides a single source that covers network security issues and uses the most common network protocols as detailed examples
- Explores the International Standards Organization's Open System Interconnect (ISO OSI) network stack and discusses common security weaknesses, vulnerabilities, attack methods, and mitigation approaches
- Includes CD-ROM with source code



INTRODUCTION TO NETWORK SECURITY

NEAL KRAWETZ



CHARLES RIVER MEDIA
Boston, Massachusetts

Core Course-XIV-17PCS10 NETWORK SECURITY AND CRYPTOGRAPHY

Credits: 4

Course Objectives:

- To introduce the classical encryption techniques for information hiding
- To analyze cryptographic techniques, protocols, formats, and standards

UNIT - I

Types of Physical Medium – Topologies – Wireless Networking: Wireless Protocols, Data Link Layer: Layered Data Link Protocols – SLIP and PPP – MAC and ARP, Network Layer: Routing Risks – Addressing – Fragmentation

UNIT - II

Internet Protocol: IP Addressing – ICMP – Security options. Transport Layer: Common Protocols – Transport Layer Functions – Gateways. TCP: Connection Oriented Protocols – TCP Connections – UDP. SSL: SSL Functionality – Certificates. SSH : SSH and Security – SSH Protocols. SMTP: E-Mail Goals – Common Servers.

UNIT - III

Security: Threat Models – Concepts – Common Mitigation Methods. Network Theory: Standards Bodies – Network Stacks – Multiple Stacks – Layers and Protocols – Common Tools. Cryptography: Securing Information – Authentication and Keys – Cryptography and Randomness - Hashes – Ciphers – Encryption – Steganography.

UNIT - IV

Data Encryption Techniques – Data Encryption Standards – Symmetric Ciphers. Public Key Cryptosystems – Key Management.

UNIT - V

Authentication – Digital Signatures – E-Mail Security – Web Security – Firewall.

TEXT BOOKS

1. Neal Krawetz, “Introduction Network Security”, India Edition, Thomson Delmar Learning, 2007.
2. V.K. Pachghare, “Cryptography and Information Security”, PHI Learning Private Limited 2009.

REFERENCE BOOKS

1. William Stallings, “Cryptography and Network Security”, Prentice – Hall of India, 2008.
2. Lincoln D. Stein, “Web Security”, Addison Wesley 1999.
3. Behrouz A. Forouzan, Cryptography and Network Security, Tata McGraw-Hill, 2007.



1 Physical medium

In This Chapter

- Types of Physical Mediums
- Topologies

1. TYPES OF PHYSICAL MEDIUMS

A *physical medium* is anything that can transmit and receive data. Data signal transmissions include modulating electric voltage, radio frequencies (RF), and light. The type of medium and modulation define the OSI layer 1 protocol. Sample protocols that are commonly used include wired protocols, fiber optics, high-volume trunk lines, dynamic connections (e.g., dialup), and wireless networks. The physical layer protocols include standards that describe the physical medium and the mechanisms for transmitting and receiving data across the medium.

1. Wired Network Protocols

Wired networks comprise some of the most common physical layer protocols. These wired networks connect end-user computers to servers and branching subnetworks from high-bandwidth trunk lines in buildings and small offices. The largest risks to wired networks come from broken cables, disconnected connectors, and eavesdropping due to direct physical access.

Coax cable was commonly used for wired networks. The 10Base-2 standard, also known as thinnet or Ethernet, used a 50-ohm coaxial cable (RG58) to transmit amplitude modulated RF signals. The designation “10Base-2” denotes a maximum throughput of 10 million bits per second—10 megabits or 10 Mbps—over two

wires (coaxial cable). The maximum cable length was 185 meters. A similar standard—10Base-5 (thicknet)—also used RG58 but had a maximum cable length of 500 meters. Thicknet was uncommon outside of most corporate infrastructures.

10Base-T and 100Base-T use twisted-pair wire such as category 3 (Cat3), Cat5, or Cat5e cable with an RJ45 connector. These standards use a low-voltage alternating current to transmit data at 10 Mbps and 100 Mbps, respectively. Faster networks, such as 1000Base-T, can provide Gigabit Ethernet over Cat5e or Cat6 cable. The maximum cable length depends on the category of cable used and the speeds being transmitted.

Cable modems are common for residential service. These use a variety of standards that manage an amplitude modulation RF signal over a coaxial cable (RG57). The data signals operate using different protocols for downstream (64-QAM, 256-QAM) and upstream (QPSK, 16-QAM). Higher layer protocols such as DOCSIS and NTSC manage the modem authentication and data transfer.

2. Fiber-Optic Networks

Fiber-optic cables carry pulses of light. This medium is commonly used with the Fiber Distributed Data Interface (FDDI) protocol. FDDI is an OSI layer 2 protocol and operates as a high-speed token ring, achieving 100 Mbps and faster. High-speed networks, such as 10-Gb Ethernet, also commonly use fiber-optic cables.

Fiber-optic networks are generally more expensive than wired networks but provide much higher bandwidths. Unlike wired networks, fiber optics are not electrically conductive. If a power surge strikes one node on a wired network, there is a potential for blowing out every node on the network. In contrast, a surge to a node on a fiber network only impacts that node. Fiber networks are commonly found in high-bandwidth and mission-critical environments.



Surge protectors and uninterruptible power supplies (UPS) are commonly used to mitigate the impact from power spikes and brownouts. But some surge protectors do not provide protection from lightning strikes. Be sure to read the owner's manual, terms, and conditions carefully.

3. Trunk Lines

Trunk lines connect major network hubs with other hubs, providing very large bandwidths. Some trunk lines use copper wire, and others use fiber-optic cable. As an example, T-1, T-3, and FT-1 are trunk phone line connections, and the number indicates the type. A *T-1* is a “trunk level 1” link. It contains 24 channels, with each channel supporting a data rate of 64 Kbps. A *T-3* is a “trunk level 3” link, consisting of three T-1 links, or 672 separate channels. A *fractional T-1* (FT-1) consists of

a subset of channels from a T-1 connection and is usually used for leased lines. The physical medium for a T-1 line may be copper wire or fiber-optic cable, and may vary by region. The standards for trunk links also vary by country: a European E-3 consists of 480 channels and is similar to a Japanese J-3 (also 480 channels but a slightly lower throughput).

Other types of trunk lines include OC-1, OC-3, OC-12, and OC-48. These optical cable networks are commonly used by phone companies and can achieve over 2.4 Gbps (OC-48).

4. Dynamic Networks

Not every network medium is static and always connected. *Dynamic networks* provide a large portion of home and small office network connections and are used when static networks are not required. Dynamic networks generally consist of a *modem* (MODulator and DEModulator) for connecting to an Internet service provider (ISP). A standard modem sends a data signal over a voice phone line. Dozens of subprotocols for modems include speed (V.34, V.90, K56Flex), error correction (V.42, MNP 2-4), and compression (V.42bis, V.44, MNP 5). A *digital subscriber line* (DSL) is a high-speed digital phone connection over a Plain Old Telephone System (POTS) link. For higher bandwidths, there is ISDN; the *Integrated Services Digital Network* is a type of FT-1 link that includes two 64 Kbps lines (B channels) and one 16 Kbps control channel (D channel).

Dynamic networks are generally less expensive for end consumers, but they do not provide the same high bandwidth as wired, fiber-optic, or trunk network connections. But these networks have one significant security benefit: they are not always connected. When the modem disconnects from the ISP, the network is no longer vulnerable to remote network attacks.

5. Wireless

Wireless networks are frequently used in places where traditional wired, fiber-optic, and trunk lines are unavailable, too costly, or too difficult to install. Rather than wiring an entire home for using a network, wireless networks permit the computer to be anywhere as long as it can access the wireless *hub*'s radio signal.

IEEE 802.11, also called *wireless fidelity* (WiFi), defines a suite of protocols commonly used for wireless networking. These protocols usually include a suffix that defines the frequency and throughput. For example, 802.11a uses the 5 GHz spectrum and can potentially achieve 54 Mbps. Both 802.11b and 802.11g use the 2.4 GHz spectrum, and may reach 11 Mbps and 54 Mbps, respectively.

Wireless networks can be very inexpensive to deploy because they require only a network *access point* (AP, or "hotspot") and a wireless network interface (SP, or *subscriber point*). Wireless networks do not require any physical medium installa-

tion such as cables. But this lack of physical infrastructure means that the network signal is not contained. A remote attacker who can receive the AP radio signal can attack the physical network. In addition, *radio frequency interference* (RFI) may diminish the effective range and throughput of a wireless AP. For example, using a 2.4 GHz cordless phone near an 802.11g hub may dramatically impede the network's range and speed.

(

5.4 TOPOLOGIES

The physical link mediums and components combine to form the physical network. The ways they combine determine the network topology (layout). The four commonly used network topologies (Figure 5.1) are bus, star, ring, and broadcast.

Each of the topologies offers a tradeoff between usability and security. Small networks may use single topology architectures, but large networks generally appear as a hybrid where connectors join different topologies.

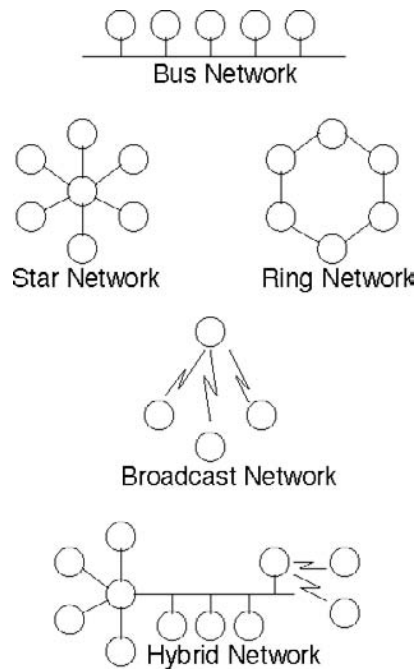


FIGURE 5.1 The five network topologies: bus, star, ring, broadcast, and hybrid.

5.4.1 Bus Networks

A *bus network* consists of a single link that includes all nodes. In a bus architecture, any transmission on the network is received by every node on the network. The primary benefit from a bus network is simplicity: nodes can be readily added and removed without significantly impacting the network. This benefit impacts security, however, in that any node on the network can readily eavesdrop on traffic from every node on the network, and a single physical DoS attack (from cutting the network to RFI) affects all nodes on the network. In addition, bus networks can suffer from performance degradation if too many nodes attempt to transmit at once. Network collisions, when two nodes transmit data at the same time, are common and can lower overall network performance. For example, two nodes (one transmitting and the other receiving) on a 10 Mbps link may benchmark their throughput at nearly 10 Mbps. But four nodes using only a single 10 Mbps bus network may benchmark their throughput at less than 5 Mbps due to network collisions.

5.4.2 Star Networks

A *star network* topology relies on a central hub for communicating with nodes. Unlike bus networks, nodes are not connected to the same physical branch and cannot necessarily observe all data from all nodes. The physical network topology of star networks does not necessarily correspond with the network traffic topology. For example, in a twisted-pair network every RJ45 connector links a node (computer) to a central hub—this is physically a star network.

Hubs usually cannot distinguish where the transmitted data needs to go, so they act as simple connectors/repeaters. They allow all branches of the star architecture see all traffic and the data flow looks like a bus network. But if the hub acts as a bridge or switch (OSI layer 2), then it can interpret the network data and direct it to the correct branches of the star network. In this configuration, the nodes may not see all traffic. If the nodes do not see all network traffic, then the threat from eavesdropping, insertion, and replay attacks is reduced.

Star networks differ from bus networks in several significant ways. Based on cost and extendibility, star networks are usually not as effective as bus networks because nodes cannot be extended off other nodes, as in a bus extension. Moreover, hubs have a limited number of ports. When all ports are used, the hub introduces an added cost for expansion. Star networks, however, can be more efficient than bus networks. Most hubs act as amplifiers, ensuring that a node can be physically far away from other nodes on the network. In contrast, a bus network's maximum distance is physically as long as the distance between the furthest nodes.

From a security viewpoint, star networks can be more resilient to DoS. RFI or network problems with one branch of the network may not propagate past the central hub. Star networks that use star topologies for data flow are more secure than bus networks because a single node cannot readily eavesdrop on neighboring nodes.

Hubs can provide a centralized point for authenticating new network nodes. In a bus network, any node can attach to the bus; authentication happens between nodes, not along the bus. In contrast, star networks may use the hub for authenticating new nodes before accessing a network. For example, modem pools operate as a star network hub, with each phone line acting as one branch. Modem pools may validate new nodes through Caller-ID, username/password authentication, or smart tokens (cards that generate random code sequences). The person dialing in is not granted access to the network until the hub authenticates the caller. The hub includes the option to authenticate new nodes prior to accessing any network layer. This authentication may occur within any of the higher OSI layers, but generally does not occur within the physical network layer.

Unfortunately, star networks have one significant limitation: the hub acts as a central point of failure. If the hub becomes inaccessible for any reason, then the network collapses.

5.4.3 Ring Networks

The *ring network* topology moves the hub from the center of the network to each node. Each node in a ring network contains two connections that lead to two neighboring nodes. Data is passed through the neighboring nodes until it reaches the destination. Ring networks typically maintain two paths between each node: clockwise and counter-clockwise around the ring. As with a bus network, new nodes can be easily added without disturbing the data flow. (In contrast, star networks can result in downtime if the hub runs out of ports and must be expanded.) The result is a robust network: any single network outage will not disrupt the entire ring architecture.

From a security perspective, ring networks are midway between a bus and star network. A single node on the ring can only eavesdrop on the traffic that it receives. Because half of the traffic likely takes a route that does not include the node, the node cannot eavesdrop on all the traffic. In addition, the impact from DoS is limited to the response by the adjacent nodes; if the neighbors do not propagate the DoS, then the DoS has little impact.

As with star networks, ring networks can either be implemented in a physical topology or data link topology. An example of a data link topology is a Token Ring network. In this configuration, all nodes connect to a media access unit (MAU). The physical layout appears similar to a star network, with the MAU acting as a central hub. Every node receives all data on the token ring. The data flow resembles a bus network, but the MAU uses a media access flag (token at OSI layer 2) to indicate when a node may transmit data. A Token Ring network is a tight integration between OSI layer 1 and layer 2.

Ring networks are not necessarily limited to two NICs per node. Although rare, these networks can be expanded to form nets, meshes, and fully connected networks. A two-dimensional grid network consists of four NICs per node, arranged in a grid fashion. Grid routing procedures, such as NW-first (send data North then West before trying South or East), can be very complex. Beyond the installation costs and expansion impact, these networks introduce an extreme level of robustness by providing multiple paths between nodes. They also can improve network speed by splitting data across different routes, and they can improve security by reducing the impact from eavesdropping and DoS.



Some Token Ring network cards include multiple channels in one physical device. For example, a single IBM Token Ring card supplies two channels, reducing the number of necessary network cards.

Expensive Rings

Ring networks can be very robust, but robustness has a cost: implementation expense and speed. Physical layer ring networks require two NICs per node, and many types of ring networks require specialized hardware. This additional cost can become prohibitive for large networks.

Ring networks are usually not as fast as bus networks or star networks. In a bus network, data is transmitted on the bus and received by the destination system (and every other system). The result is that data transmits very quickly. For a star network, the data traverses one relay system: the hub. The result is a small delay between the data arriving at the hub and being relayed out. Ring networks require the data to be relayed multiple times as it passes through adjacent nodes. This results in a slower data transmission rate—usually by a few microseconds. For example, a few nodes on a 10 Mbps bus network transfers data faster than a 10 Mbps ring network. Token Ring network delays are equivalent to relaying data through a series of network bridges.

5.4.4 Broadcast Networks

Broadcast networks are used when data is transmitted over radio frequencies rather than over a point-to-point wire. These networks are very desirable due to their low cost and simple setup—a network only needs a hub (access point, or AP) and a network card containing a transmitter (subscriber point, or SP); they do not require cables linking every node. The only requirement is for the radio signal to be strong enough between the AP and SP.

Broadcast and wireless networks extend on the limitations from the bus network. For example, in a bus network, every node can receive all transmitted data. In a wireless network, anyone with a radio receiver can receive all transmitted data. Similarly, bus networks have limited throughput due to network collisions, but network collisions can be detected. In a wireless network, two distant nodes may not be able to hear each other transmitting. In this situation, the AP either becomes unable to distinguish one signal from another, resulting in a network collision, or the AP only receives the stronger signal; the weaker signal is effectively shut out.

Bus networks are very vulnerable to DoS from network interference; interference along the network impacts every node. Broadcast networks become even more susceptible because RFI is no longer limited to noise inserted on a local network bus. Something as innocuous as a neighbor's refrigerator may interfere with your local wireless network.

In addition to the similar limitations to bus networks, broadcast networks include many unique security risks. These are detailed in Chapter 7 and include sniffing, impersonation, and antenna placement.

5.4.5 Hybrid Networks

A *hybrid network* connects multiple network architectures to permit a combination of benefits. For example, a company may use a star architecture for connecting departments but a bus architecture within each department. The benefits for this example include cost, speed, and security.

A hybrid network can be more cost effective than a star network because a single hub is less expensive than supplying a hub for everyone in a department. Using star topology between divisions and a bus within a department limits the cost.

Hybrid topologies can distribute network loads, resulting in faster networks. An active node within a department may impact the bus speed for that department, but the impact will not be felt among other departments.

Hybrid topologies reduce security risks from homogenous networks. A risk impacting one department is compartmentalized to that department. For example, an eavesdropper in the human resources department cannot spy on data from the finance department. Similarly, a hub that becomes disabled may not impact an entire corporation.

Hybrid networks are commonly used to mitigate risks. In Part IV, we will cover how OSI layer 3 uses network segmentation and hybrid architectures to limit system compromises.



Wireless Networking

In This Chapter

- Wireless Protocols



2. WIRELESS PROTOCOLS

The 802.11 specification is part of the IEEE 802 suite of standards that define both medium (wireless frequencies or physical wires) and device identification. Because many wireless devices may share the same frequency, 802.11 defines how a wireless network *subscriber point* (SP) identifies the correct wireless network *access point* (AP). These include a service set identifier (SSID) and wired equivalent privacy (WEP).

1. SSID

The *service set identifier* (SSID) is a 32-character text string used to identify an AP and distinguish it from other APs. For example, 802.11b defines a frequency range (2.4 GHz) and 11 channels within that range. An AP may be placed on any single channel, but many APs may share the same channel. The SSID is used to distinguish APs that share a channel.

2. WEP

IEEE 802.11 defines the *wired equivalent privacy* (WEP) protocol, providing a degree of security to wireless networks. In a wired network, privacy is limited to

physical access—if an attacker cannot gain physical access to the network, then the physical network is likely secure. In contrast, wireless networks broadcast a radio signal. Anyone who can receive the signal immediately gains physical access to the medium. WEP defines a cryptographic authentication system that deters unauthenticated SP access. The WEP cryptography includes keys and encryption. But, the cryptographic algorithm specified by WEP is weak and easily compromised.

1. WEP Keys

WEP supports two types of encryption: 64 bit and 128 bit. These correspond with 40-bit and 104-bit length secret keys, respectively. These *keys* are used to authenticate network access.

There are two ways to create the secret key. The first method simply allows the user to enter in the 8- or 16-character hexadecimal number that represents the key; however, this is not a convenient method for most people. As an alternative, many AP configurations permit the use of a text-based password for generating the secret keys. A text password is hashed into a 40-bit (or 104-bit) encryption key. To maintain device compatibility, nearly every wireless vendor implements the same hash functions. The same text password should generate the same key independent of the vendor.

Regardless of the generation method, the AP and SP must have the same key. Although this key is never transmitted, it is used for encrypting the wireless data stream.

2. WEP 40-Bit Password Hash



The algorithm used to convert a text string into a secret key differs based the encryption strength. The CD-ROM contains source code for WEP-password—a program that generates WEP keys based on text strings.

For the 40-bit encryption, a simple pseudo-random number generator is seeded based on the password (Listing 7.1). The generator creates four variants from the single password, any of which may be used as the WEP key.

LISTING 7.1 Source Code to Generate a 40-Bit WEP Key from a Text String

```

/*****
Create40bit(): Given a string, generate a 40-bit
(64bit WEP) public key. Display the key in hex.
4 keys are created for every 1 password.
*****/
void create40bit (char *Text)
{
    int i,j;
    int Seed[4] = {0,0,0,0};
    int TextLen;

```



```

unsigned char Result[16];

/* Initialize the password by concatenating the same string
   until we reach 64 bytes */
TextLen = strlen(Text);
for(i=0; i<64; i++) PasswordBuffer[i] = Text[i%TextLen];

/* Initialize the MD5 context */
MD5_Init(&Context);

/* Create the 128-bit signature */
MD5_Update(&Context,PasswordBuffer,64);
MD5_Final(Result,&Context);

/* Display only the first 104 bits (13 bytes) */
printf(" Key: ");
for(i=0; i<13; i++) printf("%02x",Result[i]);
printf("\n");
} /* Create104bit() */

```

7.2.2.4 WEP Encryption

WEP encryption uses the RC4 stream cipher encryption algorithm. For each packet being transmitted, an *initial vector* (IV) is generated. For 64-bit WEP, the IV is 14 bits long; the IV for 128-bit encryption is 24 bits long. The IV is combined with the secret key to seed the RC4 encryption. RC4 then encrypts the data. The transmitted packet includes the unencoded IV, the encoded data stream, and a CRC checksum. In most cases, the IV is changed between every packet to prevent data repetition.

When the AP or SP receives data, the encryption process is reversed. The RC4 algorithm combines the secret key with the unencoded IV that was transmitted with the packet. This combination is used to decode the encrypted data. The final CRC is checked to validate the decoded data.

7.2.3 WEP Cracking

Although the concept behind WEP encryption is solid, the implementation uses weak security elements. In particular, there are relatively few IV values. For attackers to crack the WEP encryption, they only need to determine the secret key. There are two main approaches for cracking WEP: brute-force password guessing and data analysis.



Although RC4 is not considered an extremely strong encryption algorithm by today's standards, the weaknesses in WEP are not primarily centered on RC4. The WEP weaknesses are due to weak key and IV selections.

1. Brute-Force WEP Cracking

The WEP password is usually based on a hash from a dictionary or common word. Simply guessing passwords and encoding them as a WEP key may quickly crack a WEP system. A *dictionary attack* cycles through a word list, trying every word as a possible key.

WEP and 802.11 define no method for deterring dictionary attacks—an attacker can try thousands of keys without ever being denied access and without ever having the AP generate a log entry concerning a possible attack. From the AP's viewpoint, there is no distinction between a corrupt packet due to a CRC error (e.g., poor radio signal) and a corrupt packet due to a decryption problem (brute-force key attack).

Many APs are configured using weak passwords. These may include people's names, addresses, or manufacturer brands. Amazingly, one security professional reported that a significant number of WEP keys are the same as the SSID. If the SSID says "ABCCorporation" then the WEP key may be the text string "ABCCorporation."

2. Data Analysis WEP Cracking

WEP encryption makes one weak assumption: if an attacker does not see the same IV, then he cannot crack the data stream. In reality, IV values repeat. For 64-bit encryption, there are only 4,096 (2^{12}) different IV values. If the IV does not change between packets, then a duplicate is immediately available. But if the IV changes between each packet, then a duplicate IV will be observed after no more than 4097 packets. When an attacker captures two packets with a duplicate IV, it is just a matter of trying different key sequences to find the ones that result in an RC4 decryption with the correct CRC checksum. By assuming weak passwords for creating the secret key, the search process can be sped up.

Given two packets with the same IV, the entire 64-bit WEP analysis can take a few minutes. 128-bit encryption may take a few hours, depending on the computer's speed. The primary limiting factor is packet volume: if there is very little network traffic, then an attacker must wait a long time before seeing a duplicate IV. But a patient attacker will eventually see a duplicate IV. And given a duplicate IV, it is only a matter of time before an attacker determines the secret key.



Data Link Layer

In This Part

- Data Link Protocols
- SLIP and PPP
- MAC and ARP

3. **LAYERED DATA LINK PROTOCOLS**

For point-to-point networks, a single data link layer protocol, such as SLIP, can perform all of the required functionality; however, few data link layer protocols span the entire layer. Instead, the data link layer usually contains multiple protocols that, together, provide the full data link functionality. The lower protocols manage the physical layer communications. The middle layer protocols manage routing and addressing, and upper layer protocols control network discovery.

1. **Low-Level Protocols**

The lower level data link protocols are directly associated with the physical layer. Examples include the following:

FDDI: This layer 2 protocol acts as a high-speed Token Ring over fiber-optic networks.

LAPF: The standard ITU Q.921 defines the Link Access Protocol for Frame Mode Services. This frame relay protocol is commonly used on ISDN, F-T1, and faster networks.

PPP: The Point-to-Point Protocol commonly connects dynamic connections, such as modem or ISDN.

Carrier Sense Multiple Access/Collision Detection: One of the most common protocols, CSMA/CD is part of the IEEE 802.3 standard and is used for most end-user Ethernet traffic. This protocol determines when data may be transmitted and detects when collisions occur. The physical layer is commonly twisted pair (10Base-T or 100Base-T), or coax such as 10Base-2 or 10Base-5.

8.3.2 Middle-Level Protocols

The middle protocols within the data link layer manage addressing. These are closely associated with specific lower-level layer 2 protocols. For example, IEEE 802.2 defines two data link sublayers: MAC and LLC. The MAC defines a unique address on a particular network. The various 802 standards (802.3, 802.4, 802.5, etc.) define the management of the MAC address. In particular, different types of networks may use different types of MAC addressing. The logical link control (LLC) is the higher portion that provides a consistent interface regardless of the MAC format.

8.3.3 High-Level Protocols

The high-level data link layer protocols manage address discovery. For example, when using MAC addressing, the ARP is used to identify the MAC address of a particular host. Because bridges may identify adjacent bridges, the IEEE 802.1 spanning tree protocol ensures that network loops do not lead to network feedback (where the same bits go round and round).

Other higher-level protocols include the AppleTalk Address Resolution Protocol (AARP) and the multilink protocol (MP) for X.25 networks.



SLIP and PPP

In This Chapter

- Simplified Data Link Services
- Point-to-Point Protocols
- Common Risks
- Similar Threats

1. SIMPLIFIED DATA LINK SERVICES

In a point-to-point network, many of the complexities from a multinode network vanish, and core functionality can be simplified. This includes simpler framing methods, flow control, and address support.

1. Simplified Flow Control

The physical layer ensures that the data received matches the data transmitted, but it does not address transmission collisions. When two nodes transmit at the same time, data becomes overwritten and corrupted. The data link layer determines when it appears safe to transmit, detects collisions, and retries when there are data errors. This is required functionality on a multinode network because any node may transmit at any time.

In a ring network, such as Token Ring or FDDI, the data link layer arbitrates when data is safe to transmit. Without this arbitration, any node could transmit at any time.



In a point-to-point network, there are only two nodes, so arbitration and flow management can be dramatically simplified. The most common types of point-to-point networks use simplex, half-duplex, or full-duplex connections.

1. Simplex

A *simplex* channel means that all data flows in one direction; there is never a reply from a simplex channel. For networking, there are two simplex channels, with one in each direction. Because there is never another node transmitting on the same channel, the flow control is simplified: there is no data link layer flow control because it is always safe to transmit. Examples of simplex networks include ATM, FDDI, satellite networks, and cable modems.

2. Half-Duplex

Half-duplex channels are bidirectional, but only one node may transmit at a time. In this situation, either a node is transmitting, or it is listening. It is very plausible for one node to dominate the channel, essentially locking out the other node. In half-duplex networks, each node must periodically check to see if the other node

wants to transmit. Examples of half-duplex networks include some dialup, serial cable, and wireless network configurations.

9.1.1.3 Full-Duplex

In a *full-duplex* network, one channel is bidirectional, but both nodes can transmit at the same time. Examples include many fiber-optic systems, where the transmitted light pulses do not interfere with the received light pulses. Some FDDI, optical cable networks, and telephone modem protocols use full-duplex channels. When using a full-duplex channel in a point-to-point network, the data link flow control system is unnecessary because it is always clear to transmit.

2. Simplified Message Framing

Message framing ensures that the transmitted data is received in its entirety. Although the physical layer ensures that the data received matches the data transmitted, simultaneous transmissions can corrupt the data. The message frame identifies possible collisions, but the flow control from point-to-point networks limits the likelihood of a collision. In point-to-point networks, the message frame only needs to indicate when it is clear for the other side to transmit.

3. Simplified Address Support

In a multinode network, the message frame must be addressed to the appropriate node. In a point-to-point network, however, all transmitted data are only intended for the other node on the network. As such, hardware addresses are not required. For backward compatibility, the LLC may report the hardware address as being all zeros or a set value based on the current connection's session. For example, a user with a dialup modem may have the data link layer assign a random MAC address to the connection. After the connection terminates, the next connection may have a different MAC address. In many implementations, the data link layer's MAC address may appear to be the system's assigned IP address; an assigned dialup IP address of 1.2.3.4 may generate the MAC address 00:00:01:02:03:04. The next time the user dials into the service provider, the system may be assigned a new IP address and generate a new MAC address.

9.2 POINT-TO-POINT PROTOCOLS

Two common point-to-point protocols are SLIP and PPP. These are commonly used over direct-wired connections, such as serial cables, telephone lines, or high-speed dialup such as DSL. Besides direct-wired connections, they can also be used for VPN tunneling over a network.

1. SLIP

The *Serial Line Internet Protocol* (SLIP) is defined by RFC1055. This simple protocol was originally designed for serial connection. SLIP provides no initial headers and only offers a few defined bytes:

END: The byte `0xc0` denotes the end of a frame.

ESC: The byte `0xdb` denotes an escape character.

Encoded END: If the data stream contains the `END` byte (`0xc0`), then SLIP re-encodes the character as `0xdb 0xdc`.

Encoded ESC: If the data stream contains the `ESC` byte (`0xdb`), then SLIP re-encodes the character as `0xdb 0xdd`.

When the OSI network layer has data to transmit, SLIP immediately begins transmitting the data. At the end of the transmission, an `END` byte is sent. The only modifications to the data are the encoded escape sequences, which are used to prevent the interpretation of a premature `END`.

Although SLIP is one of the simplest data link protocols, it leaves the network open to a variety of risks, including error detection, data size issues, network service support, and configuration options.

1. Error Detection

SLIP contains no mechanism for error detection, such as a framing checksum. SLIP assumes that higher layer protocols will detect any possible transmission corruption. This can lead to serious problems when the physical layer is prone to errors (e.g., noisy phone line) or when the network layer uses a simple checksum system.

2. Maximum Transmission Units

The *maximum transmission unit* (MTU) defines the largest size message frame that is permitted. Larger message frames are usually rejected in lieu of causing a buffer overflow in the data link driver. The SLIP message frame has MTU restrictions. The receiving system must be able to handle arbitrarily large SLIP message frames.

As a convention, most SLIP drivers restrict frames to 1006 bytes (not including the `END` character). This size is based on the Berkeley Unix SLIP drivers, a de facto standard. But nothing in the protocol specifies an actual maximum size.

SLIP also permits transmitting an `END END` sequence, resulting in a zero-length datagram transmission. Higher layer protocols must be able to handle (and properly reject) zero-length data.

3. No Network Service

Many multinode data link protocols include support for multiple network layer protocols. When data is received, the information is passed to the appropriate network layer driver. SLIP contains no header information and no network service identifier. As such, SLIP is generally used with TCP/IP only.

4. Parameter Negotiations

A SLIP connection requires a number of parameters. Because it relies on a TCP/IP network layer, both nodes need IP addresses. The MTU should be consistent between both ends of the network, and the network should authenticate users to prevent unauthorized connections.

Unfortunately, SLIP provides no mechanism for negotiating protocol options or authenticating traffic. The user must configure the SLIP driver with an appropriate network layer IP address and MTU. Flow control, such as simplex or half-duplex, is determined by the physical layer.

Authentication for SLIP connections generally occurs at the physical layer. As users connect to the service provider, they are prompted for login credentials. After authentication, the SLIP connection is initiated. After authenticating, the service provider may supply the appropriate MTU and IP addresses before switching to the SLIP connection.

SLIP does not support other options, such as encryption and compression.

2. PPP

In contrast to SLIP, the *Point-to-Point Protocol* (PPP) provides all data link functionality. Many RFCs cover different aspects of PPP: RFC 1332, 1333, 1334, 1377, 1378, 1549, 1551, 1638, 1762, 1763, and 1764. The main standard for PPP is defined in RFC1661.

The PPP message frame includes a header that specifies the size of data within the frame and type of network service that should receive the frame. PPP also includes a data link control protocol for negotiating IP addresses, MTU, and basic authentication. In addition, PPP provides link quality support, such as an echo control frame for determining if the other side is still connected. Unfortunately, PPP does not provide a means to protect authentication credentials, detect transmission errors, or deter replay attacks.

1. Authentication

For authentication, PPP supports both PAP and CHAP, but only one should be used. The PPP driver should first attempt the stronger of the two (usually CHAP) and then regress to the other option if the authentication method is unavailable. Although CHAP and PAP both provide a means to transmit authentication creden-

tials, neither provides encryption; the authentication is vulnerable to eavesdropping and replay attacks at the physical layer.

Neither PAP nor CHAP natively supports password encoding. Some extensions to these protocols, such as DH-CHAP (Diffie-Hellman negotiation for hash algorithms), MS-CHAP and MS-CHAPv2 (Microsoft NT hashing), and EAP-MS-CHAPv2 (includes the use of the Extensible Authentication Protocol), extend the authentication system to use one-way encryption rather than a plaintext transfer.

Although PPP is vulnerable to physical layer attacks, these attacks are rare. The larger risk comes from the storage of login credentials on the server. For PAP and CHAP, the login credentials must be stored on the PPP server. If they are stored in plaintext, then anyone with access to the server may compromise the PAP/CHAP authentication. Even if the file is stored encrypted, the method for decrypting the file must be located on the PPP server.

2. Transmission Error Detection

Although PPP includes a frame header and tail, it does not include a checksum for frame validation. The recipient will not detect frames modified in transit. Noisy phone lines, bad network connections, and intentional insertion attacks are not detected. As with SLIP, PPP assumes that error detection will occur at a higher OSI layer.

3. Replay Attack Transmission

PPP's message frame header contains packet and protocol information but not sequence identification. Because PPP was designed for a point-to-point network, it assumes that packets cannot be received out of order. Nonsequential, missing, and duplicate packets are not identified by PPP. As with SLIP, PPP assumes that a higher OSI layer will detect these issues.

Although these are not significant issues with serial and dialup connections, PPP can be used with most physical layer options. If the physical layer permits nonsequential transmissions, then PPP may generate transmission problems. Examples include using PPP with asynchronous transfer mode (ATM) or grid networks. In these examples, data may take any number of paths, may not be received sequentially, and in some situations may generate duplicate transmissions. Similarly, PPP may be used as a tunnel protocol. Tunneling PPP over UDP (another asynchronous protocol) can lead to dropped or nonsequential PPP message frames. Ideally, PPP should be used with physical layer protocols that only provide a single route between nodes.

An attacker with physical access to the network may insert or duplicate PPP traffic without difficulty. This opens the network to insertion and replay attacks.

9.2.2.4 Other Attacks

As with SLIP, PPP provides no encryption. An attacker on the network can readily observe, corrupt, modify, or insert PPP message frame transmissions. Because the network is point-to-point, both ends of the network can be attacked simultaneously.

3. Tunneling

VPNs are commonly supported using PPP (and less common with SLIP). In a VPN, the physical layer, data link layer, and possibly higher OSI layers operate as a virtual physical medium. A true network connection is established between two nodes, and then PPP is initiated over the connection, establishing the virtual point-to-point network. Common tunneling systems include PPP over SSH and PPP over Ethernet (PPPoE). When tunneling with PPP or SLIP, the tunneled packet header can degrade throughput performance. CPPP and CSLIP address this issue.

1. PPP over SSH

Secure Shell (SSH—described in Chapter 20) is an OSI layer 6 port forwarding protocol that employs strong authentication and encryption. SSH creates a single network connection that is authenticated, validated, and encrypted. Any TCP (OSI layer 4) connection can be tunneled over the SSH connection, but SSH only tunnels TCP ports. Under normal usage, the remote server's network is not directly accessible by the SSH client. By using an SSH tunneled port as a virtual physical medium, PPP can establish a point-to-point network connection. The result is an encrypted, authenticated, and validated PPP tunnel between hosts. This tunnel can forward packets from one network to the other—even over the Internet—with little threat from eavesdropping, replay, or insertion attacks.

An example PPP over SSH connection can be established using a single PPP command (Listing 9.1). This command establishes a SSH connection between the local system and remote `host`, and then initiates a PPP connection. The PPP connection is created with the local IP address `10.100.100.100` and remote address `10.200.200.200`. No PPP authentication (`noauth`) is provided because the SSH connection already authenticates the user.

LISTING 9.1 Example PPP over SSH Connection

```
% sudo pppd nodetach noauth passive pty \
  "ssh root@remotehost pppd nodetach notty noauth" \
  ipparam vpn 10.100.100.100:10.200.200.200
Using interface ppp0
Connect: ppp0 <--> /dev/pts/5
Deflate (15) compression enabled
local IP address 10.100.100.100
remote IP address 10.200.200.200
```



For the sample in Listing 9.1, the remote user must be able to run `pppd noauth`. This usually requires root access. When using this command without SSH certificates, there are two password prompts. The first is for the local `sudo` command, and the second is for the remote `ssh` login. Because `pppd` is executing the `ssh` command, no other user-level prompting is permitted.

After establishing the PPP over SSH connection, one or both sides of the network may add a network routing entry such as `route add -net 10.200.200.200 netmask 255.0.0.0` or `route add default gw 10.200.200.200`. The former specifies routing all `10.x.x.x` subnet traffic over the VPN tunnel. The latter sets the VPN as the default gateway; this is a desirable option for tunneling all network traffic.

2. PPPoE

PPP may be tunneled over another data link layer. *PPP over Ethernet* (PPPoE), defined in RFC2516, extends PPP to tunnel over IEEE 802.3 networks. Many cable and DSL connections use PPPoE because PPP provides a mechanism for negotiating authentication, IP address, connection options, and connection management. The PPPoE server is called the *access concentrator* (AC) because it provides access for all PPPoE clients.

PPPoE does have a few specified limitations:

MTU: The largest MTU value can be no greater than 1,492 octets. The MTU for Ethernet is defined as 1,500 octets, but PPPoE uses 6 octets and PPP uses 2.

Broadcast: PPPoE uses a broadcast address to identify the PPPoE server. Normally the PPPoE server responds, but a hostile node may also reply.

AC DoS: The AC can undergo a DoS if there are too many requests for connection allocations. To mitigate this risk, the AC may use a cookie to tag requests. Multiple requests from the same hardware address should receive the same client address.

3. CSLIP and CPPP

When tunneling PPP or SLIP over another protocol, the overhead from repeated packet headers can significantly impact performance. For example, TCP surrounds data with at least 20 bytes of header. IP adds another 20 bytes. So `IP(TCP(data))` increases the transmitted size by at least 40 bytes more than the data size. When tunneling PPP over a SSH connection, the resulting stack `MAC(IP(TCP(SSH(PPP(IP(TCP(data)))))))` contains at least 80 bytes of TCP/IP header plus the PPP and SSH headers. The resulting overhead leads to significant speed impacts:

Transmit Size: When tunneling, there are 80 additional bytes of data per packet. More headers mean more data to transmit and slower throughput.

Message Frames: Many data link protocols have well-defined MTUs. Ethernet, for example, has an MTU of 1,500 bytes. Without tunneling, TCP/IP and MAC headers are at least 54 bytes, or about 4 percent of the MTU. With tunneling, the TCP/IP and PPP headers add an additional 3 percent plus the overhead from SSH.

Stack Processing: When transmitting and receiving, twice as many layers must process each byte of data. This leads to computational overhead.

Although the increase in transmitted data and message frames can be negligible over a high-speed network, these can lead to significant performance impacts over low-bandwidth connections, such as dialup lines. For modem connections, a 3 percent speed loss is noticeable. The stack-processing overhead may not be visible on a fast home computer or workstation, but the overhead can be prohibitive for hardware systems, mission-critical systems, or real-time systems.



NOTE

Generally hardware-based network devices can process packets faster than software, but they use slower processors and do not analyze all data within the packet. Tunneling through a hardware device is relatively quick; however, tunneling to a hardware device requires processing the entire packet. The limited processing capabilities can significantly impact the capabilities to handle tunneled connections.

To address the increased data size, RFC1144 defines a method to compress TCP/IP headers. The technique includes removing unnecessary fields and applying Lempel-Ziv compression. For example, every TCP and IP header includes a checksum. Because the inner data is not being transmitted immediately, however, there is no need to include the inner checksums. This results in a 4-byte reduction. Together with Lempel-Ziv compression, the initial 40-byte header can be reduced to approximately 16 bytes.

Using this compression system with PPP and SLIP yields the *CPPP* and *CSLIP* protocols. Although there is still a size increase from the tunnel, the overhead is not as dramatic with CPPP/CSLIP as it is with PPP and SLIP.

Although CPPP and CSLIP compress the tunneled TCP/IP headers, they do not attempt to compress internal data. Some intermediate protocols support data compression. For example, SSH supports compressing data. Although this may not be desirable for high-speed connections, the SSH compression can be enabled for slower networks or tunneled connections.

Unfortunately, there are few options for resolving the stack processing issue. Tunneling implies a duplication of OSI layer processing. To resolve this, a non-

tunneling system should be used. For example, IPsec provides VPN support without stack repetition.

3. COMMON RISKS

The largest risks from PPP and SLIP concern authentication, bidirectional communication, and user education. Although eavesdropping, replay, and insertion attacks are possible, these attacks require access to the physical layer. Because a point-to-point network only contains two nodes on the network, physical layer threats against the data link layer are usually not a significant consideration.

1. Authentication

SLIP provides no authentication mechanism. Instead, an authentication stack is usually used prior to enabling SLIP. In contrast, PPP supports both PAP and CHAP for authentication [RFC1334]. PAP uses a simple credential system containing a username and password; the username and password are sent to the server unencrypted and then validated against the known credentials.

In contrast to PAP, CHAP uses a more complicated system based on a key exchange and a shared secret key (Figure 9.1). Rather than transmitting the credentials directly, CHAP transmits a username from the client (*peer*) to the server (*authenticator*). The server replies with an 8-bit ID and a variable-length random number. The ID is used to match challenges with responses—it maintains the CHAP session. The client returns an MD5 hash of the ID, shared secret (account password), and random number. The server computes its own hash and compares it with the client's hash. A match indicates the same shared secret.

Unlike PAP, CHAP authentication is relatively secure and can be used across a network that may be vulnerable to eavesdropping; however, CHAP has two limitations. First, it only authenticates the initial connection. After authentication, PPP provides no additional security—someone with eavesdropping capabilities can hijack the connection after the authentication. Second, if an eavesdropper captures two CHAP negotiations with small-length random numbers, then the hash may be vulnerable to a brute-force cracking attack.

In both PAP and CHAP, the server must contain a file with all credential information. SLIP and PPP do not support stronger authentication methods, such as those based on hardware tokens or biometrics. And, neither SLIP nor PPP encrypts the data transmission, so an eavesdropper can observe login credentials.

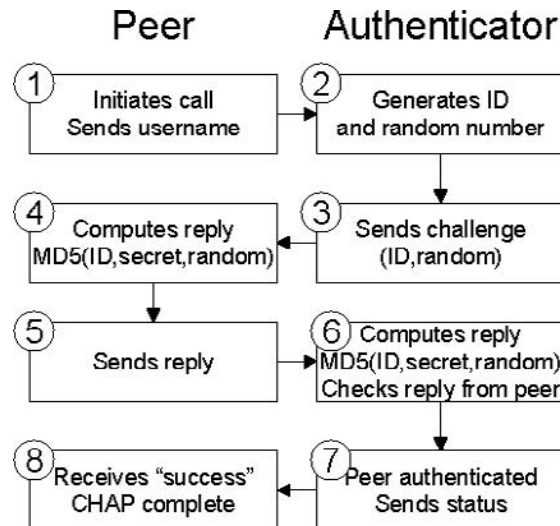


FIGURE 9.1 CHAP processing.

2. Bidirectional Communication

PPP and SLIP provide full-data link support in that the node may communicate with a remote network, and the remote network may communicate with the node. PPP and SLIP provide full bidirectional communication support. As such, any network service operating on the remote client is accessible by the entire network. Most dialup users do not use home firewalls, so any open network service may leave the system vulnerable.

Software firewalls, or home dialup firewalls such as some models of the SMC Barricade, offer approaches to mitigate the risk from open network services.

3. User Education

More important than bidirectional communication is user education. Most dialup, DSL, and cable modem users may not be consciously aware that their connections are bidirectional. Moreover, home firewalls can interfere with some online games and conferencing software such as Microsoft NetMeeting. As such, these preventative measures may be disabled and leave systems at risk.

You Can't Do That!

Most universities provide dialup support for students and faculty. One faculty member was positive that his dialup connection was not bidirectional. He

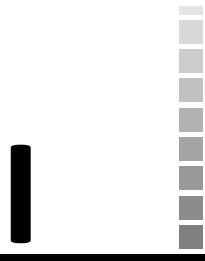
challenged members of the computer department's support staff to prove him wrong. The proof took three minutes.

First, a login monitor was enabled to identify when the faculty member was online. Then his IP address was identified. This allowed the attackers to know which of the dialup connections to attack. After being alerted to the victim's presence, the attackers simply used Telnet to connect to the address and were greeted with a login prompt. The login was guessed: root, with no password. Then the attacker saw the command prompt, showing a successful login. (The faculty member had never bothered to set the root password on his home computer.) A friendly telephone call and recital of some of his directories changed his viewpoint on dialup security.

Ironically, this faculty member worked for a computer science department. If he was not aware of the threat, how many home users are aware?

9.4 SIMILAR THREATS

The risks from point-to-point networks, such as PPP and SLIP, extend to other point-to-point systems. High-speed dialup connections such as DSL and ATM use point-to-point physical connections—DSL uses PPPoE [RFC2516], and ATM uses PPPoA [RFC2364]. For these configurations, the data link layer provides a virtually transparent connection. An attacker with physical layer access is not impeded by any data link security.



MAC and ARP

In This Chapter

- Data Link Sublayers
- ARP and RARP
- Network Routing
- Physical Layer Risks

On multiple node networks, systems use network addressing to direct data. This addressing happens in the data link layer (OSI layer 2). Any of three addressing methods may be supplied:

Unicast: Each data link message frame is intended for a specific node. All other nodes (nonrecipients) should ignore the message frame.

Multicast: The message frame is intended for a specific set of nodes but not every node.

Broadcast: The message frame is intended for all nodes on the physical network.

One of the most common addressing methods is defined by the IEEE 802 standard. The MAC defines a unique network address to each node on the network and methods for interfacing with the data link layer.

1. DATA LINK SUBLAYERS

IEEE 802 splits the OSI data link layer into two sublayers: LLC and MAC. The LLC comprises the upper half of the data link layer, and the MAC is the lower half.

1. LLC

The *logical link control* (LLC) is defined by the IEEE 802.2 standard and resides in the upper portion of the data link layer. It provides four functions: link management, SAP management, connection management, and sequence management. The LLC provides a consistent interface with the upper OSI layers regardless of the physical layer.

1. Link Management

The LLC provides the control mechanisms for protocols that require flow control. This is commonly seen in Token Ring networks but not common for Ethernet (10Base-2, 100Base-T, etc.). Many quality-of-service protocols, such as those used by ATM and wireless, use LLC link management to compute the quality.

In addition to quality of service, data link standards such as *Spanning Tree* (IEEE 802.1) use the LLC. Spanning Tree is used by bridges and ensures that routing loops do not form. Each bridge sends out a Spanning Tree message frame with an identifier. If a message frame is received with the same identifier, then the route is assumed to have a loop. Spanning Tree breaks loops by not sending message frames to hardware address that form loops. Although Spanning Tree breaks bridging loops, it also opens the network to a simple DoS attack. Any attacker on the network can reply to the Spanning Tree packet causing the bridge to assume that every path is a loop. This results in a bridge that will not bridge traffic.

2. SAP Management

Service Access Points (SAPs) are ports to network layer (OSI layer 3) protocols. Each network layer protocol uses a different 16-bit identifier to identify the network protocol. For example, IP is 0x0800, IPv6 is 0x86DD, and AppleTalk is 0x809B. The two SAP bytes are divided into a destination (DSAP) and source (SSAP), 1 byte each.

The *Sub Network Access Protocol* (SNAP) was added to IEEE 802.2 to extend the number of DSAP/SSAP values. Using SNAP, the LLC is not limited to 1 byte for

each DSAP and SSAP. Instead, the SNAP defines a 3-byte Organizationally Unique ID (OUI) and a 2-byte protocol type.

3. Connection Management

The LLC determines whether the data link layer should use a connection-oriented or connection-less communication flow. The type of connection is independent of higher layer protocols. For example, UDP is an OSI layer 4 connection-less protocol, and TCP is an OSI layer 4 connection-oriented protocol. However, both TCP and UDP may use connection-less (or connection-oriented) data link protocols.

4. Sequence Management

For connection-oriented data link traffic and connection-less traffic that requires a confirmation, the LLC includes a sequence number and uses a sliding window system. The sequence numbers ensure that each message frame is received in the correct order. The sliding window reduces communication overhead by only requiring occasional acknowledgements rather than an acknowledgement after every transmission.

each DSAP and SSAP. Instead, the SNAP defines a 3-byte Organizationally Unique ID (OUI) and a 2-byte protocol type.

3. Connection Management

The LLC determines whether the data link layer should use a connection-oriented or connection-less communication flow. The type of connection is independent of higher layer protocols. For example, UDP is an OSI layer 4 connection-less protocol, and TCP is an OSI layer 4 connection-oriented protocol. However, both TCP and UDP may use connection-less (or connection-oriented) data link protocols.

4. Sequence Management

For connection-oriented data link traffic and connection-less traffic that requires a confirmation, the LLC includes a sequence number and uses a sliding window system. The sequence numbers ensure that each message frame is received in the correct order. The sliding window reduces communication overhead by only requiring occasional acknowledgements rather than an acknowledgement after every transmission.

Big MAC Attack

The term “MAC” may be used many different ways. Although they have the same acronym, *media* access control is not the same as *medium* access control. The medium access control refers to transmission and collision detection, such as CSMA/CD. In contrast, the media access control is commonly associated with the entire OSI layer 2 protocol (LLC + medium access control). Additionally, MAC may refer to the hardware address: MAC address. This ambiguity can result in communication problems between people discussing the network. Is a “MAC attack” related to the hardware address, collision detection, or overall data link layer?

10.1.2.2 Network Addressing

Every node on the network requires a unique address. This hardware address allows unicast and multicast packets to be processed by the recipient node. As each packet is received, the destination hardware address is compared against the local hardware address. If the addresses match, then the packet is processed. Mismatches indicate a packet that is not intended for the node, so the node quietly drops the packet.

Ethernet network cards use a preset 6-byte code to specify the MAC address. These are usually written in a colon-delimited format: `00:11:22:33:44:55`. The first three octets are the *Organizationally Unique Identifier* (OUI). These usually determine the manufacturer. For example, `00:CO:4F:xx:xx:xx` is a Dell Computer network card, and `00:08:83:xx:xx:xx` denotes a Hewlett-Packard network card. In addition, the IEEE registration authority supplies an OUI subset called the *Individual Address Block* (IAB). The IAB defines the first 24 bits, only allowing 12 bits for unique identification. For example, Microsoft is allocated the MAC addresses range `00:30:00:00:30:00` to `00:30:00:00:3F:FF`.

The first three octets define the OUI. The remaining three octets are vendor specific—they may indicate the particular make and model of the network card or a third-party vendor. Some portion of the last three octets is pseudo-unique; two identical network cards purchased at the same time will likely have different 6-octet sequences. However, there are a limited number of unique octets. For example, a manufacturer may use two octets as a unique identifier (65,536 combinations) but mass-produce 200,000 network cards. To resolve potential address duplications, most network cards permit the network driver to override the preset MAC address. As such, any user can change the MAC address to any value.

Because an attacker can change his MAC address, authentication based on MAC addresses is weak at best. An attacker can readily assume any other MAC address. Similarly, an attacker may change his MAC address for the duration of an attack, which makes it difficult to track down the actual node.

Attackers may target the MAC address through hardware profiling reconnaissance, impersonation, and load attacks.

10.1.2.3 Acquiring Hardware Addresses

The LLC ensures that multinode and point-to-point network interfaces use the same interface. Each physical layer interface is associated with a unique identifier (name) and MAC address. Additional information, such as a network layer IP address, may also be associated with an interface. Although hardware addresses are only needed for multinode networks, point-to-point networks are also assigned hardware addresses by the LLC. For example, the hardware address for the loop-back interface (`lo`, `localhost`, etc.) is usually assigned the hardware address `00:00:00:00:00:00`.



The `getmac.c` program on the CD-ROM displays all network interfaces, hardware addresses, and associated IP addresses. The main component uses the `ioctl()` function to list information associated with each interface (Listing 10.1). An example output from `getmac` (Listing 10.2) shows both multinode and point-to-point network interfaces.

LISTING 10.1 The `ListInterface` Function from the `getmac.c` Program

```

/*****
ListInterface(): A computer may have multiple network cards.
List each interface, hardware address, and IP address.
*****/
void ListInterface ()
{
    struct ifreq Interface;
    struct ifreq *InterfacePointer; /* pointer to an interface */
    struct ifconf InterfaceConfig;
    char Buffer[0x8000]; /* configuration data (make it big) */
    int i;
    int Socket;

    /* Prepare a socket */
    /*** Any socket will work, don't require root ***/
    Socket = socket(AF_INET,SOCK_STREAM,0);
    if (Socket < 0)
    {
        printf("ERROR: Failed to open socket.\n");
        exit(-1);
    }

    /** SIOCGIFCONF returns list of interfaces **/
    InterfaceConfig.ifc_len = sizeof(Buffer);
    InterfaceConfig.ifc_buf = Buffer;

```



```

if (ioctl(Socket,SIOCGIFCONF,&InterfaceConfig) < 0)
{
printf("Error obtaining %s IP address\n",Interface.ifr_name);
exit(-1);
}
/** list every interface **/
for(i=0; i*sizeof(struct ifreq) < InterfaceConfig.ifc_len; i++)
{
InterfacePointer=&(InterfaceConfig.ifc_req[i]);

/* Load the interface mac address */
strcpy(Interface.ifr_name,InterfacePointer->ifr_name);

ioctl(Socket,SIOCGIFHWADDR,&Interface);

/* Display the results */
/** The "unsigned int" and "0xff" are just so negative
characters appear as positive values between
0 and 255. **/
printf("Interface: %-6s ",InterfacePointer->ifr_name);
printf("MAC: %02x:%02x:%02x:%02x:%02x:%02x ",
(unsigned)(Interface.ifr_hwaddr.sa_data[0])&0xff,
(unsigned)(Interface.ifr_hwaddr.sa_data[1])&0xff,
(unsigned)(Interface.ifr_hwaddr.sa_data[2])&0xff,
(unsigned)(Interface.ifr_hwaddr.sa_data[3])&0xff,
(unsigned)(Interface.ifr_hwaddr.sa_data[4])&0xff,
(unsigned)(Interface.ifr_hwaddr.sa_data[5])&0xff
);
printf("IP: %u.%u.%u.%u\n",
(unsigned)(InterfacePointer->ifr_addr.sa_data[2])&0xff,
(unsigned)(InterfacePointer->ifr_addr.sa_data[3])&0xff,
(unsigned)(InterfacePointer->ifr_addr.sa_data[4])&0xff,
(unsigned)(InterfacePointer->ifr_addr.sa_data[5])&0xff
);
}
close(Socket);
} /* ListInterface() */

```

LISTING 10.2 Sample Output from `getmac.c`

```

Interface: lo      MAC: 00:00:00:00:00:00  IP: 127.0.0.1
Interface: eth0    MAC: 00:60:08:ca:2e:2c  IP: 10.1.1.247
Interface: eth1    MAC: 00:01:03:69:4d:77  IP: 10.2.21.38

```

10.1.3 MAC Vulnerabilities

Although the MAC provides the means to communicate information between hosts on a network, it also introduces potential attack vectors. Attackers may use

MAC information for reconnaissance, impersonation, or for directed load-based attacks.

1. Hardware Profiling Attacks

The first step in attacking a system is reconnaissance. An attacker blindly attempting to attack an unknown system will rarely be successful. The OUI provides hardware and operating system information to an attacker. For example, if an attacker sees a source MAC address with the OUI `00:0D:93`, then the attacker knows the manufacturer is Apple Computers. The operating system associated with the MAC address is likely a version of MacOS and not Windows, Solaris, or other operating system. Similarly, `00:20:F2` indicates Sun Microsystems—the host is likely running a version of SunOS or Solaris.

Attacks against the data link layer are rare and require direct physical layer access. Although hardware profiling is a viable information reconnaissance technique, the attacker is likely to already know the type of system. In environments where it is desirable to obscure the hardware profile, most network drivers permit changing the MAC address. The method to change the MAC address differs by operating system and may not be supported by some drivers:

Windows 2000: The advanced configuration for most network adapters permits setting the hardware address (Figure 10.1).

RedHat Linux: The root user can use the `ifconfig` command to change the adapter's hardware address. This is a three-step process. First, the interface must be taken down, `ifconfig eth0 down`. Then the hardware address is changed, `ifconfig eth0 hw ether 00:11:22:33:44:55`. Finally, the interface is brought back up: `ifconfig eth0 up`.

Mac OS X: The Macintosh OS X operating system uses a variation of the `ifconfig` command: `ifconfig en0 ether 00:11:22:33:44:55`. Unlike Linux, the interface does not need to be down to change the address.



Some network adapters, such as some Macintosh Powerbook's wireless network cards, do not support changes to the MAC address. In some situations, this is a limitation of the hardware; in other cases, this is a driver limitation.

2. Impersonation Attacks

Users with administration privileges can change their MAC address. An attacker may intentionally change the address to duplicate another node on the network. If both systems are active on the network, it is likely that both systems will interfere with each other. The attack effectively becomes a DoS.

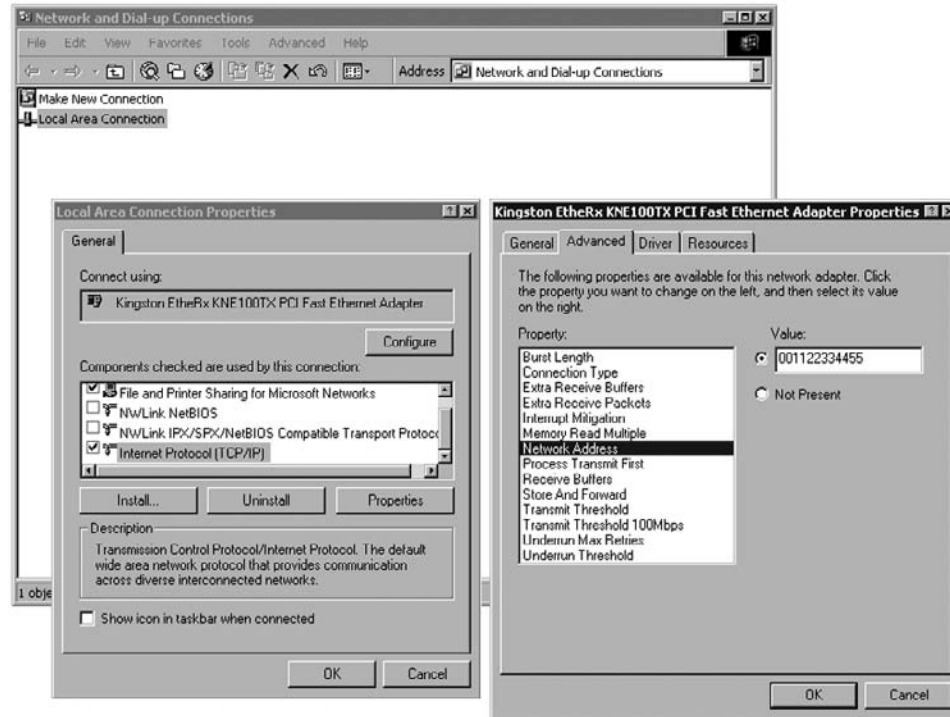


FIGURE 10.1 Changing the hardware address under Windows2000.

In some situations, two nodes can coexist on the same network with the same hardware address. If both systems use different network layer (OSI layer 3) services or different data link SAPs, then they may operate without interference. This type of impersonation may bypass some IDSs, particularly when specific nodes are filtered based on hardware address.

Finally, the impersonation attack may be used to bypass MAC filtering. Systems that validate access based on MAC addresses can be bypassed by impersonating an appropriate hardware address.

10.1.3.3 Load Attacks

As well as being part of the OUI, the first octet contains addressing flags. The least significant bit of the first octet indicates a multicast packet. Every odd value for the first octet is a multicast. The value FF indicates a broadcast packet. Attackers can quickly initiate a load attack because they can change their MAC address to be a broadcast or multicast address. Although this will have no impact on the attacker's system, all other nodes on the network will attempt to process every packet intended for the attacker's node.

An example of this style of a load attack assigns the hardware address FF:FF:00:00:00:00 to the attacker's node. All other nodes on the network will view packets directed to the attacker's node as a broadcast packet.



Although most versions of Linux, BSD, MacOS, and Windows support setting a broadcast address as the MAC address, this is operating system- and driver-specific. Under Windows XP, one test system did not support changing the MAC address, and another disabled the network card when a broadcast address was entered, removing it from the list of available network devices. The card could only be re-enabled through the hardware device manager.

10.2 ARP AND RARP

The *Address Resolution Protocol* (ARP) is an example of a service access protocol. ARP, defined in RFC826, assists the network layer (OSI layer 3) by converting IP addresses to hardware addresses. The *Reverse Address Resolution Protocol* (RARP) converts a hardware address to an IP address.

ARP packets include a function code such as ARP request, ARP reply, RARP request, and RARP reply. ARP and RARP requests are broadcast packets that include the query information. For example, an ARP request contains an IP address. The expectation is that all nodes will receive the broadcast, but only one node will claim the IP address. The node that identifies the IP address will transmit an ARP reply (unicast) back to the querying host. The query results are stored in an ARP table, which is viewable with the `arp` command (Listing 10.3). The ARP table lists all nodes on the local network that have responded to ARP requests.

LISTING 10.3 Sample Output from the `arp` Command

```
$ /sbin/arp
Address      HWtype  HWaddress      Flags Mask    Iface
10.1.3.1     ether   00:50:18:03:C0:20  C             eth0
10.1.3.2     ether   00:C0:F0:40:42:2F  C             eth0
10.1.3.3     ether   00:11:D8:AB:22:F4  C             eth0
```

ARP traffic is not passed beyond the local network. Bridges, routers, and gateways do not relay ARP packets. Nodes that are not on the local network will never be listed in the local ARP table.

10.2.1 ARP Poisoning

Mapping IP addresses to hardware addresses (and vice versa) can add a delay to network connections. *ARP tables* contain a temporary cache of recently seen MAC

addresses. Therefore, ARP packets are only required when a new IP address (or MAC address) lookup needs to be performed. Unfortunately, these cached entries open the system to ARP poisoning, where an invalid or intentionally inaccurate entry is used to populate the table.



ARP replies are not authenticated, but ARP reply values are placed into the ARP table. Moreover, if a later ARP reply is received, it can overwrite existing ARP table entries. The `arp_stuff.c` program on the CR-ROM and in Listing 10.4 is one example of a tool for poisoning ARP tables. This program generates an ARP reply and sends a fake MAC address and IP address pair to a target system. As the target processes the packet, it adds the ARP reply's information to the system's ARP table. In this attack, no ARP request is required.

LISTING 10.4 Code Fraction from `arp_stuff.c` for Poisoning an ARP Table

```
void LoadInterface (int Socket, char *InterfaceName,
                   struct sockaddr_ll *SocketData)
{
    struct ifreq Interface;
    memset(SocketData,0,sizeof(struct sockaddr_ll));
    SocketData->sll_protocol = htons(ETHERTYPE_ARP);
    SocketData->sll_halen    = ETH_ALEN;
    strcpy(Interface.ifr_name,InterfaceName);
    ioctl(Socket,SIOCGIFINDEX,&Interface);
    SocketData->sll_ifindex = Interface.ifr_ifindex;
    ioctl(Socket,SIOCGIFHWADDR,&Interface);
    memcpy(SocketData->sll_addr,Interface.ifr_hwaddr.sa_data,
           ETH_ALEN);
} /* LoadInterface() */

/*****
int    main    (int argc, char *argv[])
{
    packet Packet;
    struct sockaddr_ll SocketData;
    in_addr_t IPaddress;
    int Socket;
    int rc;

    /* load Ethernet header */
    memset(&Packet,0,sizeof(packet));
    Text2Bin(argv[3],Packet.ether_header.ether_shost,ETH_ALEN);
    Text2Bin(argv[5],Packet.ether_header.ether_dhost,ETH_ALEN);
    Packet.ether_header.ether_type = htons(ETHERTYPE_ARP);

    /* load ARP header */
    Packet.arp_header.ar_hrd = htons(ARPHRD_ETHER);
    Packet.arp_header.ar_pro = htons(ETH_P_IP);

```

```

Packet.arp_header.ar_hln = ETH_ALEN;
Packet.arp_header.ar_pln = 4;
Packet.arp_header.ar_op = htons(ARPOP_REPLY);

/* Load ARP data */
/** store the fake source ARP and IP address **/
Hex2Bin(argv[3],Packet.arp_sha,6);
IPaddress = ntohl(inet_addr(argv[2]));
Packet.arp_sip[0] = (IPaddress & 0xff000000) >> 24;
Packet.arp_sip[1] = (IPaddress & 0x00ff0000) >> 16;
Packet.arp_sip[2] = (IPaddress & 0x0000ff00) >> 8;
Packet.arp_sip[3] = (IPaddress & 0x000000ff) >> 0;
/** set the real target ARP and IP address **/
Hex2Bin(argv[5],Packet.arp_tha,6);
IPaddress = ntohl(inet_addr(argv[4]));
Packet.arp_tip[0] = (IPaddress & 0xff000000) >> 24;
Packet.arp_tip[1] = (IPaddress & 0x00ff0000) >> 16;
Packet.arp_tip[2] = (IPaddress & 0x0000ff00) >> 8;
Packet.arp_tip[3] = (IPaddress & 0x000000ff) >> 0;

Socket = socket(PF_PACKET,SOCK_RAW,htons(ETHERTYPE_ARP));
LoadInterface(Socket,argv[1],&SocketData);

/* Send data. sendto() does not require a connection */
rc = sendto(Socket,&Packet,sizeof(Packet),0,
            (struct sockaddr *)&SocketData,sizeof(SocketData));
close(Socket);
return(0);
} /* main() */

```

2. ARP Poisoning Impact

The result from an ARP poisoning can range from a resource attack to a DoS or MitM attack.

1. Resource Attack

Some systems have a limited number of ARP entries that can be cached. By sending a large number of false ARP entries, the ARP table can fill up. When the table fills, there are only two options: ignore additional ARP entries or throw out the oldest entries. If the system ignores additional entries, then no new nodes on the local network can be contacted. If the node throws out the oldest ARP entry, then the result is a much slower network performance due to constant ARP queries for each packet that the system wants to transmit.

2. Denial of Service (DoS)

Newer ARP replies overwrite older ARP replies in the ARP table. If an entry in the ARP table is overwritten with a bad MAC address, then future connections to the overwritten node's IP address will fail. For example, if the ARP table entry `00:11:22:33:44:55 = 10.1.2.3` is overwritten by the poisoned ARP reply `00:11:11:11:11:11 = 10.1.2.3`, then all subsequent traffic to `10.1.2.3` will fail because it will be sent to the wrong MAC address.

3. Man-in-the-Middle (MitM)

The MitM attack routes all traffic through a hostile node. Similar to the DoS attack, the ARP table entry is overwritten with a different machine's MAC address. In this situation, the new node will receive all traffic intended for the old node. By poisoning both nodes (Figure 10.2), the hostile node can establish a successful MitM.

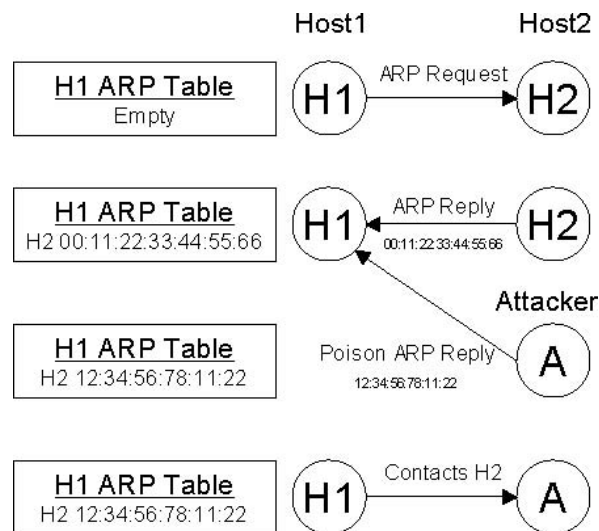


FIGURE 10.2 ARP poisoning as a MitM attack.

10.2.3 Mitigating ARP Poisoning

Although very problematic, the scope of an ARP poisoning attack is limited to the local network; ARP packets do not travel beyond bridges, routers, or gateways. Although the range of the attack is limited, it can still impact network security. The few options for limiting the impact of ARP attacks include hard-coding ARP tables, expiring ARP entries, filtering ARP replies, and locking ARP tables.

1. Hard-Coding ARP Tables

Normally ARP tables are populated dynamically, as each ARP reply is received; however, ARP tables can be populated statically using the operating system's `arp` command. This command statically sets a MAC address and IP address pair into the ARP table and prevents future modification.

2. Expiring ARP Entries

Cached entries in an ARP table may timeout. Expired entries, where the same MAC and IP address combination are not observed, can be removed from the ARP table. This mitigation option limits the impact from resource attacks.

3. Filtering ARP Replies

Not every ARP reply must be inserted into the ARP table. Linux and BSD limit cache entries to ARP requests that were sent by the local host. This prevents unsolicited ARP replies from entering the ARP table. In contrast, most Windows operating systems accept unsolicited ARP replies and insert them into the ARP table. Although ARP reply filtering does prevent unsolicited entries, it does not prevent new replies from overwriting existing entries.

4. Locking ARP Tables

ARP tables can be temporarily locked. In this situation, an established connection (such as an IP connection) locks the ARP table entry for a short duration. During this time, new ARP replies cannot overwrite the locked table entry—ensuring that an established connection cannot be changed while it is established and mitigating MitM attacks. A MitM attack can only be initiated during the short duration between the system's ARP request and initiating network connection. Systems that support ARP table locking are uncommon.

3. NETWORK ROUTING

Network devices such as switches and bridges commonly use ARP packets. In particular, these devices focus on ARP replies. Unfortunately, these systems are susceptible to ARP attacks such as switch poisoning and switch flooding.

1. Switches

Network hubs and switches link two or more network segments that use the same physical medium. Data transmitted on one network segment is retransmitted through the hub to all other network segments. *Hubs* are considered “**dumb**” network devices because they simply receive and retransmit data.

In contrast to hubs, *switches* are “*smart*” network devices. Switches maintain their own internal ARP table and associate each ARP reply with a physical port on the switch. As each packet is received, the switch identifies the destination MAC address and routes the packet to the specific port rather than sending it to every port. Switches only retransmit the packet to every port when the destination MAC address is unknown.

A switch can identify MAC addresses in three ways:

ARP Requests: Many switches know how to generate ARP requests. When an unknown destination is encountered, the switch generates an ARP request across every port and waits for a reply.

ARP Replies: Any ARP reply, even one not generated by the switch, is used to populate and update the switch’s ARP table.

Frame Headers: Each message frame contains a source MAC address. The switch can use this to associate the source MAC address with a specific port.

2. Bridges

Bridges link two networks that use the same data link protocol. For example, a bridge may span networks that use coaxial 10Base-2 and twisted pair 100Base-T. Both of these physical layer media use the same LLC, MAC, and CSMA/CD protocols. As with switches, bridges may associate MAC addresses with specific interfaces. Any message frame, where the source and destination addresses are on the same network (and same bridge interface), are not retransmitted through the bridge. Similarly, bridges with multiple interfaces may route message frames based on an internal ARP table.

3. Switch Attacks

As with any network node, switches and bridges are vulnerable to poisoning and flooding attacks.

1. Switch Poisoning Attacks

Switches maintain an ARP table to route traffic through the switch to specific physical network ports. An ARP poisoning attack can corrupt the ARP table. The poisoning ARP reply can associate another node’s MAC address with a different port. This effectively cuts off the victim node from the network and sends all traffic intended for the victim through the attacker’s port. Because switch poisoning redirects network traffic, this attack allows connection hijacking.

10.3.3.2 Switch Flooding Attacks

Nodes operating in promiscuous mode cannot receive any network traffic beyond the immediate local network. Switches and bridges normally ensure that nodes only receive traffic intended for the local physical network. An attacker, using ARP poisoning, can flood a switch's ARP table. Because switches cannot afford to drop packets, most switches regress to a hub state when the switch's ARP table fills. In this state, all nodes receive all traffic; a node operating in promiscuous mode can begin receiving all network traffic that passes through the switch.

Most bridges and high-end switches support static ARP entries. Using static ARP tables in these network devices can mitigate the impact from switch flooding and poisoning.

Although flooding attacks can enable network monitoring by a promiscuous node, segmenting the network with multiple switches and bridges still mitigates the impact from this attack. For example, an onion network topology can have any single bridge/switch overwhelmed by a flooding attack, but not everyone will be compromised. As each device regresses to a hub state, the overall network volume increases. Eventually the network will become unusable due to too much network traffic; the onion topology will limit the DoS to a few rings. In addition, IDS and IPS monitors will likely detect a sudden increase in network traffic (and ARP replies), indicating a network attack.

10.4 PHYSICAL LAYER RISKS

Although the IEEE 802 data link layer is closely associated with the physical layer, it does operate independently. As such, all risks associated with the physical layer also impact the data link layer. For example, nothing within this data link layer can identify a splicing or insertion attack against the physical layer. Because the MAC frame information does not contain timestamps or cryptography, physical layer replay and insertion attacks are successful. The assumption is that higher-layer protocols will detect and prevent these types of attacks.



Network Layer

In This Chapter

- Routing Risks
- Addressing
- Fragmentation

11.1 ROUTING

Network *routing* permits two distinct data link layers to communicate. Without network routing, the data link layer would require all nodes to communicate directly with all other nodes on the network. Large networks would become bottle-

2. **ROUTING RISKS**

In the OSI model, network routers are the only option for communicating with distant networks. Direct attacks against the router will, at minimum, interfere with the capability to communicate with other networks. Even when the physical and data link layers remain intact, the network layer can be disabled, which prevents network routing.

Router-based attacks appear in many forms: direct attacks, table poisoning, table flooding, metric attacks, and router looping attacks.

1. **Direct Router Attacks**

A *direct router attack* takes the form of a DoS or system compromise. A DoS prevents the router from performing the basic routing function, which effectively disables network connectivity. Historically, these attacks have been load based: if

the network volume on a particular interface is too high, then the router will be unable to manage the traffic, including traffic from other network interfaces.



Although most PCs have very fast processors, most hardware routers use slower CPUs. A 2 GHz PC running Linux may make a dependable router, but a Cisco PIX is a commonly used commercial router that uses a 60 MHz processor. Hardware solutions are typically faster than software solutions, but slower processors can be overwhelmed with less effort.

Although rare, router compromises can be devastating. The router can be reconfigured to forward traffic to a different host, block traffic from specific hosts, or arbitrarily allocate new, covert subnets. Because routers span multiple subnets, a compromise router impacts the integrity and privacy of every network connected to it.

2. Router Table Poisoning

As with the data link layer's ARP table, the network layer's routing table is vulnerable to poisoning attacks. Few network protocols authenticate the network's traffic. Forged or compromised network traffic can overwrite, insert, or remove routing table entries. The result is not very different from a compromised router: the poisoned table can be reconfigured to forward traffic to a different host, block traffic from specific hosts, or arbitrarily allocate new, covert subnets.

Network layer protocols support dynamic routing tables, where the table is automatically generated and updated based on the observed network traffic. These protocols are more vulnerable because (1) a new node may generate poisoning data, and (2) few dynamic nodes are verifiable. Critical routers should use static routing tables to deter poisoning.

3. Router Table Flooding

Routers generally do not have large hard drives or RAM for storing routing tables. The routing table size is usually limited. Devices that do not use static routes must manage route expiration and table filling. An attacker can generate fake data that the router will use to populate the routing table. When routing tables fill, the router has three options: ignore, discard oldest, or discard worst routes:

Ignore New Routes: Routers may choose to ignore new routes. An attacker will be unable to dislodge established table entries but can prevent new, valid entries from being inserted into the table.

Discard Oldest Routes: As with ARP tables, routing tables may discard routes that are not used. A large flooding attack can dislodge established table entries.

Discard Worst Routes: The router may consider routing table metrics. Less desirable paths can be replaced with more desirable routes. Although default or highly desirable routes are unlikely to be replaced, new routes that appear desirable may dislodge table entries for alternate paths.

For a router table flooding attack to be successful, the attacker must know how the router responds to a full routing table. For example, an attack that generates many fake paths that appear optimal is effective if the router discards the worst routes, but the attack will have little impact if the router ignores all new paths.

Most dynamic routing tables also support static entries. Critical routes should be configured as static entries.

4. Routing Metric Attacks

A *routing metric attack* poisons the dynamic metrics within a routing table. This attack can make a preferred path appear less desirable. For example, most data link and network layer protocols support quality-of-service packets. These are used for flow control as well as determining connectivity quality. An attacker who forges quality-of-service packets can modify dynamic metrics. The result may be slower throughput, longer paths, or more expensive networking costs. In the worst case, the router may disable a desirable path.

Just as static paths mitigate flooding attacks, metric attacks are defeated by static metrics. Fixed paths should use static metrics. In networks where dynamic metrics are critical, refresh rates should be adjusted so the router rechecks desirable routes often.

5. Router Looping Attacks

Many network protocols attempt to detect and prevent network *loops*, where data forwarded through one interface is routed to a different interface on the same router. Network loops cause excessive bandwidth consumption and can result in feedback that consumes all available network bandwidth. Whereas some network layer protocols provide means to expire undelivered packets, many protocols exist to assist in the detection of network loops. Protocols such as the data link layer's Spanning Tree (IEEE 802.1), and the network layer's Border Gateway Protocol (BGP) [RFC1772] and Routing Information Protocol (RIP) [RFC2453] provide the means to detect network loops.

When a network router identifies a network loop, the path is removed from the routing table. A looping attack generates a false reply to a loop check, making the router falsely identify a network loop. The result is a desirable network path that is disabled. Although static values can prevent table and metric attacks, looping attacks can still disable paths.

Fortunately, looping attacks are difficult to perform. An attacker must control two systems—one on each router interface being attacked. As the loop-check query is observed by one system, a message is sent to the second system with the reply information. The second system creates the false loop reply and sends it to the router.

As stated in RFC2453, “The basic RIP protocol is not a secure protocol.” Security can be added through the use of cryptography. For example, RFC2082 adds an MD5 checksum to the RIP packet, and each router contains a shared secret, preventing an attacker from blindingly sending valid RIP replies. More complicated schemes may use asymmetrical keys or encrypted RIP packets.

Many network protocols assign a *time-to-live* (TTL) value to packets. The TTL is decremented by each network relay. If the TTL reaches zero, then the packet is assumed to be undeliverable. TTLs break network loops by preventing the packet from indefinitely relaying through a loop.

11.3 ADDRESSING

The data link layer provides address support for multinode networks. Each address simply needs to be a unique sequence. The data link layer does not use addressing for any other purpose besides identifying a unique node on the network. Data link layer addressing is analogous to referencing people by their first names. As long as there is only one person named Bart in a room, there is no confusion when someone mentions Bart’s name.

Network layer addressing not only provides a unique address but also provides a method to route information to that address. Even though there are many people named Bart, there is only one at 742 Evergreen Terrace in Springfield. The address provides routing information for Bart’s messages.

Network addresses are independent of network cards or nodes. The same address may be assigned to many different nodes. This is common for fail-over server clusters, where a duplicate node compensates for any adjacent node failures. Different NICs on the same node may also have the same address. This is common for fail-over network interfaces and dual-homed systems that contain two network interfaces on two different subnets. Similarly, a single NIC may be assigned multiple network addresses (*multihosting*). Although a one-to-one association of addresses to NICs is common, it is not a requirement.

The two approaches for providing address information are numeric and name routing.

11.3.1 Numeric Addressing

Numeric addressing uses a sequence of numbers to identify a particular node on a specific network. This is analogous to contacting someone with a telephone number. The phone number routes the call to a particular location. Telephone numbers contain country codes, regional exchanges, and unique identifiers to route calls. For networking, the numeric address includes a subnet identifier and unique identifier. Different network layer protocols use different numerical routing systems. Examples of numerical routing protocols include IP, IPv6, VINES, IPX, and X.25.

Different numeric addressing schemes permit different node depths. If all end nodes are at the same depth from the uppermost router, then the network has a *uniform depth*. X.25 and IPX are examples of uniform depth protocols. In contrast, IP and IPv6 provide *variable depth* networks, where different nodes may be much further away from central routers. In contrast, VINES provides an *ad hoc* network, where there is no central routing system.

11.3.1.1 X.25

The X.25 network layer protocol applies the concept of telephone network address routing to data connections. The X.25 standard defines functions found in the OSI physical, data link, and network layers. Each of these subfunctions are defined by subset protocols of X.25. For example, the X.25 physical layer is managed by the X.21bis modem protocol—similar to V.24 and V.28. The X.25 data link layer is implemented using the Link Access Procedure, Balanced (LAPB) protocol, and the network addressing functionality is defined by X.121 [Cisco2001].

Each X.121 network address consists of three main components. The first component is a three-digit country code, similar to a telephone country code. The second component is a one-digit regional exchange identifier for identifying a particular *packet switched network* (PSN). The final component is a *national terminal number* (NTN) that may consist of up to 10 digits. The country and PSN identifiers identify a particular subnet, and the NTN identifies the unique node of the network. An X.121 address can be up to 14 digits long.



Two additional fields preface the X.121 address header. The first specifies the source address length, and the second is the length for the destination address. Although these are critical to interpreting the headers, they are not essential for understanding the addressing scheme.

X.121 is an example of a *uniform-depth addressing* scheme. All end nodes connect to a PSN, and all PSNs connect to a set of country routers. Each country router must know how to connect to other country routers and to the local PSNs. In contrast, the PSN only needs to route data between the country router and the end nodes.

2. IPX

IPX is a network protocol commonly associated with Novell NetWare. IPX uses a simplified addressing scheme. Each node is assigned a 12-byte address. The first 4 bytes identify the network number, and the next 6 bytes provide the unique node ID. The remaining 2 bytes define a service identifier.

To route data between nodes, the data is forwarded to the network's router and the 10-byte network address is analyzed. Each router maintains a table that associates network numbers to network interfaces. The routing table is used to direct traffic to the appropriate network. A single router may connect to multiple networks or other routers, which permits large networks; however, each router requires a table for navigating the network.

In addition to the address information, each IPX address header includes 2 bytes that are unique to IPX: they define a service. IPX services are used in lieu of a control protocol and tell the IPX driver how to interpret the packet. For example, the service 0x451 defines a NetWare Core Protocol (NCP) packet, and 0x456 is a diagnostic packet.

3. VINES

Banyan *Virtual Integrated Network Service* (VINES) provides network addressing and is part of Xerox Corporation's Xerox Network Systems (XNS) protocol suite. VINES uses an addressing scheme similar to IPX but does not use a centralized router. This defines an *ad hoc* network.

Each node on a VINES network is referenced by a 48-bit numeric address. The first 32 bits identify the server associated with the network. The remaining 16 bits identify each unique node.

Whereas X.25 and IPX use predefined network addresses, VINES strictly uses dynamic addresses. Each client initially requests a network address from the network server. The server provides the node with the server's number and unique node address. Although there may be many servers on the same physical network, the node only associates with one server.

In a VINES network, any host may contain multiple network interfaces, with each interface on a different network. All dual-homed nodes may act as routers. VINES routing tables are dynamically passed between nodes. As a routing table is received, the metrics are updated and merged with other received tables. The resulting merged table determines the accessible paths.

4. IP

The *Internet Protocol* (IP) is one of the most widely used network layer protocols (see Chapter 12). This protocol, described in RFC791, uses four bytes to provide address information. The bytes are usually written in a dotted-decimal format:

A.B.C.D, for example, 10.1.3.100. Subnet identification is based on bit-masked subsets. The network mask 255.255.0.0 combines with the IP address 10.1.3.100 to define the 10.1.x.x subnet. Any IP address that begins with 10.1 is a valid member of this subnet.

Unlike IPX or X.25 networks, an IP network may be very deep—the number of routers that must be traversed to reach a node varies with each network path.

11.3.1.5 IP Network Routing

When a packet is received by a router, the router examines the subnet for the packet's destination. The destination address is compared with a routing table containing a list of known networks and netmasks. The packet is then forwarded through the appropriate network interface. If there is no known path to the destination, then the packet is forwarded to a default router.

As an example (Figure 11.1), consider a machine on the 192.168.15.x subnet that wants to route a packet to a host at 10.50.45.111. Because the destination is not on the local subnet, it is passed to a default router. The default router handles all traffic intended for other networks. In this example, the 10.x.x.x subnet is not in the local network, so the data is passed to a different router that (hopefully) knows how to route to the destination's subnet. The 10.x.x.x router knows the different fractional subnets and passes the data to the appropriate subnet (10.50.x.x). The final router directs the traffic to the specific destination node.

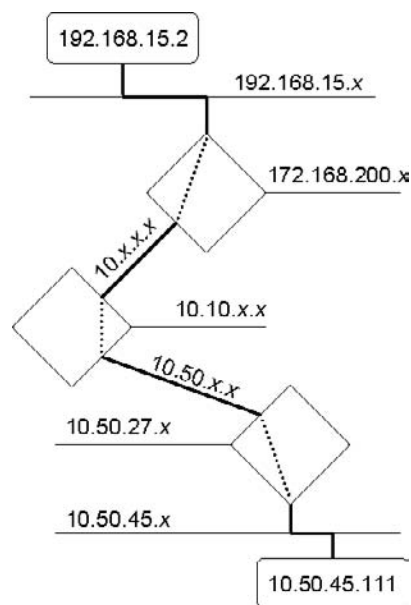


FIGURE 11.1 Sample IP network routing.

11.3.1.6 IP Multiple Routes

IP networks permit multiple routes to a particular destination. Each routing table entry is associated with a metric that indicates the cost of the network connection. The cost may indicate the number of bridges between networks, the network speed, or other factors such as service provider usage charges. If one network path is unavailable, the router simply chooses an alternate route. When no alternate routes are available, the packet's destination is considered unreachable.

The Cold War

The U.S. Department of Defense established the Advanced Research Projects Agency (ARPA) in 1958 after the Soviet Union launched the Sputnik spy satellite. ARPA was responsible for developing new military technologies. In 1972, ARPA became DARPA (*D* for Defense).

One of the ARPA projects was a network for linking computers. The ARPANET, as it was called, linked computers from military, university, and strategic corporations. ARPANET was designed as a decentralized network—a single nuclear strike would not be able to disable the entire network, and traffic would route around any disabled network segment. By the mid-1980s, the ARPANET had transformed into the Internet, connecting millions of computers worldwide. The TCP/IP network stack, as well as IP network addressing, evolved from ARPANET.

Today's Internet contains many different routes between major network segments. When a single segment becomes unavailable due to a power outage, backhoe, or network attack, the remainder of the Internet simply routes around the impacted segment.

11.3.1.7 Dynamic IP Allocation

IP addresses can be assigned statically or dynamically. *Static* addresses require an administrator to assign a specific IP address, subnet mask, and default gateway to each node.

Although IP does not natively support dynamic allocations, protocols such as OSI layer 7's Dynamic Host Configuration Protocol (DHCP) [RFC2131] and Bootstrap Protocol (BootP) [RFC1534] provide dynamic address support. For *dynamic addresses* a server exists on the network to respond to broadcast queries for an IP address. The address server associates an IP address with the node's hardware address, ensuring that the same address is not provided to two different nodes.

11.3.1.8 IPv6

The IP has undergone many revisions. The commonly used IP is also referred to as IPv4—the fourth revision. IPv4 was refined in 1981 by RFC791. Today, the largest limitation to IPv4 is address space; there are only 4,210,752 predefined subnets. To extend subnet allocation, IPv6 was introduced in 1995 [RFC1883]. IPv6 extends IP addresses from 32 bits to 128 bits. IPv6 is often called IPng, or *IP Next Generation*.

Most IPv4 subnets are allocated to companies in the United States. IPv6 is widely adopted outside of the United States, where IPv4 subnets are more difficult to acquire.

IPv6 address notation can be daunting. Addresses are usually written as 32 hex digits: `0011:2233:4455:6677:8899:AABB:CCDD:EEEE`. Leading zeros between colons can be omitted, as can repeating “:” sequences. Thus, `1080:0000:0000:0000:0000:0007:1205:7A1E` can be written as `1080:0:0:0:0:7:1205:7A1E` or `1080::7:1205:7A1E`.

For compatibility, IPv6 supports the inclusion of IPv4 addresses. The IPv4 address `10.5.193.40` is the same as the IPv6 address `0:0:0:0:FFFF:10.5.193.40` or simply `:FFFF:10.5.193.40`.

Network routing is similar to IPv4, where addresses and netmasks are associated with router interfaces. Unlike IPv4, IPv6 multicasting is a defined, routable address range [RFC2375].

11.3.2 Name-Based Addressing

Numeric addressing requires advanced knowledge of the overall topology. Numeric routing requires the router to know the location of each numeric network. For example, each router on an IP and IPv6 network must know its own subnets as well as a path that leads to other subnets. A poorly organized network may allocate too many addresses for one network segment and not enough for a different segment.



IPv6 was created because IPv4 allocated too many subnets to large corporations and not enough to smaller providers. IPv6 is more common outside of the United States because IPv4 allocations are unevenly distributed. (Because the ARPA developed the Internet, the United States took most of the addresses.) Relatively few IPv4 subnets are available outside of the United States.

Although numeric addressing is very common, name-based network addressing provides more flexibility. Rather than allocating networks and subnets based on numerical addresses, *name-based addressing* uses text strings to identify each node. Longer strings permit more complicated networks. Two examples of name-based protocols for addressing include AX.25 and NBRP.

5. FRAGMENTATION

Each network protocol has limitations concerning maximum packet size, and these may vary based on the physical network infrastructure. For example, most IP implementations use a default *maximum transmission unit* (MTU) of 1,500 bytes. However, IP over SMDS [RFC1209] uses 9,180 bytes, and IP over ATM AAL5 [RFC1626] specifies an MTU of at least 8,300 bytes. When data from a higher OSI layer is larger than the MTU, the network protocol can *fragment* the data into blocks matching the MTU size. When receiving data, the network protocol reassembles the fragments into the original (larger) data block.

For optimal performance, the network and data link fragments should be the same size (including room for headers), although this may not be feasible when changing data link protocols. A router that spans networks with differing MTUs may collect fragments, reassemble packets, and then refragment the data for better network performance.

The network layer is responsible for fragmenting data from higher OSI layers into network packets and then reassembling the fragments upon receipt. Fragments may be managed using a sequence number or offset value.

1. Fragments and Sequence Numbers

The data link layer may not preserve the order of transmission, and alternate routes may shuffle the order that the packets are received. One method to ensure proper reassembly is to label each fragment with a sequence number. As each fragment is received, they are reassembled in order.

A flag is required to determine when the last fragment is received. This may be included in the first fragment as an identifier stating the total number of fragments. Alternately, each packet may contain the total number of fragments (e.g., “this is number seven of nine”), or the last fragment may contain an end-of-fragment marker. When all fragments are received, the reassembled data block is passed up the network stack.

IPX is an example of a network protocol that uses sequence numbers for fragmentation tracking. IPX’s Sequenced Packet Exchange (SPX) protocol includes the fragment sequence number as well as the total number of fragments.

2. Fragments and Offsets

When using fragments with offsets, each packet contains an identification field, length, and memory offset value. The identification field identifies fragments from the same data block. Instead of including a sequence number, an offset is provided that identifies which part of the original sequence appears in the packet. Finally, the length of the fragment packet is known. Each packet in the sequence contains a flag

that identifies it as being part of a fragment; the last packet uses a flag identifying the end of the series (Listing 11.3). IP and IPv6 are examples of network protocols that use offsets for managing fragments.

LISTING 11.3 Sample Packet Fragmentation

```
Original data size: 3000 bytes.  
Fragmented IP packets using offsets and an MTU of 1200 bytes:  
  Packet #1: length 1200, flag=fragment, offset=0  
  Packet #2: length 1200, flag=fragment, offset=1200  
  Packet #3: length 600, flag=last_fragment, offset=2400  
Fragmented IPX packets using sequences and an MTU of 1200 bytes:  
  Packet #1: length 1200, sequence = 1 of 3  
  Packet #2: length 1200, sequence = 2 of 3  
  Packet #3: length 600, sequence = 3 of 3
```

Fragment management with offsets has one significant gain over sequence numbering: fragments can be fragmented. When using sequence numbers, fragmented data cannot be further fragmented (such as going to a network with a smaller MTU) without first reassembling the entire packet. For example, packet #2 cannot be split because packets #1 and #3 already exist. In contrast, fragments with offsets can always be split. The two packets contain smaller total lengths and different offset values.

2



Internet Protocol (IP)

In This Chapter

- IP Addressing
- ICMP
- Security Options

IP was designed for routing information between networks. Its fault-tolerant design enables data to move around disabled or unavailable network segments. The failure of a single node or subnet should not impact the entire network. Although network robustness was a consideration, general security issues were not significant design concerns. Many of IP's security risks stem from fundamental oversights made 30 years ago, whereas other exploits stem from common implementation flaws and similar coding traits.

12.1 IP ADDRESSING

As discussed in Section 11.3.1.4, IP uses a numeric routing system to identify subnets and specific addresses. Each IP address contains 4 bytes, usually written in a dotted-decimal format (*A.B.C.D*) that denotes the subnet and unique identifier.

Although IP addresses are commonly represented in the dotted format, they may also be written in hex or decimal. For example, the IP address 10.1.3.100 can also be written as 0x0A010364 or 167,838,564 $((10 \times 256 \times 256 \times 256) + (1 \times 256 \times 256) + (3 \times 256) + 100)$. Although the dotted format is most common, many Web browsers (and other applications) also support hostnames written as integers.

The integer representation is frequently used to obscure some hostnames from being immediately identifiable. For example, people who distribute junk email may use integers for hostnames to bypass some spam filters and to obscure the address from regular users. Fortunately, it is relatively straightforward to convert hex and integers to IP addresses (Listing 12.1).

LISTING 12.1 Sample Code to Convert a Decimal IP Address to Hex and Dotted Format

```

/* Convert a decimal number to an IP address */
/* Displays IP address in decimal, hex, and dotted format. */
#include <stdlib.h>
#include <stdio.h>

int main (int argc, char *argv[])
{
    long Num;

    if (argc != 2)
    {
        printf("Usage: %s 12345678\n", argv[0]);
        exit(-1);
    }

    Num=atol(argv[1]);
    printf("%ld = %08lx = %d.%d.%d.%d\n", Num, Num,

```



```

        (Num >> 24)&0xff, (Num >> 16)&0xff,
        (Num >> 8)&0xff, (Num)&0xff);
    } /* main() */

```

12.1.1 Subnet Classes

The three main categories of subnets (Table 12.1) are class-A, class-B, and class-C. A *class-A* subnet contains 1 bit to designate a class-A style address, 7 bits for the network, and the remaining 3 bytes as unique identifiers (*B.C.D*). There are only 64 different class-A subnets defined by the 7 bits of network addressing, but there are 16,777,216 unique identifiers per subnet. Class-A subnets are commonly referenced by their 1-byte subnet identifiers. For example, the host 10.27.65.2 resides in “Net10”

The *class-B* subnet is defined by having the first 2 bits “10”. The next 14 bits identify the subnet, and the remaining 2 bytes define the unique identifier. There are 16,384 unique class-B subnets. Class-B subnets are commonly described by the subnet bytes *A.B*, with the unique host denoted by *C.D*.

Class-C subnets begin with the 3 bits “110” and use the next 22 bits for the unique subnet. Although there are 4,194,304 unique class-C subnets, each only contains 256 unique addresses. Class-C subnets are commonly described by the first 3 bytes. For example, the host 192.168.15.2 is in the 192.168.15 subnet.

TABLE 12.1 IP Subnet Classes

Class	High-Order Bits	Subnet Range	Unique Nodes per Subnet
A	0	0.x.x.x – 127.x.x.x	16,777,216
B	10	128.n.x.x – 191.n.x.x	65,536
C	110	192.n.n.x – 223.n.n.x	256
D	1110	224.n.n.n – 239.n.n.n	special; multicast packets [RFC3171]
E	11110	240.n.n.n – 247.n.n.n	reserved [RFC3330]
F	111110	248.n.n.n – 251.n.n.n	special; extended addressing [RFC1365]
others	111111	252.n.n.n – 255.n.n.n	reserved

n identifies part of the subnet. *x* identifies a byte for the unique identifier.

The remaining IP network addresses begin with the bit sequence “111” and define a set of extended addresses. These denote special-purpose network addresses,

and few are defined [RFC3330]. One example is Net224. The subnet 224.x.x.x is used for multicast IP addresses.

12.1.2 Network Masks

A subnet can be identified based on a combination of the network address and a *network mask*. A network mask, or *netmask*, is a set of 4 bytes that are combined using a bitwise-AND to define the subnet (Table 12.2). If two addresses, when combined with a netmask, yield the same value, then both addresses are on the same subnet.

TABLE 12.2 Sample Subnet Combination

	IP Address	Binary Format
Address:	10.1.5.100	00001010.00000001.00000101.01100100
Netmask:	255.0.0.0	11111111.00000000.00000000.00000000
Combine Address with Netmask Using a Bitwise-AND Subnet:	10.0.0.0	00001010.00000000.00000000.00000000

The type of network class can be used to determine a netmask. For example, the address 10.5.193.40 can also be written in hex as 0A05C128, or in binary, 00001010 00000101 11000001 00101000. Because the first bit is zero, this is a class-A address. The first 8 bits identify the subnet: Net10. A class-A subnet only has an 8-bit netmask, so the netmask is 255.0.0.0. The netmask may also be written in hex as F0000000 or by the shorthand “/8”. The shorthand notation specifies the number of sequential bits that form the netmask.

Network address space may be subdivided beyond the standard classes. For example, the class-A subnet 12.x.x.x (Net12) may be subdivided based on bit ordering. Net12 may contain any combination of 256 class-B subnets; 65,535 class-C subnets; or fractional subnets. A *fractional subnet* does not use a network mask that ends on a byte boundary. Whereas a class-A subnet uses an 8-bit mask, a fractional subnet may contain any number of bits in the mask. The subnet 12.15.128.0/21 defines the address range 12.15.128.0 to 12.15.135.255 within Net12.



Although netmasks are usually denoted by a series of sequential bits, this is not required. A nonsequential network, such as the netmask 255.255.170.0 (11111111 11111111 10101010 00000000) is just as usable as a sequential netmask.

3. Broadcast Addresses

The IP network protocol supports broadcast addressing. Packets with a destination address of 255.255.255.255 are treated as local broadcast packets [RFC1812]. This broadcast address is not routable; it is strictly used for local IP broadcasts.

1. Routable Broadcast Addresses

Along with the 255.255.255.255 broadcast address, each subnet contains a local broadcast address. The last IP address in a subnet is typically used as a broadcast address. For example, the subnet 192.168.0.0/16 has the broadcast address 192.168.255.255. This permits a remote host to transmit a broadcast packet to a remote network. For the example subnet, the Linux command to ping the entire 192.168.0.0/16 subnet is

```
ping -b 192.168.255.255
```

This command displays all nodes that respond to the ping request. Unlike the Linux command, the Windows `ping` command only shows that *a* system responded. It does not indicate which system responded.



The last IP address is typically a broadcast address, but it is not required. Any IP address can be configured as a subnet's broadcast address. And some network devices support disabling the subnet broadcast address entirely.

2. Determine Subnet Size

Although subnets may appear as class-A, class-B, or class-C, the actual size of the subnet may vary. An attacker can use broadcast pings to determine the size of a remote subnet. By increasing the netmask by 1 bit and determining the relative broadcast address, a ping scan can determine the likely size of the subnet.

To prevent this type of scan from revealing all systems on a network, most routers respond to the echo request but do not forward the packet to the entire subnet. This scan detects the router, not all hosts on the subnet. Other options to deter this type of scan include blocking ICMP echo requests, disabling broadcast addresses, or moving the broadcast address to a different IP address. Although it is possible to configure a router to respond to all ICMP echo requests, this is generally undesirable because an attacker may attempt to perform a detailed attack against all IP addresses that reply.

4. Routing

Each system in an IP network maintains a routing table that associates a network address and netmask with a network interface. When a packet is being transmitted, the destination IP address is compared with the subnet associated with each interface. When a match is found, the data is transmitted through the associated interface.

Routing tables may also associate a *gateway*, or external router, with a particular route. The gateway is a router that will relay packets to the desired subnet. Gateways are used when a remote network is not directly reachable from the transmitting host.

Each routing table contains a *default gateway*. This is a gateway that should receive all traffic that does not match any other known routers. Most PCs and single-homed workstations only contain a default gateway in the routing table.

5. Metrics

Routing tables do not require uniqueness. Two different table entries may route to the same subnet. This overlap permits fail-over redundancy—when one route is unavailable, the next one is attempted. To distinguish between alternate paths, routing tables include a numerical metric. Routes with lower metrics identify more desirable paths. A metric of 0 indicates an adjacent network.

When a packet is ready for transmission, the route with the lowest metric is attempted first. If there is a transmission failure, then the next path is attempted. When two paths have identical metrics, the decision may be resolved through round robin, load balancing, first listed, or default route selection.

Although systems are supposed to use fallback routes, this is not always the case. For example, if a PC has two network cards on the same network, it may conclude that neither route is available if one card fails. Similarly, if the default route is unavailable, some operating systems will not select an alternate default route. Finally, if the data link layer does not report any errors, then the network layer may not detect transmission problems and may not select alternate routes. Each of these conditions appears as a network connectivity problem, even though the routing table and network adaptors may seem to be configured correctly.

Decisions, Decisions

An employee called into a corporate helpdesk to report that his laptop could not connect to the Internet from a conference room. The laptop worked fine from his desk and from his home, but not from the conference room. At first glance, the routing table appeared correct; both the wireless and wired network interfaces were configured properly.

The helpdesk engineer began asking questions about the employee's usage. When at his desk, the laptop used a wired connection. At home, the laptop used a wireless access point. In the conference room, the employee used a wired connection; however, a wireless AP was available. (This caused a subtle problem).

When the laptop was in the conference room, it had two default routes: one over the wired network and one that was wireless. Although the wired network could connect to the Internet, the default route for the wireless network was being selected first. For security reasons, the wireless AP could only access a limited network, not the corporate intranet or external systems. All packets leaving his laptop were using the wireless connection instead of the wired. By disabling the wireless network, the problem was resolved. Alternately, the employee could have removed the default route for the wireless network or assigned the wireless route a higher metric.

6. TTL

IP networks may contain very long routes between two subnets. IP networks may also contain loops. Long routes may be desirable, but loops can quickly consume all network bandwidth. To avoid packets being indefinitely forwarded through loops, IP uses a *time-to-live* (TTL) counter. The TTL is decremented each time the packet passes through a router. If the TTL reaches zero, then the destination is considered to be unreachable and the packet is discarded. For IP, the TTL is 1 byte, with a maximum value of 255. Subnets connected by more than 254 routers are unreachable.

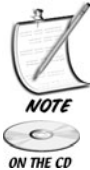
Many operating systems do not begin with a TTL of 255. For example, many Linux and BSD distributions use a default TTL value of 64 [RFC1340]. Windows 2000 uses a default TTL of 128. The default setting is operating system specific and may be changed by a system administrator. Under Linux, BSD, and most other Unix platforms, the command `sysctl -a | grep ttl` displays the default TTL setting. Different TTL settings may limit bidirectional communication. A packet from a sender to a recipient may be deliverable, but the return packet may decrement to a TTL of zero before being delivered.

7. Nonroutable Addresses

Some network addresses are not intended to be publicly available. RFC1918 defines three sets of reserved network addresses: `10.0.0.0/8`, `172.16.0.0/12`, and `192.168.0.0/16`. These addresses are intended for private networks. Other non-routable addresses include extended addresses, such as `250.0.0.0/8`, and loopback addresses (Net127), such as `127.0.0.1`.

12.2 ICMP

One of the network layer's core functions is to provide quality-of-service support. IP does this through the Internet Control Message Protocol (ICMP). ICMP provides support for testing, flow control, and error handling (Table 12.2). Whereas many of the ICMP types are strictly responses, other types define bidirectional query/reply services.



The CD-ROM contains icmpgen.c, a program for generating test ICMP packets.

TABLE 12.2 Common ICMP Functions

Type	Name	Service	Purpose
8/0	Echo Request/Reply	Testing	Determine connectivity
11	Time Exceeded	Testing/Error	Determine if the TTL expired
3	Destination Unreachable	Error	Identify failed delivery
4	Source Quench	Service	Basic flow control
5	Redirect	Service	Indicate more desirable route
17/18	Mask Solicitation	Service	Identify subnets
9/10	Router Advertisement	Service	Identify viable routers

12.2.1 ICMP Support

Each ICMP packet contains three parts. The first part indicates the type of ICMP packet (e.g., `ICMP_UNREACH`, `0x03`), numeric code for the packet type, and checksum for the ICMP header. The second part contains code type specific data. The final part contains packet data.

Although ICMP is a specified requirement for IP, not every system implements all functionality. For example, a host may support echo requests but not support router advertisement. Other examples include the following:

ICMP Checksums: RFC791 defines the ICMP checksum (Listing 12.2), but does not specify the behavior if the checksum is invalid. For example, some versions of Linux appear to ignore ICMP checksums, whereas Windows, OS/2, and BSD drop invalid packets. Many routers ignore checking the ICMP checksum in lieu of faster processing time.

LISTING 12.2 Sample Code to Compute ICMP Checksum

```

/*****
IcmpChecksum(): Compute ICMP checksum.
Packet includes the ICMP header and data.
Returns the checksum in network-byte order.
*****/
int IcmpChecksum      (unsigned char *Packet, int PacketSize)
{
    int i;
    long Sum=0;        /* at least 32 bits */
    int FinalChecksum; /* at least 16 bits */

    /* Algorithm: Add all 16-bit words */
    for(i=0; i<PacketSize-1; i+=2)
    {
        if (i==2) continue; /* skip the stored checksum value */
        /* add to the checksum */
        Sum = Sum + ((Packet[i]*256) + Packet[i+1]);
    }
    /* Take care of any odd numbered packets */
    if (PacketSize % 2) { Sum += (Packet[i]*256); }

    Sum = (Sum >> 16)+(Sum & 0xffff); /* add upper word to lower */
    Sum += (Sum >> 16); /* add carry */
    FinalChecksum = ((~Sum) & 0xffff); /* one's compliment */
    return(htons(FinalChecksum)); /* return in network-byte order */
} /* IcmpChecksum() */

```

Undefined Codes: Each type of ICMP packet may include codes for defining subtypes. For example, type 3 indicates an unreachable destination. The various codes for type 3 describe the actual failure. Yet nearly every operating system responds to an ICMP echo request when an undefined code is provide.

Broadcast Addresses: Some hosts handle broadcast addresses differently. An echo request sent to the 255.255.255.255 broadcast address may receive a different set of replies than a request sent to a specific subnet (e.g., 10.255.255.255 in the 10.0.0.0/8 subnet).

12.2.2 Echo Request and Reply

The ICMP echo service provides a basic test to determine whether a host is accessible. A host will transmit an echo request (*ping*) to a remote system. The remote system receives the request and generates an echo reply (*pong*) back to the calling system. If the transmitting system does not receive a pong, then the remote system may be offline or unavailable. Similarly, the delay between the ping and pong can be

used to measure network latency. The `ping` command is available on most operating systems. This command generates an ICMP echo request and waits for a reply.

Unfortunately, attackers may use the ICMP echo server for attacks and reconnaissance. Common attacks include ping scans, the Ping of Death, and Smurf attacks.

1. Ping Scans

The first step in attacking a system is simply identifying that the target exists. *Ping scans* or *ping sweeps* can be used to search subnets for systems that are online. A ping sweep generates many ICMP echo requests that are directed toward each IP address on a subnet. Each echo reply indicates a potential target for an attacker. Tools such as `nmap` and `spray` can perform ping scans.



A ping scan can be very tough on a network, generating thousands of packets in a few seconds. Repeated scans can be effective DoS attacks by consuming bandwidth, particularly when many hosts reply.

NOTE

An alternate form of a ping sweep uses an ICMP echo request sent to a broadcast network address for the destination. Rather than scanning an entire subnet, a single ICMP echo request is broadcasted to the network, and every host on the network replies.

A direct method to prevent detection from ping sweeps is to disable ICMP echo support. Under Linux, this can be accomplished with the commands:

```
sysctl -w net.ipv4.icmp_echo_ignore_all=1
sysctl -w net.ipv4.icmp_echo_ignore_broadcasts=1
```

Many firewalls contain the option to discard ping requests. However, inexpensive home firewalls may not support disabling ICMP echo requests, or they may only support discarding pings from the WAN.

2. Ping of Death

The echo request includes a payload for testing network capacity and MTU. This permits a network administrator to identify whether network issues are related to packet size. For example, small packets may be delivered, but large packets may fail.

Unfortunately, not all systems handle the MTU properly. In 1996, a vulnerability was discovered where an ICMP payload could be larger than the expected MTU. Dubbed the *Ping of Death*, the oversized packet would cause systems to overflow network stack space, which resulted in a system crash or hang [CERT1996].

The problem stems from an assumption concerning packet sizes. Most transport layer protocols specify MTU size that is smaller than the network layer's maximum transfer size. For example, TCP specifies an MTU no greater than 65,536

bytes, whereas IP can transport 81,854 bytes (size before fragmentation). As such, most operating systems only allocate 64 Kb of address space for packets. But, IP can transmit much larger packets. An echo request can contain more data than the available buffer size, which causes an overflow at the recipient.

In November 1996, nearly all operating systems were vulnerable to the Ping of Death (although there were a few exceptions). Today, most networked hosts are not vulnerable to this attack. But, new network products are constantly released and occasionally vulnerable [Bugtraq2000, Trifinite2004].

12.2.2.3 Smurf Attack

An ICMP packet contains a source and a destination, but no authentication is associated with either address. Hosts that receive an echo request send an echo reply to the request's source address. A *Smurf attack* specifies a victim's IP address as the destination and then sprays thousands of systems with ICMP echo requests. Each system that receives the echo request sends an echo reply to the victim. Even though a single echo reply is a small packet, thousands can easily overwhelm a network. Because the true source of the attack is not associated with the packet, the victim is overwhelmed by a DoS from an anonymous attacker.

Although Smurf attacks have been known since 1998 [CERT1998], there are few local solutions to deter this style of attack. In general, hosts should be configured to ignore ICMP echo requests, and external routers should not forward ICMP echo requests or replies. Egress filtering can be used to impede a host from initiating a Smurf attack by blocking packets with forged source addresses.

Smurf-A-Riffic!

The Smurf attack gained its name from some little blue cartoon characters. Pierre Culliford originally created the Smurfs in 1958. In the early 1980s, Hanna and Barbara animated the Smurfs for NBC. The theme throughout each cartoon was the power of teamwork. Although a single Smurf was weak, a group of Smurfs could conquer any problem.

With regards to the network-based Smurf attack: A single echo reply is harmless, but a million can shut down a network.

12.2.3 Time Exceeded

ICMP provides error-handling support for IP. One such error concerns undeliverable packets. If the TTL of a packet reaches zero before delivery, the router is supposed to drop the packet and transmit an ICMP error message to the source. There are two types of TTL expiration messages: *TTL count exceeded* and *TTL reassembly expired*. A TTL count exceeded indicates that the TTL counter decremented to zero

before being delivered. The TTL reassembly expiration error indicates that fragments began to expire before all parts were received.

The TTL can be used for an uncommon attack, allowing the identification of hosts that reside behind firewalls. The parasitic scanning tool (**parascan**) from the Paketto Keiretsu tool suite (<http://www.doxpara.com/>) varies the TTL field to identify the path to a remote host. This tool operates by performing a replay attack against an already established connection; however, the replay packets are modified with different TTL values. Normally firewalls block route detection; tools such as **traceroute** cannot scan inside a firewall. By replaying traffic from an existing connection, **parascan** can pass through firewalls; because existing bidirectional traffic is already established, the modified packets are allowed through. As each TTL value is attempted, different routers—even those inside a private network—generate ICMP TTL exceeded messages. This allows the scanning host to identify the full path to a protected host.

12.2.4 Destination Unreachable

There are a variety of reasons that the network layer cannot deliver a packet. All of these are reported to the network layer through an ICMP destination unreachable packet. Each unreachable packet includes a code describing the reason for the error. Table 12.3 lists the defined return codes.

TABLE 12.3 ICMP Destination Unreachable Codes

Code	Purpose
UNREACH_NET	Network unreachable
UNREACH_HOST	Host unreachable; implies network is reachable
UNREACH_PROTOCOL	OSI transport layer protocol unavailable
UNREACH_PORT	OSI transport layer port unavailable
UNREACH_NEEDFRAG	Fragmentation required, but the “don’t fragment” flag was set
UNREACH_SRCFAIL	Source routing failed
UNREACH_NET_UNKNOWN	Network is unknown
UNREACH_HOST_UNKNOWN	Host is unknown
UNREACH_ISOLATED	Network or host is isolated
UNREACH_NET_PROHIB	Access to the network is prohibited
UNREACH_HOST_PROHIB	Access to the host is prohibited

Code	Purpose
UNREACH_TOSNET	Type of service (TOS) is unavailable for the network
UNREACH_TOSHOST	TOS is unavailable for the host
UNREACH_FILTER_PROHIB	Prohibited filtering
UNREACH_HOST_PRECEDENCE	Unreachable due to host precedence; indicates low priority host
UNREACH_PRECEDENCE_CUTOFF	Unreachable due to precedence cut-off; indicates low priority traffic

1. Unreachable Packet Format

Each ICMP packet contains three parts. The first part indicates the type of ICMP packet (e.g., `ICMP_UNREACH`, `0x03`), the numeric code for the packet type, and the checksum for the ICMP header. The second part contains code type specific data. For unreachable packets, this is an unused field. The final part contains information that relates to the packet. For unreachable packets, this is the header from the failed packet. The OSI transport layer uses the header to identify the failed connection.

2. Unreachable Attacks

Attackers that monitor IP traffic can create ICMP unreachable packets and transmit them to the packet source. Any router along the path is expected to return ICMP packets for errors, and the routers along the path are unknown to the sender, so the attacker's packet will not appear abnormal. As a result, an attacker can use ICMP to terminate arbitrary network connections. This becomes an effective DoS attack.

3. Unknown Unreachable

ICMP destination unreachable packets are intended to inform a host when a packet cannot be delivered. Yet, there are several situations where a destination unreachable packet may not be generated or received:

TTL Exceeded: Different hosts use different default TTL values. The TTL on the ICMP packet may expire before being delivered. For example, the sender may use a TTL of 128, whereas the router that detected the error uses a TTL of 64. If the sender and router are more than 64 hops away, the ICMP error will never reach the sender.

Firewalls: Many firewalls and routers do not forward ICMP traffic. The error reply may become filtered.

Undetected Failure: The packet may become lost in transit due to a transmission error or routing failure.

5. Source Quench

ICMP provides a very basic form of flow control. If the transmitting system is sending packets too quickly, the receiving system may send a *source quench* request. This request informs the sender to “send slower.” Source quench has a number of limitations:

No Specified Rate: The source quench request does not specify a transmission rate. Different senders will slow by different amounts. A host being overwhelmed by network traffic may need to send many source quench requests.

No Specified Duration: The source quench does not include a duration for the request. The quench request takes effect immediately and lasts for the remainder of the network connection. To remove a source quench request, a new connection must replace the old connection.

No Send Faster: Although source quench is used to request a slower transmission rate, there is no support for increasing the transmission rate. Established connections may get slower and slower, but they can never speed up.

No Authentication: An attacker can flood an established connection with forged source quench requests. The result is an extremely slow network connection.

According to RFC1812, “Research seems to suggest that Source Quench consumes network bandwidth but is an ineffective (and unfair) antidote to congestion.” Routers and nodes may choose to ignore source quench requests altogether.

6. Redirect

When a network contains multiple routers, one route may be more desirable than another. Routers have the option of providing ICMP redirect messages to a host. These packets inform the node to use an alternate route. Using this approach, a single node may have a very small routing table (just a default gateway) and learn of alternate routes dynamically and as needed.

The four types of supported redirection are for a network, host, type of service (TOS) for a network, and TOS for a host. These allow redirections based on the destination’s subnet, specific address, or OSI transport layer service.

The most damaging attacks can come from redirect requests. An attacker can forge a redirect request that informs a host to send data to a nonexistent gateway

(DoS) or to a hostile node, which lead to a MitM attack. For these reasons, secure nodes should ignore redirect requests and use preconfigured routing tables.

To disable redirect support under Linux, use a command similar to

```
sysctl -w net.ipv6.conf.eth0.accept_redirects=0
```

The network adapter (e.g., eth0) may differ between hosts. Windows 2000 can disable ICMP redirects by changing the registry value for `EnableICMPRedirect` in the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters` registry key.

7. Mask Solicitation

ICMP provides a service for one host to request the network mask from another host. Defined in RFC950, a new host on the network may transmit an ICMP mask solicitation. The responding hosts supply enough information for the new host to identify the subnet. The subnet is identified by the responder's IP address and the mask provided in the ICMP data.

Although originally designed for automated network configuration, few systems use ICMP mask solicitation. Instead, other protocols such as BootP and DHCP (both OSI layer 7) are used for configuring hosts on a network. Although most systems do not support this ICMP service, the ones that do may supply a remote attacker with information concerning the network configuration.

8. Router Advertisement

Router discovery provides a way for nodes to discover neighboring routers. Described in RFC1256, each ICMP router advertisement lists the networks reachable from the sender. Hosts may passively receive router advertisements or may actively send an address request. An address request asks all routers to reply.

The information in the router advertisement does not specify the quality of the router. Poor route selections, such as a slow or long path, is intended to be corrected through ICMP redirects.

4. SECURITY OPTIONS

IP is the most common network layer protocol. Any system on the Internet must support IP. Although IP has a significant number of fundamental flaws and implementation-specific vulnerabilities, there are a few steps that can mitigate risk exposure. These include disabling unnecessary features, using nonroutable IP addresses, filtering IP traffic, and employing security-oriented protocols when possible.

1. Disable ICMP

Although RFC791 describes ICMP as providing testing, flow control, and error handling, none of the functions provided by ICMP are essential for network routing. Testing is not always needed, and the physical, data link, and higher layers can better manage flow control. Although the error-handling functions are desirable, they are not always reliable. Combined with the security risks associated with ICMP, it becomes safer and more efficient to completely disable ICMP support.

Unfortunately, most operating systems do not provide the means to completely disable ICMP (aside from recompiling a kernel). As such, ICMP should be bidirectionally filtered at the firewall.

2. Nonroutable Addresses

Hosts that do not need to be directly connected to the Internet can use nonroutable network addresses. RFC1597 defines a set of subnets that are not routable. The subnets are designated for use on private networks (Table 12.4).

TABLE 12.4 NonroutableSubnets

Address Range	Scope
10.0.0.0 - 10.255.255.255	Class-A
172.16.0.0 - 172.31.255.255	16 Class-B
192.168.0.0 - 192.168.255.255	256 Class-C

By using nonroutable network addresses, an attacker on the network cannot directly query the protected host. Moreover, security is no longer based strictly on firewalls and filtering. If a firewall is temporarily misconfigured or disabled, the hosts become isolated from the network rather than directly vulnerable.

Internet access to the private LAN can be provided through dual-homed proxies or network address translation (NAT).

12.4.3 Network Address Translation (NAT)

Network address translation (NAT) is an OSI layer 4 (transport layer) service that relays packets from an isolated network through a common gateway [RFC1631, RFC2663]. The NAT gateway is a dual-homed system bridging the Internet (outside) with a private network (inside).

For each outbound packet, the NAT server stores a mapping in an internal relay table. The table contains (1) the source's IP address and transport layer port, (2) the destination's IP address and transport layer port, and (3) a port number for the NAT server's external interface. The packet is then relayed to the outside network using the NAT server's external IP address and port number. The external destination views the packet as coming from the NAT server and not from the internal system.

When the NAT server receives a packet on the outside network interface, it compares the packet's destination with the internal relay table. The packet is then forwarded to the internal source.

Using a NAT server, many private hosts can relay through the same external IP address. NAT servers provide two security-oriented side effects: anonymity and privacy.

Limited Anonymity: All packets relaying through a NAT server appear to come from the NAT server. Although the external destination of the packet is known, the internal source is anonymous. An attacker may be able to identify the ISP and specific IP address of the NAT server, but the attacker does not know which system behind the NAT server is responsible for the connection.

Nonroutable Privacy: An attacker cannot initiate the connection to an internal host. Any packet that reaches the NAT server's external interface but does not have an internal relay table entry is dropped. The drop occurs because the NAT server does not know how to route the packet—there is no internal destination. This side effect turns a NAT server into a very effective firewall, blocking uninvited external traffic.

Most small-office/home-office (SOHO) firewalls, such as those provided by SMC, Linksys, Netgear, and D-Link, use NAT for the primary firewall functionality. This has the benefit of supporting multiple systems on a single DSL or cable modem connection while providing nonroutable privacy.

IP Reallocations

In the mid-1990s, it became apparent that the allocations of IP addresses were uneven. Some network providers were allocated very large subnets, whereas other providers had very few. Due to the uneven allocation, some networks (and countries!) were running out of IP addresses. Although many solutions were developed, NAT was one of the most successful.

Using NAT, a single service provider could share one routable IP address among thousands of nonroutable addresses. In theory, one external NAT address could support an entire company. In truth, load balancing and port limitations limit the number of hosts that can reside behind a NAT server. Usually NAT servers are configured to provide proxy support for 100 to 1,000 hosts. Special-purpose proxies, such as Web proxies, can alleviate much of the NAT server's network load.

An alternative to using NAT is IPv6. IPv6 provides more network addresses. But IPv6 could face the same allocation issues as IPv4 if the subnets are not carefully distributed.

When addresses first appeared misallocated, a common cry was to reallocate the Internet. Many class-A networks were initially allocated to companies that were believed to be long-term technology leaders. For example, Net3 was allocated to General Electric and Net9 was provided to IBM [IANA2005, RFC1166]. In 2002, Hewlett-Packard (HP) acquired Compaq computers. HP had been allocated Net15. Compaq, having previously acquired DEC, gave HP a second class-A network: Net16. The two class-A networks provide HP with enough address space for over 33 million systems; HP has more address space than all of Taiwan (15,546,372) and Brazil (15,097,741) combined.

While reallocating IPv4 subnets could ease the issues around limited address space, it comes with a hefty cost. Forcing every company in the world to reassign network addresses can cost billions and disrupt Internet operations. Most systems within companies such as Hewlett-Packard, IBM, and General Electric do not need routable network addresses; corporate firewalls prevent Internet traffic from routing directly to hosts. These companies could use private networks with NAT for external access, but the corporate cost currently outweighs the network need.

12.4.4 Reverse-NAT (RNAT)

The largest drawback with NAT is the inability to host a network server. Because all network traffic must initiate from within the private network, a server cannot reside within the private subnet. Normally NAT cannot be used when a network hosts Web, email, or other network services.

Reverse-NAT (RNAT) is an OSI layer 4 protocol that provides a static mapping from an external port on the NAT server to an internal port and IP address on the private network. Using RNAT, an external connection to port 80 (HTTP) on the NAT server can be routed to a Web server on the private subnet.

Many home firewalls provide both NAT and RNAT services. NAT provides the firewall functionality, whereas RNAT permits service-oriented applications. Most of these firewalls also provide support for a catchall server on the private subnet. Sometimes called a *DMZ Host*, the catchall server receives all external connections that are not covered by other RNAT mappings. But unlike a true DMZ, the DMZ host resides in the private network. If the DMZ host is compromised, it can directly attack other private systems.

5. IP Filtering

Most true (non-NAT) firewalls support packet filtering based on the IP packet header. Filter rules can restrict packets based on specific hosts, subnets, or type of service (e.g., TCP, UDP, or ICMP). This filtering can apply to the source or destination address to provide maximum control over IP support.

As mentioned in Section 12.4.1, ICMP traffic should be disabled. An IP filtering firewall can easily be configured to drop all ICMP packets. This effectively filters out the undesirable traffic.

Whereas an OSI layer 3 firewall only views the IP packet header, a layer 4 firewall can also filter based on specific ports and services. This can be used to restrict access to Web or email servers; only Web requests can reach the Web server, and only email can reach the email server.

6. Egress Filtering

Most firewalls are configured to be very restrictive for incoming (outside to inside) traffic while being unrestricted for outbound traffic. This configuration provides the maximum amount of safety from an external attacker while offering the maximum convenience to internal users. But, this configuration allows an attacker within the internal network to stage an attack against an external source. Attacks, such as those that use forged sender IP addresses, can originate from within the internal network.

Egress filtering applies firewall rules to outbound traffic. The simplest egress filters operate at the network layer by preventing packets with forged (or misconfigured) sources addresses from exiting the network. More complicated egress filtering can be performed at higher OSI layers (Table 12.5).

Although IP addresses can be forged, egress filtering prevents forged addresses from exiting a network.

TABLE 12.5 Sample Egress Filtering by OSI Layer

Layer	Type of Filtering
7 (Application)	Web URL restrictions (e.g., anti-porn filters); email/spam filtering; virus scanning
6 (Presentation)	VPN and SSL authentication
5 (Session)	Restrictive DNS and LDAP access
4 (Transport)	Port access restrictions (e.g., forbid IRC or chatroom software)
3 (Network)	Host and subnet restrictions based on network address; inbound and egress filtering
2 (Data Link)	Host based on source hardware address
1 (Physical)	None

12.4.7 IPsec

The main security issues around IP concern authentication, validation, and privacy:

No Authentication: Any host can transmit an IP packet and make the packet appear to come from another host; there is no authentication.

No Validation: IP packets use a simple checksum to validate the content's integrity. If corruption occurs at the network, data link, or physical layers, the checksum will likely identify the error. But the checksum does not provide protection against an intentional modification. An attacker can intercept a packet, modify the contents, and apply a valid checksum.

No Privacy: All IP data is transmitted without any encoding. Unless a higher OSI layer provides some method of encryption, the data is effectively transmitted unencoded (plaintext). An attacker can observe the contents of every packet.

A suite of RFCs was developed to address these limitations. For example, RFC2402 describes an authentication header for IP packets. RFC2409 defines a key exchange system. *IPsec* is a collection of security standards for the network layer. By designing IPsec as a collection of standards, new security options can be added, and less-secure solutions can be removed.

Using IPsec, two hosts can communicate using an authenticated and encrypted network protocol. Authentication may be performed based on the network layer host, subnet, type of service, transport layer port, or application layer user. Unlike regular IP, higher-layer protocols do not need to provide their own encryption and

do not need to be modified to use IPsec. This makes IPsec ideal for secure connections and VPN solutions.

12.4.8 IPv6

IPv6 provides a network layer that replaces the older IPv4 (IP) protocol. IPv4 was used as a basis, whereas IPv6 contains a different (and optimized) header format as well as some significant improvements.

Address Space: IPv6 increases the address space from 32 bits to 128 bits. This mitigates the impact from poor IPv4 subnet allocations. Although IPv6 does provide more address space, the protocol has not been widely adopted yet; it is too soon to tell if the addresses are properly allocated. IPv6 could face the same allocation shortage as IPv4, although it is less likely to happen.

Broadcast Routing: For IPv4, multicast packets and remote network broadcasting were afterthoughts. As such, most IPv4 routers do not support relaying IPv4 multicast packets. In contrast, IPv6 natively supports routing multicast and broadcast packets.

Integrated Security: IPv6 includes functionality for authenticating and encrypting connections at the network layer. Higher-layer protocols do not need to be modified for use over the encrypted channel.

Integrated VPN: IPv6 natively supports encapsulation, which allows network tunneling. Together with encryption and authentication, IPv6 provides a full VPN solution.

From a security aspect, IPv6 and IPsec are essentially equivalent. Both provide strong authentication, encryption, and VPN support. IPv6 includes integrated security features with the option to extend the protocol, and IPsec incorporates a variety of independent security solutions. Many of the features found in IPsec and IPv6 are based on the same RFC specifications.



The main distinction between IPsec and IPv6 is support. IPsec is an optional extension to IPv4. An application may request IPsec, only to find that it is not available. In contrast, the security in IPv6 is mandatory. All stacks that support IPv6 must support the integrated security settings, even if the calling application does not require it to be activated.

From a usability aspect, the effort to switch from IPv4 to IPv6 is nontrivial. Routers and network devices must all be upgraded to support the IPv6 protocol. Many newer systems either support IPv6 immediately or have IPv6 drivers available, but other devices, such as older routers, will need to be replaced. In contrast,

any router that supports IPv4 can readily support IPsec. IPsec only needs to be supported by the source and destination, not by the intermediate systems.

By providing a larger address space, IPv6 forces the use of a larger network header. The IPv4 header adds a minimum of 20 bytes to each packet. In contrast, IPv6 adds a minimum of 40 bytes to each packet. Although the IPv6 header is optimized compared to IPv4, longer network addresses lead to longer headers. The overall result is that IPv6 normally consumes more bandwidth than IPv4.



NOTE

Both IPsec and IPv6 perform bidirectional exchanges before establishing authenticated and encrypted connections. This generates more traffic than the “insecure” IPv4. Similarly, IPsec and IPv6 both include header options for encrypting data. This increases the header sizes for both secure protocols. Taking all of this into account, IPv6 still generates larger headers than IPsec because IPsec uses the smaller IPv4 header for addressing.

2



Transport Layer

In This Chapter

- Common Protocols
- Core Transport Layer Functions
- Gateways

1. COMMON PROTOCOLS

Although there are hundreds of application layer protocols (FTP, HTTP, SMTP, SNMP, and DHCP are only the beginning of a long list), dozens of common physical layer standards (Ethernet, 802.11, etc.), and many choices for network layer protocols (IP, IPv6, IPX, X.25, etc.), there are relatively few transport layer protocols. Unlike the physical, data link, and network layers, transport protocols are usually independent of the lower layers.

The two most common transport protocols are *Transmission Control Protocol* (TCP) and *User Datagram Protocol* (UDP). These are supported by nearly all network layer protocols. Other transport protocols are less common and include SCTP, NetBIOS-DG, and a variety of AppleTalk protocols.

1. TCP and UDP

TCP and UDP are almost universally supported. The main distinction comes from their connection management: TCP is a connection-oriented protocol, whereas UDP is for connection-less communications. Both of these protocols are detailed in Chapter 15.

TCP is commonly used for established bidirectional communication [RFC793]. Application layer protocols such as Web (HTTP), email (SMTP), and Secure Shell (SSH) use TCP. TCP acknowledges all transmitted data, which provides a degree of security through transmission validation.

TCP validates the amount of data transmitted but not the data itself.



NOTE

In contrast to TCP, UDP is connection-less [RFC768]. UDP transmits data but does not validate that the data was received. Because UDP does not wait for acknowledgements, it generally transmits data much faster than TCP; however, UDP does not detect lost packets. UDP is commonly used by higher-layer protocols where the loss of a single packet does not affect the data transfer. For example, streaming audio across the network uses a large number of packets, but each packet contains a very small amount of audio data. The loss of a single packet may result in small audio distortions but does not usually degrade the reception significantly. Streaming video uses even more bandwidth. The loss of a packet may make an image temporarily appear blocky but does not corrupt the entire image. For these reasons, UDP is preferred by many application layer protocols that provide audio

and video, such as VoIP and Microsoft NetMeeting. Although these protocols could use TCP, the delays from connection-oriented packet acknowledgements can noticeably impact quality.

Not every high-layer protocol that uses UDP supports packet loss. For example, the application layer's *network file system* protocol (NFS) uses UDP for fast file transfers, even though a lost packet will result in file corruption or extended delays due to retries.

Transport Overhead

UDP has a very simple packet format. Each UDP packet (called a *datagram*) contains 4 bytes for source and destination ports (2 bytes each—ports are discussed in Section 14.2.1), the length of the datagram (2 bytes), and a 16-bit checksum. The remainder of the datagram contains the data to be transmitted. The entire UDP header contains 8 bytes.

In contrast to UDP, TCP is connection-oriented; it must provide additional information for sequencing and control information. Like UDP, the TCP header includes source and destination ports (2 bytes each), but TCP also include a 4-byte sequence number. TCP is much more sophisticated than UDP and provides options for flow control and data size [RFC793, RFC1122, RFC2001]. Whereas UDP headers always contain 8 bytes, TCP headers can vary in size. The smallest TCP header requires 20 bytes.

When transferring a large file, UDP requires significantly less overhead. For example, when transmitting a 1 MB file (1,048,576 bytes), both TCP and UDP divide the data into packets (or datagrams). Although the amount of data per packet can vary, systems generally try to restrict one transport-layer packet to the data link MTU size. For Ethernet, the MTU is 1,500 bytes, and an IP header consumes 20 bytes. The minimum TCP header is 20 bytes, which results in 1,460 bytes of data per packet, 719 packets to transfer 1 MB, and a total of 14,380 bytes from the header. In contrast, UDP only requires 8 bytes for the header, resulting in 1,472 bytes of data per packet and 713 packets for a total overhead of 5,704 bytes. TCP requires more than a 250 percent increase in overhead compared to UDP—and this does not include the initial TCP protocol negotiation, packet acknowledgements, or connection management (none of which exist in UDP).

14.1.2 SCTP

Although TCP and UDP are widely used, they were not designed for security—both have limited facilities for authenticating data and no method for supporting privacy. In addition, TCP's continual acknowledgements can hinder large data stream transmissions. Designed as a replacement to TCP, the *Stream Control*

Transmission Protocol (SCTP) addresses these shortcomings [RFC2960, RFC3286, RFC3309]. For example, SCTP uses a sliding window for packet acknowledgements. Rather than periodically stopping transmission and waiting for an acknowledgement, an acknowledgement window is provided. Large data streams can continue transmitting while receiving periodic acknowledgements. This provides connection-oriented functionality with a lower overhead than TCP.

SCTP includes support for packet authentication and encryption. RFC2960 defines the option to include a Message Authentication Code in SCTP and to encrypt user data. The cryptographic functions used by SCTP are the same security precautions used by IPsec and IPv6 [RFC1750]. As with IPsec and IPv6, SCTP can only provide security when connecting to a known host and using preshared keys.



RFC2960 is one of the few RFCs that discusses all the security ramifications, including authentication, privacy, and nonrepudiation.

NOTE

Although SCTP provides better security and performance than TCP, it has not been widely adopted. SCTP drivers are available for Linux and BSD, but these drivers do not currently ship with most operating system distributions. In addition, SCTP is not supported by most firewalls and NAT systems and lacks compatibility with TCP. All of this impedes widespread adoption.

14.1.3 NetBIOS-DG

File-sharing protocols, such as NFS, use UDP for data transfers. The lower overhead from UDP makes it more desirable than TCP. For Windows, the protocol of choice is NetBIOS. The *NetBIOS Datagram Service* (NetBIOS-DG) provides similar functionality to UDP. NetBIOS-DG is used for transporting data between Windows systems.



The NetBIOS API provides functionality at the OSI session and transport layers. NetBIOS-DG is a transport layer function, whereas NetBIOS-NS (name server) and NetBIOS-SSN (session service network) are examples of session layer functionality.

NOTE

NetBIOS-DG is an open protocol defined by RFC1001 and RFC1002, however, it is commonly associated with the proprietary Microsoft *Server Message Block* (SMB) protocol. Although the open source community has attempted to reverse engineer SMB for Linux and Unix systems, these ports are constantly under development. The open source SAMBA project (<http://www.samba.org>) offers a very usable implementation of NetBIOS for supporting SMB. But, the SAMBA implementation does not provide 100 percent of the functionality found on Windows systems.

The NetBIOS protocol is strongly associated with Windows operating systems. Microsoft's Network Neighborhood and Shares (drive and device sharing) use NetBIOS to communicate. This protocol was designed for local networks but regularly appears across large networks and the Internet. Unfortunately, NetBIOS is generally considered an external risk because the common implementations provide direct access to services and files on the system. Between November 2004 and 2005, the Internet Storm Center recorded in excess of 500,000 scans per day for the NetBIOS ports 137/udp and 139/udp, and 50,000 for port 138/udp. Most of these scans were conducted by malware and computer viruses. Together, these ports denote some of the most widely scanned and exploited services on the Internet. To limit virus outbreaks, many ISPs automatically filter these NetBIOS ports.



Although the NetBIOS ports 137, 138, and 139 (TCP and UDP) are heavily scanned, they are not as targeted as the SMB-over-TCP port 445/tcp. Between November 2004 and 2005, the Internet Storm Center recorded in excess of 5,000,000 packets scans per day. Although many service providers filter the NetBIOS ports, 445/tcp is not regularly filtered.

14.1.4 AppleTalk

Similar to NetBIOS for Windows, Apple's operating systems incorporate the AppleTalk protocol. AppleTalk includes a variety of transport protocols that are optimized for specific tasks. For example, the AppleTalk Data Stream Protocol (ADSP) is optimized for streaming data, whereas the Name Binding Protocol (NBP) is used for hostname resolution. Other AppleTalk transport protocols include the AppleTalk Transaction Protocol (ATP) and Echo Protocol (AEP).



Most Apple products have strong support for AppleTalk, TCP, and UDP.

Although AppleTalk is a proprietary protocol, the open source Netatalk project (<http://netatalk.sourceforge.net/>) provides full AppleTalk support to Linux, BSD, and Unix systems.

14.2 CORE TRANSPORT LAYER FUNCTIONS

The transport layer abstracts the networking functionality from the upper application layers. This includes connection management, packet assembly, and service identification. The two core elements that make these functions possible are transport layer ports and sequencing.

14.2.1 Ports and Sockets

The transport layer uses the network layer to establish a connection to another node. The network layer's route provides a *socket* across the network [RFC793]. Sockets may either be active or passive. An *active socket* indicates an established network connection, either on a client or server. In contrast, a *passive socket* is one that is not in use. Servers use passive sockets while awaiting and listening for network connections.

The transport layer creates *ports*. Each port contains a unique identifier for a specific higher-layer service. A single socket may host many ports, but a port requires a socket. Ports may be associated with specific higher-layer protocols or dynamically allocated. For example, the Web (HTTP) uses a service on port 80 of the TCP protocol—denoted as 80/tcp. Email uses 25/tcp, and DNS uses both 53/udp and 53/tcp. By assigning services to known ports, every client knows how to find each service. If the service is not on a standard port, then someone (or something) must specify where the port is located. For example, the remote procedure call (RPC) protocol uses a well-known port for communicating the dynamic port numbers for specific services.

Although servers are usually assigned specific ports, clients may use any port. The server learns the client's port number when the client connects to the server. Each transport layer header specifies the source and destination port numbers.

Most remote network attacks target specific services on specific ports. But, many denial-of-service attacks may target ports or sockets. This places the transport layer in a unique position: transport attack vectors include ports, sockets, and specific protocol implementations.

Socket to Me

The term *sockets* has two meanings that are context specific. First, it can refer to a general programming interface. Unix *sockets programming* uses the sockets API for accessing transport layer protocols. Within the transport layer, however, sockets have a different meaning. A socket is a network connection, defined by four values: source network address, source port, destination network address, and destination port [RFC793]. A half-opened socket indicates a socket with source values but no destination.

14.2.2 Sequencing

The transport layer receives a block of data from the upper layers and separates the data into *packets*. Each packet is assigned a unique *sequence identifier* that is used to track the packet. The transport layer maintains a list of sequence numbers that

have been used on a port—new sequence numbers are not used until the entire sequence is used. This prevents a packet with a duplicate sequence number from being confused with a previously seen packet. Packets that are clearly out of sequence are either held until the full sequence is received or simply rejected.

Because sequence numbers are used to maintain the transport layer's ordering, they form a common attack vector. Sequencing can be used for transport layer hijacking.



For some connection-less protocols, packet ordering and sequencing are not required. UDP, for example, does not provide sequencing.

NOTE

3. Sequence Hijacking

An attacker who observes a transport layer packet must identify the sequence to insert or hijack the connection. If the attacker sends forged packets that are out of sequence, then they will likely be ignored by the target system; however, forged packets with valid sequence numbers can appear authentic and acceptable to the target system. For this reason, sequence numbers may not be generated incrementally. If the observed packet sequence is #1, #2, #3, then the attacker may assume the next packet is #4. If the observed sequence is #1, #352, #80, then an attacker may not be able to predict the next sequence number. In addition, if all sequence identifiers are used before the sequence repeats, then an attacker can simply observe all identifiers until there is only one left.

1. Random Sequence Numbers

Random initial sequence numbers are commonly used at the start of a transport connection. This deters an attacker from guessing the first packet; however, random sequence numbers within an established connected-oriented transport connection are uncommon. Instead, transport layer protocol such as TCP and SCTP use the packet header to indicate the current sequence number as well as the next number. An attacker who observes this type of header knows the next sequence number to imitate. In addition, the attacker can specify the next forged packet sequence number, effectively locking out half of an established connection. If the transport connection provides an established login, then the attacker immediately gains access to the login.

Because a socket connection includes both source and destination, a hijacker must assume the identity of either the source or the destination. This can be done through redirection or impersonation. For example, ICMP allows the redirection of connections to other hosts or ports. In addition, the victim may undergo some form of DoS attack to remove any competition with an impersonator.

2. Blind Sequence Attacks

Although common transport protocols are vulnerable to attackers that can observe the packets, they are not necessarily vulnerable to attackers that cannot intercept the data stream (*blind attackers*). Sequence numbers are commonly started with an initial random value but increment throughout the course of the session. An observer can easily identify the sequence pattern and hijack the session, but a blind attacker cannot identify the initial sequence number. As such, a blind attacker cannot determine the sequence number and compromise a transport layer connection.

3. Sequence Reconnaissance Attacks

Unfortunately, many transport layer protocol implementations use a pseudo-random number generator that becomes predictable. The complexity of the random number generator directly leads to the predictability of the packet sequence. An attacker can establish many transport connections and determine the pattern for initial sequence numbers. This weakness allows a blind attacker to potentially compromise transport layer connections. Older operating systems, such as TCP for OS/2 and Windows 95, use very predictable packet sequences. Newer network devices may use random positive increments (e.g., Linux and Windows 2000) or very random values (e.g., OpenBSD and related BSD systems). Tools such as Nmap (`nmap -o -v`) evaluate the quality of the sequence generating system.



Although most modern operating systems use very random algorithms for generating transport layer sequence numbers, network gateways and other devices do not. Many gateways, firewalls, and mobile devices use trivial, pseudo-random number generators.

4. Hijacking Mitigation Options

There are few solutions to transport layer hijacking. Most services that require security rely on higher OSI layer protocols to authenticate connections. For example, SSH connections can be hijacked through TCP, but the hijacker is unlikely to be able to authenticate or respond correctly to SSH packets. This type of hijacking appears as an established SSH connection that suddenly disconnects. This results in a secure failure, where the service stops the attack but does nothing to prevent an ongoing DoS attack. Secure transport layer protocols, such as SCTP, are seldom deployed but do provide options for transport authenticating connections between known hosts.

14.2.4 Connection Management

Much of the transport layer functionality operates similarly to the lower networking layers. For example, transport layer protocols may be connection-oriented or connection-less. In a *connection-oriented* protocol, the recipient acknowledges each data transfer; *connection-less* protocols do not. This is the same concept used by the OSI network, data link, and physical layers—each supports connection-oriented and connection-less protocols. There is no connectivity dependency between the OSI layers, however. A stack may be implemented with a connection-less physical layer (e.g., 10Base-2), a connection-oriented data link layer (IEEE 802.5 Token Ring), a connection-less network layer (IP), and a connection-oriented transport layer (TCP). Even if the transport protocol is connection-oriented, the lower protocols are not required to be connection-oriented.



Although it is possible to have connection-oriented protocols at each of the OSI layers, the resulting communication overhead can be overwhelming. Using connection-oriented protocols at the data link, network, and transport layers can more than quadruple the amount of physical layer traffic.

The ability to mix-and-match connection methods allows improved performance and reliability. For example, many computers only have one network adapter that supports a twisted pair physical layer protocol (commonly 10Base-T or 100-Base-T). These are connection-less protocols. The data link layer, IEEE 802.2 and 802.3, can support connection-oriented connections but nearly always operates in a connection-less mode. The network layer, IPv4 or IPv6, is connection-less (unless security is enabled through IPsec or IPv6 security). This means that the data transfer is likely connection-less. The transport layer can apply connection-oriented functionality even when the lower layers are connection-less. Because only one layer is providing a connection-oriented service, data does not need to be validated at every step. A single acknowledgement can be received at a higher OSI layer and validate the data transfer.

In the worst-case performance scenario, each layer uses a connection-oriented protocol. This means that each transport layer packet needs a transport layer acknowledgement. A single transport layer packet may be split into multiple network layer packets, resulting in multiple network-layer acknowledgements for each transport layer packet. Similarly, the data link and physical layers may subdivide the data and require transfer acknowledgements. Although this type of worst-case scenario is rarely, if ever, observed in common network environments, the scenario does appear to a lesser degree with VPNs. For example, both IPv6 and SSH provide VPN support, and both provide connection-oriented services. Tunneling a protocol that depends on TCP over SSH through an IPv6 connection introduces three

layers of connection-oriented services and therefore leads to more packets being generated.

In contrast to the performance loss, connection-oriented layers do improve reliability by validating the transfer at each stage. The four types of connection-oriented confirmations are per packet, per set, sliding window, and implicit. Each of these approaches balances tradeoffs between performance and reliability.

Proxies and Tunnels

Both proxies and tunnels relay packets. Both also typically use connection-oriented services, although connection-less protocols do exist. The big difference between proxies and tunnels is in *how* they relay packets.

When using a *tunnel*, such as a VPN, an entire network stack is relayed. If an application protocol that uses the TCP connection-oriented transport protocol (e.g., HTTP or SMTP) is tunneled, then all TCP acknowledgements must also traverse the tunnel. Because all traffic is tunneled, the result is a secure connection, but the number of packets needed for sending the data increases.

In contrast, *proxies* only relay higher-layer protocols. A transport layer proxy, such as a SOCKS server (Chapter 13), only relays session, presentation, and application layer protocols. Transport layer acknowledgements are not relayed through a transport layer proxy. Similarly, NAT (Chapter 12) functions as a network layer proxy. When using NAT, transport layer acknowledgements are relayed, but network layer acknowledgements are not. In general, tunneling generates more packets, which leads to more network overhead and lower performance but also more security options. In contrast, because proxies do not relay acknowledgements, they usually generate less traffic and offer higher performance but only add security through indirection.

Tunneling protocols that use cryptography and authentication are less vulnerable to eavesdropping, hijacking, and MitM attacks than proxies with cryptographic support. This is because a transport layer proxy must decode the transport layer before relaying the data.

14.2.4.1 Per-Packet Confirmation

The simplest type of connection-oriented service uses *per-packet confirmation*. Each transmitted packet receives an acknowledgement. This increases the bandwidth at the physical layer, regardless of the layer performing the per-packet confirmation. To reduce bandwidth consumption, most protocols generate very small acknowledgement packets; the acknowledgement is generally smaller than the data.

2. Per-Set Confirmation

In *per-set confirmation*, a group of packets may be transmitted with one acknowledgement accounting for the entire group. The NFS application layer protocol uses per-set confirmation for file transfers. The confirmation reports the packets received, as well as the ones that were missed.



Many applications that use UDP can also use TCP. For example, NFS may use TCP or UDP but uses UDP by default.

3. Sliding Window Confirmation

The *sliding window confirmation* approach extends the concept of per-set confirmation. Rather than transmitting a block of data and then waiting for a reply, the sliding window continually transmits data. Acknowledgements are sent periodically and confirm a set of transmitted packets. Some acknowledgements may identify a few packets, whereas others can identify many. Sliding windows result in faster transmission rates because the sender does not stop unless no acknowledgements are received. Sliding window confirmation is used by protocols such as TCP and SCTP.

Unfortunately, sliding windows also offer attackers a window of opportunity. In per-packet confirmation and per-set confirmation, the attacker must respond faster than the destination's acknowledgement. With sliding windows, there is a longer period for the attacker to attempt to hijack or poison a connection. Chapter 15 discusses blind TCP attacks that exploit weaknesses in the sliding window approach.

4. Implicit Confirmation

Not every connection-oriented protocol is explicitly connection-oriented. Some protocols, such as those that use cryptography, may be designated as connection-less but actually provide connection-oriented acknowledgements. For example, when the presentation layer protocol SSL (Chapter 19) uses encryption, transport layer packets are chained together through the SSL encryption. Even if the transport layer is connection-less (e.g., UDP), the overall transport operates in a connection-oriented fashion; if a packet is not confirmed, then the encryption fails.

Encryption within the transport or higher layers can lead to implicit connection-oriented communications. Examples include the Kerberos session layer protocol and the presentation layer protocols SSH and SSL.



Few transport protocols support authentication or encryption, but none also support connection-less cryptography. SCTP, for example, supports authentication and encryption but is a connection-oriented protocol. Connection-less transport protocols, such as UDP and NetBIOS-DG, do not provide cryptographic support.

14.2.5 Packet Ordering

The transport layer may divide data into multiple packets, combine small data blocks into larger packets, and multiplex packets through multiple connections. Moreover, the network layer routes the packets to remote systems but does not guarantee the order of the packet's arrival. Different routes may result in different packet ordering. To ensure that the upper OSI layers receive data as it was sent, the transport layer uses the packet sequence numbers to ensure data ordering.

The Nagel Algorithm

Packet headers generate a large amount of overhead. For example, a TCP packet sent over IP on a 100Base-T network contains a minimum of 54 bytes of packet header: 14 for MAC addressing, 20 for IP, and 20 for TCP. When transmitting a series of short bytes, such as keyboard entries in a terminal window, the header size comprises a majority of the data being sent. If there are enough small packets, then network congestion due to acknowledgement delays may occur.

In 1984, John Nagel proposed a solution to this type of network congestion. Detailed in RFC896, the approach delays transmissions at the transport layer. Rather than sending every keystroke, systems may wait for a fraction of a second (usually 200 ms to 500 ms—the time between keystrokes) in case more small data blocks need to be transmitted. When the transfer buffer fills, or a timeout occurs, the data is placed in a packet and sent through the socket, reducing the number of headers needed to perform the transfer. This method of waiting for a packet to fill a transport buffer is commonly referred to as the *Nagel Algorithm*.

The Nagel Algorithm does improve network performance, but it can negatively impact software that requires real-time input. Applications may disable the Nagel algorithm by using the socket option `TCP_NODELAY` (Listing 14.1) to force real-time data transmissions. But, disabling the Nagel algorithm can lead to time-based attacks, such as the SSH timing attack discussed in Chapter 20.

LISTING 14.1 Code Sample for Disabling the Nagel Algorithm for TCP Connections

```
int On=1;
tcp_socket = socket(PF_INET, SOCK_STREAM, 0);
setsockopt(tcp_socket, IPPROTO_TCP, TCP_NODELAY, &On, sizeof(On));
```


14.2.6 Keep-Alive

Network and lower OSI layers may connect and reconnect as needed, without impacting the transport layer. Network devices such as NAT routers (Chapter 12), however, may decide a transport layer connection is terminated when no traffic is seen after a few minutes. NAT devices and firewalls normally store a table of addresses, ports, and protocols that are permitted to enter the network. After a timeout from lack of use, table entries are dropped. The result is that a low-volume transport layer connections may become disconnected due to network timeouts.



The connection timeout duration varies among firewalls. Some SOHO NAT firewalls timeout after 10 minutes. Heavy-duty commercial firewalls may timeout after hours.

To prevent network timeouts, many transport layer protocols support keep-alive packets. A *keep-alive packet* creates network traffic without transferring application data. These are intended to prevent timeouts from low activity.

Low-end network devices simply look for traffic from a known remote network address that uses a specified protocol (e.g., TCP or UDP) and port. An attacker can use this information to keep a path through a firewall or NAT router open long after the actual connection has terminated. By sending forged packets that specify the source address, destination address, protocol, and port, the attacker can keep the path through the firewall open indefinitely. In the worst case, the network service inside the firewall may be susceptible to attacks from forged packets. At best, the firewall (or NAT device) is vulnerable to a DoS as the attacker keeps all ports open—filling the table of allowed connections and preventing new connections.

Higher-end firewalls and NAT devices support *stateful packet inspection* (SPI). SPI monitors transport layer packets to identify sequence numbers and connect/disconnect requests. Using SPI, an attacker cannot trivially bypass a firewall; however, SPI can only analyze transport protocols that it knows—usually TCP and UDP. Other transport protocols, such as NetBIOS-DG or SCTP are usually not be monitored by SPI systems.

14.3 GATEWAYS

Bridges, routers, and gateways refer to network devices that convert between protocols. *Bridges* (Chapter 8) operate at the data link layer and span networks that may use different physical layer standards. *Routers* (Chapter 11) operate at the network layer and can pass traffic between separate data link layers. Similarly, *gateways* are transport layer devices that can pass data between different network layer

protocols. For example, a gateway may link traffic from an IPv6 network to an X.25 network.

Gateways operate at the transport layer and can filter connections based on data link, network, or transport protocols. For example, a gateway may only pass authenticated IPv6 traffic a private IP network or allow all TCP data to pass from a specific hardware address to a specific subnet. Because of its filtering capabilities, the term “gateway” is frequently synonymous with “firewall.”

What's in a Name?

The term *gateway* may be used with network or transport layer devices, but with different meanings. At the network layer, a gateway is any device that links two networks. This term is usually seen as a *default gateway* or default routing device.

At the transport layer, *gateway* has a very different meaning. *Bridge*, *router*, and *gateway* define specific terms that correspond with OSI stack functionality (layers 2, 3, and 4, respectively). Not all devices with these names operate at these layers, however. For example, many home firewall routers support transport-layer analysis. The capability to perform stateful packet inspections (SPI) requires transport layer support—a router at the network layer cannot perform SPI. The NetGear *WGR614 Wireless Router* is technically a gateway because it supports SPI. Similarly, the SMC *Barricade Router* (e.g., the 7004ABR) provides RNAT based on port and protocol (this requires gateway functionality). To add to the terminology confusion, Belkin offers a product called a *Wireless DSL/Cable Gateway Router*.

Product names do not necessarily reflect their functionality or technical definitions. A device labeled as a *router* may actually be a gateway, and a *gateway* may actually be a router. Similarly, many devices marketed as *hubs* may actually be switches, bridges, routers, or gateways. When evaluating network devices for security implications, be aware that the product's name may not match the functionality.

2 **TCP**

In This Chapter

- Connection-Oriented Protocol
- TCP Connections
- UDP

15.1 CONNECTION-ORIENTED PROTOCOL

TCP is a connection-oriented protocol and offers two guarantees for higher layer protocols: reliable data delivery and sequential data ordering. As a transport layer protocol, TCP also provides facilities for flow control and multiple connections [RFC793].

TCP uses a byte-stream data flow model. In this model, the byte ordering is important, but the TCP packets are not. TCP packets may be combined or split as long as the data's byte ordering is maintained. This is different from other protocols such as IEEE 802.1, PPP, and IP, where the message blocks delimitate data flow.

TCP uses a segmented header to enclose transmission data (Figure 15.1). Besides the basic 20-byte header, TCP also may include optional header information. The smallest TCP packet (containing no data and no options) is 20 bytes, whereas the largest can exceed 64 KB. Usually a TCP packet with data is only a few kilobytes in size, which ensures that one TCP packet can fit within one data link layer frame.

The data being transmitted by the TCP packet is often called application data or user data. Although the data may come from the application layer or user layer (outside the OSI stack), the session or presentation layers can also generate transport data.



NOTE

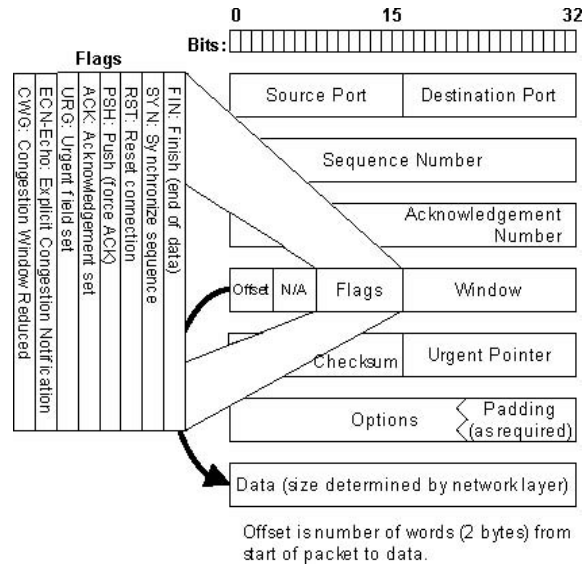


FIGURE 15.1 A basic TCP header.

1. Reliable Acknowledgements

TCP guarantees reliable data delivery. Each byte sent by a TCP packet is confirmed with an acknowledgement. If no acknowledgement is received, then it assumes that the data transfer failed. To reduce the number of acknowledgements, TCP uses a windowing system. This system acknowledges the amount of data received.

1. Checksums

TCP encases the application data within a TCP header. The TCP header includes a simple checksum value to detect transmission corruptions. The checksum function is a 16-bit CRC function, similar to the checksum used by IP, and covers the entire TCP header and application data. Corrupted packets with invalid checksums (that somehow manage to pass lower layer checks) are discarded and not acknowledged.



NOTE

Although RFC791 and RFC793 state that TCP packets with invalid checksums should be discarded, some TCP implementations do not validate checksums. By skipping the validation, portable devices, gateways, and packet filters can reduce per-packet processing time and improve speed performance.

15.1.1.2 Retries

To ensure data is properly delivered, TCP resends unacknowledged data. The retry time interval varies based on the network speed, but is usually less than 2 minutes. Data transfers may be retried many times before the connection is assumed to be inactive and closed. This long duration can hang applications that wait for TCP to deliver data. For example, if an application blocks on writing to the network socket, then the application may appear temporarily hung while TCP waits for transfer confirmation. If there are 7 retries, 1 minute apart, then an application may hang for 7 minutes before detecting the failed connection. Under most Linux systems, TCP defaults to 15 retries (see the Linux command, `sysctl net.ipv4.tcp_retries2`); delays under Linux can be as long as 30 minutes. Most Windows systems retry 5 times, resulting in shorter delays.

Retransmission timeout (RTO) durations are based on the *round trip time* (RTT) of each TCP packet. This is a measurement based on the time it takes to receive an acknowledgement for a given packet transmission [RFC793]. In general, faster networks have smaller RTT values, leading to a shorter retry duration. In the best case, a fast network will result in rapid detection of unacknowledged packets. In the worse case, TCP may hang for minutes as it times out and retries.

15.1.2 Sequence Numbers

TCP guarantees the order of application data delivery. Although lower layer protocols may transmit and receive packets out of order, TCP will reconstruct the ordering before passing the data up the stack to an application. The reconstruction is based on sequence numbers. Each TCP packet is assigned a sequence number, and subsequent packets increase the sequence number by the amount of data transported. For example, if the sequence number is 12345 and 100 bytes of application data are transmitted, then the next sequence number will be 12445.

TCP provides full-duplex communication; both sides may transmit data at the same time. To minimize the number of packets, each TCP packet may data transfer *and* acknowledge a previous transmission. The packet header stores the current sequence number for the data being transmitted, as well as an acknowledgement for previous data received.

TCP does not acknowledge data receipt directly. A direct confirmation would acknowledge each transmitted packet. Instead, the TCP *acknowledgement packet* (ACK) confirms the data but not the packet itself. Each ACK includes the next expected sequence number. If all data has been received, then the next expected sequence number would match the next transmission's sequence number. Using this approach, a gateway or proxy can collect packets, combine (or split) TCP data, and reformat the data into new packets. The reply identifies the amount of data received and not individual packets.

3. Flow Control

TCP manages data flow using control flags and windows. These two methods combine to provide transport state, acknowledgements, priority, and optimized buffering.

1. Control Flags

Each TCP packet header contains 12 bits for use as control flags. One flag denotes an acknowledgement, another identifies a reset request, and a third defines urgent data. In total, there are six primary control flags (Table 15.1). The flags do not need to be used symmetrically. For example, the client may send a reset to the server to tell the server to resend all unacknowledged data; however, this does not reset the data from the client to the server—it only resets the state from the server to the client.

TABLE 15.1 TCP Control Flags [RFC793, RFC3168]

Name	Bit	Purpose
FIN	0x001	Finish; denotes the end of a connection
SYN	0x002	Synchronize; denotes the start of a connection
RST	0x004	Reset; requests a connection reset
PSH	0x008	Push; require recipient to passed all data up the stack and send an acknowledgement
ACK	0x010	Acknowledge; sender acknowledges receipt of data
URG	0x020	Urgent; data in this packet takes precedence over regular data transfers and handling
ECE	0x040	Explicit Congestion Notification – Echo; allows notification when packets are dropped
CWR	0x080	Congestion Window Reduced; notifies recipient when less window space is available
Remainder	0xF00	Reserved

15.1.3.2 Window Size

In a basic connection-oriented system, each transmission is acknowledged. The acknowledgement confirms data delivery, but creates significant overhead and delays. For example, if the transport layer needs to pass 100 KB of data, it may segment the data into 100 packets containing 1 KB each. Each of the 100 packets would require a confirmation—another 100 packets. To reduce the amount of bidirectional communication, TCP allows each packet header to define the amount of data that can be transferred before requiring an ACK. If the client specifies a window size

of 8,192 bytes, then the server can send up to 8,192 bytes of data before requiring an ACK.

For large transfers, larger window sizes can be used; however, the TCP header only supports 16 bits for the window size—a maximum of 65,535 bytes. RFC1323 defines an optional extension to the TCP header for window scaling. The scale indicates the number of bits to right-shift the window size. As an example, a window value of 0x05B4 indicates a 1460-byte window. But when combined with a right-shift of 3 (multiply by 2^3 , or shift the bits 3 positions), the value denotes 11,680 bytes.



A window size of 1460 bytes is very common. It is based on an Ethernet packet data size of 1,500 bytes (MTU 1500), minus the 20 bytes for the IP header and the 20-byte minimum for the TCP header. A shift of “3” indicates that the recipient can buffer up to eight (2^3) packets before forcing an acknowledgement.

Even when using large transfer sizes, it can be desirable to force an acknowledgement before the recipient’s buffer fills. The sender’s *push flag* (PSH) is usually sent with the last packet in a large block. This flag informs the recipient to acknowledge the entire data transfer before the next transfer begins.

15.1.4 Multiple Connections and Ports

The source and destination port numbers are combined with the source and destination network addresses to define each TCP connection. Data from one set of ports and addresses (one connection) is not intermixed with data from another set of ports and addresses. This allows the host system to manage many connections at once.

A TCP server binds to a specific port. For a client to connect to the server, it must first acquire its own local port number. Then it connects to a server and establishes a unique transport session defined by the network client and server addresses as well as the transport client and server ports. This unique combination of source and destination ports and network addresses allows a single Web server to operate on 80/tcp while transferring data to a variety of clients. As long as the clients have different network addresses—or the same network address but different client ports—the connection remains unique.

15.2 TCP CONNECTIONS

TCP maintains three basic states: initialization, data exchange, and disconnect.

15.2.1 TCP Connection Initialization

Initialization uses a *three-way handshake*: SYN, SYN-ACK, and ACK (Figure 15.2). The client initially sends a TCP packet containing the SYN flag to a specific port on the server. The packet contains the initial sequence number that will be used by the client. The server replies with both SYN and ACK flags set (SYN-ACK). The server sets its initial sequence number and acknowledges the client's sequence number. Finally, the client transmits an ACK and confirms the server's sequence number. Using this approach, both sides confirm the established connection. If the ACK from either side fails to transfer, then data cannot be transmitted and the failure can be detected.

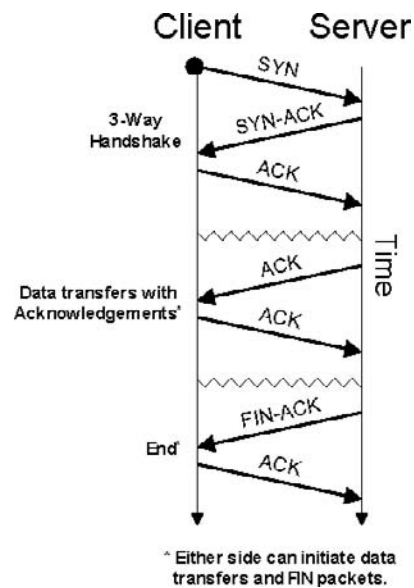


FIGURE 15.2 TCP three-way initialization handshake, data transfer, and connection termination.

Besides exchanging initial sequence numbers, the TCP header may also include information concerning the window size and other TCP options; however, the initial handshake does not transfer any application data.

The SYN flag is only used during the initial connection. Any SYN flags sent after establishing the connection are either ignored or generate a reset packet. Because the SYN packet may be retransmitted due to an initial connection timeout or duplicated by lower network protocols, it is possible for duplicate SYN packets to

be received. After the SYN-ACK is transmitted, the client's sequence number is incremented by one. This makes duplicate SYN packets appear with invalid sequence numbers, forcing the packet to be ignored. In contrast, a SYN with a valid sequence number indicates a network problem (or an attack) and forces a connection reset.

2. TCP Data Exchange

After establishing a TCP connection with the three-way handshake, data packets may be transmitted. Each data packet includes an ACK flag and the data. When the client sends data to the server, the server increments the client's sequence number (a counter) and acknowledges the data with an ACK packet. The server may include response data along with the data acknowledgement.

To reduce the network congestion from acknowledging every packet, the window size identifies the amount of data that may be transmitted before requiring an acknowledgement. If the receiver's window size is 3,000 bytes, then the sender may send up to 3,000 bytes before requiring an ACK. But there is a risk: the receiver assumes that the sender is tracking the amount of data transmitted. However, there is nothing to stop the sender from transmitting more data. The receiver must ignore or drop data received after its buffer fills and before the ACK is transmitted. For this reason, both sides in a TCP connection must track acknowledgements to ensure that no data is lost. The sender knows information was lost when no ACK is received.

When neither side transmits data, the idle connection takes no network bandwidth. Network proxies and NATs may view inactivity as a terminated connection. There is an extension to TCP for generating keep-alive packets [RFC1122], but it must be listed as a supported TCP option during the initial three-way handshake; support is not mandatory nor guaranteed. A keep-alive packet may break a connection or generate a reset if the recipient does not support it. In general, higher-layer protocols that anticipate long periods of inactivity are expected to generate their own keep-alive packets.

3. TCP Disconnect

TCP identifies the end of a connection when one side transmits a packet containing a *finish flag* (FIN). The closure is completed when the recipient transmits an ACK for the FIN. As with the SYN packets, the FIN packets are not expected to transmit data. If any data is transmitted after the FIN, then the recipient responds with a reset (RST).



NOTE

Using a FIN packet generates a graceful shutdown of the connection, but aborts can also terminate connections. An abort is signaled by an immediate RST (without a FIN) and can happen at any time.

7. UDP

TCP provides a connection-oriented service that guarantees packet delivery and order of delivery; however, the overhead from guaranteed delivery is not always essential. Some higher-layer protocols, such as H.323 (used by VoIP and NetMeeting; <http://www.openh323.org/>), manage message ordering and retries. In other cases, network connectivity is direct and reliable, so dropped packets and resequencing are not significant risks. And some protocols, such as DNS, use short information transfers, so elaborate connection validation ends up generating more network traffic than the actual application's request and reply. As a result, full connectivity at the transport layer is not always essential. The *User Datagram Protocol* (UDP) is a simplified transport protocol commonly used for connection-less services [RFC768].

Because many of the TCP functions are not needed (e.g., sequence numbers and flow control), UDP packets have a simplified header. The header only includes a source port, destination port, data length, and checksum. Whereas a TCP header adds a minimum of 20 bytes to each packet, UDP only adds 8 bytes. In addition, UDP transmissions do not generate acknowledgements.

Although UDP requires significantly less network bandwidth than TCP, it is more prone to attack. UDP attacks are usually based on unvalidated packets and impersonation.

1. Unvalidated Inbound Sources

In TCP, each client is authenticated with the server through the use of a three-way handshake. Completion of the handshake implies a valid network host. In contrast, UDP servers perform no initial handshake. Any host can connect to a UDP server, and the connection is not authenticated.

With TCP, the server faces flooding from SYN packets but usually not from ACK or RST packets. In contrast, any type of UDP packet can potentially flood a server. Servers buffer a finite number of UDP packets. Under Linux, the command `sysctl net.unix.max_dgram_q1en` shows the default number of queued UDP packets (usually set to 10). If more UDP packets are received, then the additional packets are dropped until the buffer is emptied. UDP packets can quickly overwhelm slow UDP services. For example, a slow computer receiving streaming audio may drop UDP packets, resulting in clipped audio.

2. UDP Hijacking

UDP servers receive packets from any hosts without authentication. As a result, any client can send a packet to any UDP server. Management of any session or connection must be handled by higher-layer protocols. Unfortunately, this means that UDP is very vulnerable to hijacking. An attacker can easily forge the correct network addresses and UDP ports to insert data into the recipient. A blind UDP attack only needs to guess the ephemeral port number—no more than 65,536 guesses and usually only takes a few seconds. In contrast, a blind TCP attack can require hundreds of thousands of packets and takes minutes to guess the port and sequence numbers.

3. UDP Keep-Alive Attack

TCP has clear designations for opening and closing a session. A SYN initiates a connection, whereas a FIN closes the connection. Firewalls and proxies can dynamically open and close connections as needed.

In contrast to TCP, UDP has no clear indicators for open or closed connections. As a result, most firewalls open UDP ports when the first outbound connection is seen but do not close the port until after a period of inactivity. Attackers can use this weakness to hold open UDP ports. Even if the client is no longer listening to the packets, an attacker can send UDP packets into the firewall to keep the firewall's port open. If enough ports are kept open, then no new ports can be opened. This effectively disables UDP through the firewall.

4. UDP Smurf Attack

ICMP Smurf attackers are detailed in Chapter 12. This attack uses forged return addresses to flood a remote network (or host) with packets. UDP is also prone to this attack. In a UDP Smurf Attack, a forged packet is sent to a UDP server. The attacker forges the victim's network address as the packet sender. The server responds by sending one or more UDP packets to the victim. Although a few UDP packets will not significantly impact a network, thousands per second can cripple a network.

5. UDP Reconnaissance

Unlike TCP, UDP offers few options for system profiling and reconnaissance. Because UDP has no window size, sequence number, or header options, these cannot be used to profile the system.

UDP port scans rely on ICMP and packet replies. If no UDP service exists on a scanned port, then an ICMP “Destination unreachable” packet is returned. In contrast, some UDP services return replies to failed connection. Any UDP reply indicates an existing service. No reply indicates a service that received the packet and did not reply. The only way to defeat this type of port scan is to not return any ICMP packets. This makes no reply indistinguishable from no service.

2 **SSL**

In This Chapter

- SSL Functionality
- Certificates

2. SSL FUNCTIONALITY

Netscape initially developed SSL for securing Web-based connections. Without some type of encoding, HTTP would send Web traffic—including passwords and logins—in plaintext across the network.

In any protocol, cryptography leads to two critical issues. First, different countries have different laws regarding cryptography. Thirty-nine countries, including the United States, Russia, and Japan, have signed the Wassenaar Arrangement. This agreement regulates cryptographic software as munitions and tools of war. Non-Wassenaar countries have differing levels of cryptographic controls. For example, China and Israel have stricter regulations concerning the import, export, and usage of cryptographic systems. In contrast, South Africa, Chile, and Lebanon have virtually no restrictions. [RSA2000] Because different countries have different restrictions, any protocol that uses cryptography would fall under differing regulations, which hinders worldwide adoption.

Second, new and stronger algorithms are constantly being developed. If a protocol is tightly coupled to a specific algorithm, then it could quickly become outdated. Similarly, if a cryptographic algorithm were later shown to be weak, then a tightly coupled protocol would become a weak or insecure protocol.

SSL arose from these two requirements. SSL is not closely tied to any particular cryptographic algorithm. It allows a selection from different algorithms and permits new algorithms to be added. Because different systems support different algorithms, SSL provides the means to negotiate algorithms. SSL operates in three distinct phases: negotiation, key exchange, and data transfer.

1. SSL Connection

The first phase of the SSL protocol establishes network connectivity. Because SSL operates at the presentation layer, a network socket must be established through the lower layers. In particular, SSL operates independently of the transport layer, network addressing, and lower OSI layers. In addition, SSL operates above the session layer. Because the session layer manages disconnects and reconnects at the transport and network layers, these do not necessarily reset SSL connections.

To track the connection state, SSL maintains a *context*. The context is an abstract data structure that stores session, negotiation, and data transfer information. Initially the context is empty (set to default values). The SSL initialization handshake is used to populate the context with state and connection information.

2. SSL Negotiation

During the connection handshake, the SSL client and server negotiate cryptographic algorithms. The client transfers a list of supported cipher combinations. The list is transmitted in priority order—the first cipher is more desirable than the second cipher, the second is more desirable than the third, and so on. The server evaluates the cipher list and selects the first combination that is supported by the server. If no combination is viable, then there can be no SSL connection.

The cipher combinations specify four cryptographic elements: SSL version, key exchange, encryption cipher, and validation algorithm. The different available combinations form the list of available cipher sets. Table 19.1 shows a set of cipher combinations used by SSL. Other combinations, besides those in Table 19.1, are also possible.



Although SSL only provides a framework for negotiating and managing cryptographic systems, it is commonly described as an encryption system. This is because different versions of SSL are closely affiliated with specific cipher combinations called cipher suites.

TABLE 19.1 SSL Cipher Suites from OpenSSL 0.9.7e

SSL 2		
SSL2-DES-CBC-MD5	SSL2-DES-CBC3-MD5	SSL2-EXP-RC2-CBC-MD5
SSL2-EXP-RC4-MD5	SSL2-RC2-CBC-MD5	SSL2-RC4-64-MD5
SSL2-RC4-MD5		θ

SSL 3

SSL3-ADH-AES128-SHA	SSL3-ADH-AES256-SHA
SSL3-ADH-DES-CBC-SHA	SSL3-ADH-DES-CBC3-SHA
SSL3-ADH-RC4-MD5	SSL3-AES128-SHA
SSL3-AES256-SHA	SSL3-DES-CBC-SHA
SSL3-DES-CBC3-SHA	SSL3-DHE-DSS-AES128-SHA
SSL3-DHE-DSS-AES256-SHA	SSL3-DHE-DSS-RC4-SHA
SSL3-DHE-RSA-AES128-SHA	SSL3-DHE-RSA-AES256-SHA
SSL3-EDH-DSS-DES-CBC-SHA	SSL3-EDH-DSS-DES-CBC3-SHA
SSL3-EDH-RSA-DES-CBC-SHA	SSL3-EDH-RSA-DES-CBC3-SHA
SSL3-EXP-ADH-DES-CBC-SHA	SSL3-EXP-ADH-RC4-MD5
SSL3-EXP-DES-CBC-SHA	SSL3-EXP-EDH-DSS-DES-CBC-SHA
SSL3-EXP-EDH-RSA-DES-CBC-SHA	
SSL3-EXP-RC2-CBC-MD5	SSL3-EXP-RC4-MD5
SSL3-EXP1024-DES-CBC-SHA	SSL3-EXP1024-DHE-DSS-DES-CBC-SHA
SSL3-EXP1024-DHE-DSS-RC4-SHA	SSL3-EXP1024-RC2-CBC-MD5
SSL3-EXP1024-RC4-MD5	SSL3-EXP1024-RC4-SHA
SSL3-NULL-MD5	SSL3-NULL-SHA
SSL3-RC4-MD5	SSL3-RC4-SHA

1. SSL Version

There are many versions of the SSL protocol. Although SSLv1 (version 1) is seldom used, SSLv2 and SSLv3 are common. In addition, the *Transport Layer Security* (TLS) protocol may be used [RFC2246]. TLSv1 was designed as an open replacement for the patented SSL.

2. SSL Key Exchange and Authentication Algorithm

After establishing the connection and negotiating the cipher set, a key exchange is performed using the negotiated method. Many key exchange algorithms are supported by SSL. The most common are the following:

- Diffie-Hellman key exchange (DH)
- RSA (named after its authors: Rivest, Shamir, and Adleman)
- Digital Signature Algorithm (DSA) from the Digital Signature Standard (DSS)



A fourth system, called Fortezza, uses a digital token-based system for key exchanges, authentication, and encryption. Although it is supported by the SSL protocol, many SSL implementations ignore this option.

There are many variations for each of these key exchange systems. For example, the DH key exchange includes ephemeral Diffie-Hellman (abbreviated EDH or DHE) and anonymous Diffie-Hellman (ADH). DHE provides *forward secrecy*—every connection uses a new set of DH keys so a compromised connection cannot be used to crack previous connections. In contrast, ADH may not change keys between each connection. Both ADH and DHE are commonly used with digital signature algorithms such as RSA and DSS. ADH does not use a signature-based authentication system, which leaves it open to MitM attacks.

The DH key exchange is used for generating a shared secret value (see Chapter 4). In contrast, the RSA and DSS algorithms sign a digital challenge. Both parties can validate the challenge signature to validate the remote host. SSL systems may negotiate different DH variations as well as digital challenges. A client may propose key exchange algorithms such as “DHE with RSA,” “ADH with DSA,” or a la cart ciphers such as “ADH” or “RSA”

3. Encryption Cipher

In addition to the key exchange and authentication algorithm, the negotiated cipher set specifies an encryption cipher. The selection includes the cipher (DES, 3DES, RC2, RC4, and AES) and a choice of bit sizes and block chaining algorithms. For example, the client may specify “AES256” for 256-bit AES encryption or “DESCBC” for DES with CBC chaining (see Chapter 4 for chaining methods and encryption algorithms).

4. Null Cipher

SSL provides a method for negotiating ciphers, but it does not specify that ciphers need to be strong or secure. Insecure algorithms, such as ADH, are usually included in the cipher suite but may be disabled by default. A client requesting SSL3-ADH-DES-CBC-SHA will be unable to use this combination to connect to a server if ADH is not enabled.

The *null cipher* is another example of an insecure option. This cipher performs no authentication and no encryption—data is transmitted unencoded. The null cipher is usually used for testing and debugging; it is not intended for secure communication. Network programmers may enable and use the null cipher along with a packet sniffer (such as Wireshark, `snort`, or `tcpdump`) to diagnose network communication problems. Without the null cipher, packet sniffers would be unable to identify network corruption because encryption prevents viewing application data.

Although the null cipher is not secure, it is available for use by most SSL implementations. OpenSSL (<http://www.openssl.org/>), the GNU Transport Layer Security Library (<http://www.gnu.org/software/gnutls/>), and Microsoft's SSL all include the null cipher option. Fortunately, the null cipher is usually disabled by default. It takes a conscious effort to enable the null cipher for use with SSL, and it must be enabled on both ends of the SSL connection. Unfortunately, developers occasionally forget to disable the null cipher after testing. Potentially, a client and server could negotiate and use the null cipher for transferring sensitive information.

19.2.2.5 Validation Cipher

Although key exchanges provide authentication, and encryption offers privacy, neither explicitly performs integrity checking. Although it is unlikely that a corrupt packet would pass decryption and authentication, data validation is not explicitly provided. For explicit validation of data integrity, SSL supports MD5 and SHA1 digests.

Although in theory every combination of key exchange, encryption algorithm, and validation cipher is possible, some variations are never implemented. For example, the DSS algorithm explicitly specifies use of the SHA1 digest. Although DSS could be used with MD5, the specification of DSS explicitly restricts the validation algorithm.

19.2.3 SSL Key Exchange

During the initialization handshake, the client transmits a list of desirable ciphers and a *digital challenge*. The challenge is used to initialize the selected key exchange. The server signs the challenge and the client validates the signature.

SSL may also transfer certificates. The X.509 certificate contains information that can be validated by a third party (section 19.3 discusses certificates). The RSA and DSS algorithms can validate certificates.

The *key exchange* permits authentication between the client and server. This mitigates the threat from a MitM attack. When using certificates or preshared keys, the exchange also validates the identity of the client and server. Although the key exchanges are performed using asymmetrical algorithms, the authenticated connection can be used to transfer a seed for use with a symmetrical (and faster) encryption algorithm. For example, SSL3-DHE-DSS-AES256-SHA specifies using SSLv3 to communicate. DHE is used to initially exchange a shared secret key, and DSS validates the client and server. After validation, both ends begin using the symmetrical 256-bit AES algorithm for encrypting data transfers. Throughout this entire process, SHA1 is used for validating packet contents.

4. SSL Data Transfer

Following the algorithm negotiation and key exchange, both ends may transfer application data. Information from the OSI application layer is encrypted using the negotiated algorithm (e.g., DES or AES) and passed down the OSI stack. On the receiving end, the SSL protocol receives the encrypted data from the OSI session layer, decrypts the data, and passes the information up to the application layer.

5. SSL Communication Flow

The different negotiations used by SSL, including cryptographic algorithms, key exchanges, and certificates, are performed concurrently (Figure 19.1).

Step 1: The client sends a `hello` record that includes the version of SSL being used (usually TLS 1.0), a list of viable cipher specifications (see Table 19.1), and a random challenge.

Step 2: The server responds with the selected cipher set and either a server certificate (*server-side certificate*) or part of a key exchange. The server may also request a certificate from the client (*client-side certificate*). The server generates a response to the random challenge that can be validated by the client. This allows the client to identify the server—preventing potential MitM attacks between the client and the server. This does not validate the server; it only prevents a MitM from hijacking the established connection.

Step 3: If the server provides a certificate, then the client contacts a certificate authority (CA) and validates the certificate. Although it is considered secure for the client to validate the certificate, there is nothing in SSL requiring certificate validation. Contacting a CA can add time to the initial connection. If speed is an issue, then the client may choose to not validate the server's certificate. In addition, some cipher sets, such as SSL3-NUL-MD5 and SSL3-RC4-MD5, do not require a server certificate.

Step 4: The client responds to the server, initiating a key exchange. If a client-side certificate is available, then the client may provide it. (If the server requests a client-side certificate, then one must be provided.) If the server provides a certificate, then the client also provides a message encrypted with the certificate. Similar to Step 2, this allows the server to identify but not validate the client.

Step 5: If the client provides a certificate to the server, then the server may contact a CA to authenticate the client. As with Step 3, the server is not required to validate the client, but servers usually validate client-side certificates.

Step 6: The server sends an encrypted handshake to complete the validation process. At this point, the negotiations and exchanges are complete; the client

and server are both authenticated and validated. An attacker cannot act as a MitM or impersonate the client and server. All further communication uses the agreed upon symmetrical encryption algorithm, which is seeded with the negotiated key exchange values.

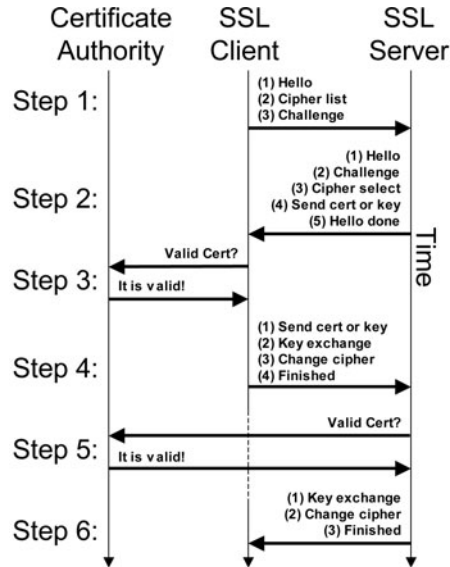


FIGURE 19.1 SSL initialization flow between client and server.

SSL includes a few out-of-band signals for managing the encrypted tunnel. These include the `hello` record for initiating the handshake, a `cipher change` signal to begin using symmetrical encryption, and a `finished` signal to denote the end of an SSL connection. RFC2246 includes a few other signals for managing retransmissions, failures, and renegotiations.

19.3 CERTIFICATES

Although many systems employ key exchange technologies to mitigate MitM hijacking, SSL's strength comes from endpoint validation. Through the use of digital certificates, the client can authenticate the server and vice versa. SSL supports two types of certificates. Server-side certificates are stored on the server, and client-side certificates are stored on client. Although there is support for different certificate formats, SSLv1 and TLS currently use the X.509 certificate format.



Certificates are sometimes called certs as a shorthand notation. A cert is a certificate.

NOTE

1. X.509 Certificates

Each *X.509 certificate* follows a well-defined format and provides information for identification and validation [RFC2459]. Each X.509 certificate defines three components: identification, public key, and private key.

1. X.509 Identification

The X.509 identification contains field-value pairs that provide information about the certificate. The data is stored in the ASN.1 format. Some of the certificate fields form a *distinguished name* (DN) that identifies the certificate. Some of the common DN attributes are listed in Table 19.2. For example, the DN for HTTPS on Amazon.com is

```
/C=US/ST=Washington/L=Seattle/O=Amazon.com Inc./CN=www.amazon.com
```

The identification also includes the issuer's information. For Amazon.com, this is

```
/C=US/O=RSA Data Security, Inc./OU=Secure Server Certification Authority
```

TABLE 19.2 Common Distinguished Name Attributes

Field Name	Purpose
CN	Common name (usually a person or server)
O	Organization (e.g., company name)
OU	Organization unit (e.g., department in a company)
C	Two-digit country code
ST	State or province
L	Locality (e.g., city or address)
Email	Email address for contact



Not every attribute in the distinguished name is required. In addition, attributes can be repeated. For example, a certificate may include multiple organization unit (OU) fields.

NOTE

When the certificate is passed from the server to the client, the client can use the DN to validate the server's certificate with a CA. Together with the public key, the CA can validate that the key belongs with the DN.

Some clients may also use the DN to validate the server. For example, if the common name (CN) is www.amazon.com, but the certificate comes from www.badguy.com, then the certificate may be forged or stolen. In general, certificates should only come from the correct domain.

19.3.1.2 X.509 Public and Private Keys

SSL uses the X.509 certificate as a container for transferring keys in a *Public Key Infrastructure* (PKI). The PKI defines a set of public and private keys. The public X.509 certificate is used to transport the identification and public key across the network, whereas the private key is not shared. During the key exchange, a message is encoded with the private key and decoded with the public key to authenticate each party.



If the system only uses server-side certificates, then only one side can be validated. A basic key exchange prevents hijacking established SSL connections but does not authenticate the client.

The two main certificate validation algorithms are RSA and DSS. In RSA, the key exchange encodes a message with the private key and decodes it with the public key to authenticate each party. DSS transmits a digest of the public key rather than the key itself. When compared with RSA, DSS further impairs an attacker by preventing a direct attack against the public key.

19.3.2 Server Certificates

Server-side certificates are used to validate the server to the client. The server sends a public key (or in the case of DSS, a public key digest) to the client. It is up to the client to validate the certificate. Validation is usually done in one of two automated ways. The client may already have a copy of the server's key. This copy can be used to validate the server. More often, the client consults a trusted third-party that can validate the certificate. A trusted *certificate authority* (CA) is usually not the same host as the system being validated, so it can validate the certification.

Some domains operate their own CA servers for validating their certificates. This is problematic because it allows attackers to validate their own certificates. For example, VeriSign runs its own CA servers. If an attacker compromises the verisign.com DNS entry, then there is no way for a client to validate that VeriSign's certificate is authentic.

A third option is commonly used by interactive systems, such as Web browsers using HTTPS. If the certificate cannot be verified automatically, then the user is prompted to validate the certificate. Unfortunately, most users do not manually verify certificates—users usually just select “**Y**” or “**K**” to continue. The desire to access the Web site is usually stronger than the perceived threat of an attack. As a result, interactive prompting for certificates usually leads to risks and high potentials for exploitation.

3. Client Certificates

Client-side certificates are used to authenticate the client to the server. Very secure systems use these to ensure that only authorized clients access the OSI application layer. Failure to provide a required, valid client-side certificate blocks the SSL connection. In addition, if both client and server use certificates, then it alleviates the threat of a MitM attack.

Without client-side certificates, the client is effectively anonymous to the server. In lieu of certificates, client-side identification may be passed to the application layer in the form of a username and password. For example, most online banks only use server-side certificates that authenticate the bank to the client. For client-side authentication, banks rely on the application layer, which is usually in the form of a username and password that is transmitted over the established SSL connection.

In many cases, client-side certificates are not necessary. Using only a server-side certificate, the client can be sure that the server is not an imposter, and an established SSL connection is not vulnerable to session hijacking.

Although a trusted third party usually validates server-side certificates, the server usually authenticates client-side certificates. Each client-side certificate is generated by the server and distributed before the first client connection. By authenticating certificates locally, the server gains a speed improvement over querying a remote system for authentication. In this situation, the server acts as its own CA.

4. Certificate Authority (CA)

A CA is a system that can validate a certificate. Although the CA server may be hosted by the certificate’s domain (or on the same server as the certificate), they are usually hosted on separate systems or in remote domains. The system using the CA server for validation trusts the results from the CA server. For example, a client that sends a server-side certificate to the CA server trusts that the CA server will properly validate the certificate.

To prevent a hijacker from impersonating a CA server, most servers pre-share their own public keys. A client validating a server-side certificate will encode the

certificate with the CA server's public key. A domain-name hijacker will be unable to decode the request and cannot reply to the client.

A trusted CA server may perform any of three actions:

Accept: The certificate is found to be valid. The CA server responds with an acceptance.

Reject: The certificate is invalid, and the submitter is asked to reject the certificate. This usually happens for expired certificates, but invalid certificates can also lead to rejections.

Revoke: In the event that a certificate is compromised, the CA server can be ordered to revoke a certificate. Checking for revocation is very important because it is the only way to distinguish a valid and active certificate from a compromised certificate.



Certificates contain start and end dates that specify when they are valid. Although a client can check these dates, a client without a sense of time (or with an unset clock) may pass expired certificates to a CA server. Expired certificates generate revocation replies.

19.3.5 Certificate Generation

To validate a certificate, the CA server must know about the private certificate that contains the private key. This is a multistep process. First, the server generates a public and private key pair. For example, this can be done with OpenSSL using

```
openssl genrsa -des3 2048 -out key
```

OpenSSL will prompt for a pass phrase for protecting the private key. This deters attackers that manage to compromise the system and copy key files.

After generating system keys, a *certificate-signing request* (CSR) is generated and passed to the CA server:

```
openssl req -new -key key -out key.csr
```

The CSR (`key.csr`) contains a public key and is sent to the CA server. The CA server uses the CSR to encode an identity certificate. This identity certificate is used by the SSL certificate exchange to authenticate the server. Although this process is indirect—requiring a key pair, CSR, and identity certificate—it prevents an attacker from hijacking the initial key generation and distribution.

2 SSH

In This Chapter

- SSH and Security
- The SSH Protocol

1. SSH AND SECURITY

The SSH protocol addresses each of the basic security concepts (see Chapter 1):

Confidentiality: Each SSH connection is encrypted, preventing an eavesdropper from viewing information. For added security, SSH periodically re-exchanges keys to ensure that a compromise of one set of keys does not compromise the entire session. In contrast, CTCP, IPsec, and IPv6 only exchange keys at the beginning of the connection.

Authentication: Before establishing a connection, the client must authenticate the server *and* the server must authenticate the client. Client authentication can be any combination of certificates (keys), passwords, or digital tokens. Although SSH is usually used with one-part authentication (a password or a key), it can support two- and three-part authentication systems. The server only uses a certificate to authenticate with the client.



There are three types of keys with SSH. Cryptographic keys are used to seed cryptographic algorithms, host keys are used to authenticate SSH servers, and client keys are asymmetrical keys used as digital certificates for authentication.

Authorization: SSH limits the traffic that can enter the tunnel and can restrict how data exits the tunnel. For remote login access, SSH restricts authorization to the user's login privileges.

Integrity: SSH uses encryption and cryptographic checksums to validate each packet. Any packet that fails an integrity check is viewed as an attack, which terminates the connection.

Nonrepudiation: Each packet is cryptographically signed using an HMAC, ensuring that the data actually came from the sender.

SSH has three primary uses: establish a secure network tunnel, provide a VPN with port-forwarding characteristics, and supply application-layer login functionality.

2. THE SSH PROTOCOL

The SSH protocol operates in five phases: algorithm negotiation, key exchange, server authentication, client authentication, and data transfer.

1. Phase 1: Algorithm Negotiation

As with SSL, SSH supports a variety of cryptographic algorithms. Upon the initial connection, the systems negotiate the cryptographic algorithms that will be used. Different SSH versions and platforms support different algorithms. For encryption, 3DES is required for default compatibility between SSH clients and servers. Other algorithms, such as DES, IDEA, CAST, AES, and Blowfish, are optional. SSH authentication may use RSA or DSA keys. The integrity algorithm depends on the version of SSH. The older version, SSHv1 (SSH version 1), uses MD5 for performing integrity checks. SSHv2 uses SHA1.

The SSH algorithm negotiation is very similar to SSL. The client offers a list of supported cipher combinations, and the server responds with its selection. There are some significant differences between the negotiations for SSL and SSH:

Combinations: Unlike SSL, where cipher sets are predefined, SSH keeps them separate. Any combination of cryptographic key exchange, host-key (certificate) exchange, encryption, MAC, and compression algorithms are supported. Although SSL only supports DSS certificates with MD5 checksums (as defined by the DSS specification), SSH can support DSS with MD5 or SHA1.

Weak Ciphers: Some SSL cipher sets include weak ciphers that are generally undesirable. They are included because they are part of predefined, standard cipher sets. SSL clients, such as Web browsers, may warn the user when weak ciphers are selected. In contrast, SSH simply does not support weak algorithms. If an algorithm is considered weak, then it is not offered as an option.

Bidirectional Negotiation: With SSL and SSH, the client offers a list of ciphers, and the server makes a selection; however, SSH follows this with a list of ciphers from the server, and the client makes a selection. It becomes very possible for the negotiation to result in two different sets of algorithms. The client may use 128-bit AES when transmitting to the server, and the server may select 3DES when responding to the client. A cryptanalysis attack against the traffic may require cracking two completely different algorithms.

2. Phase 2: Key Negotiation

The algorithm negotiation is immediately followed with a key exchange using the Diffie-Hellman (DH) algorithm. SSH performs key exchanges periodically—usually every hour. This way, if an attacker does manage to identify the keys, only a one-hour window of the conversation is compromised. Considering that SSH uses a large key size (512 bits or larger), an attacker is unlikely to crack any session within a usable timeframe.

3. Phase 3: Server Authentication

After completing the key exchange, all information is transmitted using encryption. The first data transmitted is the SSH server's public key. If the server has never been seen before, then the client prompts the user to validate the key. After validation, the server's key is added to the client's list of known servers. However, if the server *has* been seen before, then the client compares the key with the cached version. If the cache does not match, then the client usually aborts—unlike HTTPS (Chapter 19), the SSH user is not prompted to continue when the server fails to validate.

Most SSH clients associate server keys with network addresses. Unfortunately, servers are occasionally reinstalled or assigned new addresses. The result is a mismatch between the server's key and the client's cache. The user must resolve the issue, and the resolution varies with different SSH clients:

OpenSSH: The OpenSSH client (<http://www.openssh.org/>) is used on most Unix systems. For OpenSSH, users must manually edit their host-key cache files to remove the mismatched server entry. The file is stored in the user's home directory: `$HOME/.ssh/known_hosts`. For users who connect to many SSH servers, this can be a very large file. To resolve mismatched cache entries, users must edit the file and delete the line for the offending server. This is significantly different from HTTPS, where most browsers prompt users to continue.

PuTTY: For Windows SSH clients, PuTTY (<http://www.putty.nl/>) is common. PuTTY stores the host-key cache in the Registry under `HKEY_CURRENT_USER\software\SimonTatham\PUTTY`. But rather than editing the Registry, PuTTY prompts the user when the host key does not match. The user has the option to accept the new host key and overwrite the old one.

Mocha Telnet: For PocketPC, WindowsCE, PalmOS, mobile phones, and most other platforms, Mocha Telnet (<http://www.mochasoft.dk/>) is available for supporting SSH. Unlike OpenSSH and PuTTY, Mocha Telnet does not cache server keys. As a result, all servers are always accepted.

Other SSH clients, such as F-Secure SSH, ZOC, and Pragma SSH all use different caching methods. Resolving mismatched keys requires knowing the specific type of SSH client.

4. Phase 4: Client Authentication

After authenticating the server, the client is authenticated. The client may transmit any combination of a password, a challenge signed by the client key, or a digital identifier for authentication. If the authentication works, then the client successfully connects. If the authentication fails, however, the client is disconnected. SSH

servers may permit multiple password retries, but repeated failures result in a disconnection.

20.2.5 Phase 5: Data Transfer

After negotiating the algorithms, exchanging keys, and authenticating both the client and server, an encrypted VPN tunnel is established. The SSH VPN permits multiple channels—similar to the transport layer's ports. Through concurrent channels, SSH supports many simultaneous connections. For example, a login shell may be on channel #0, an X11 graphical traffic on channel #1, and a VPN proxy on channel #2. All of this data transfers across an encrypted tunnel that changes keys every few minutes.

2 SMTP

In This Chapter

- Email Goals
- Common Servers

2. **EMAIL GOALS**

SMTP was designed for the timely, but not immediate, delivery of text messages. This is different and distinct from other communication protocols. For example, IRC and Jabber provide real-time communication. All intended recipients see each message immediately. In contrast, email may take a few seconds, minutes, or even hours to be delivered. SMTP was also not designed for transferring binary data—attachments and binary file support were added later.

The initial Internet was designed for robustness. Nearing the end of the Cold War, the network was initially designed for redundancy so that a single nuclear strike would not disable the entire network. SMTP followed this same design: a single network failure would not prevent email delivery. Instead, email messages are

passed host-to-host until it reaches the destination. As long as a path exists, email will be delivered.

Besides timely and robust delivery, SMTP was designed to be extendable. This flexible protocol can be easily extended to support most data formats, encryption systems, and some authentication.

1. SMTP Data Format

SMTP uses a simple file transfer protocol to deliver email. Each email message is a file, and each file contains three components: meta header, blank line, and content. The meta header consists of `field: value` pairs. These are used to identify recipients, subject, and routing information. These header entries are used for tracing emails and identifying delivery information. A single blank line is used to separate the header from the content.

Email content is defined as printable ASCII characters. Other types of content, such as binary files and non-English languages, must be encoded in printable ASCII. Formats, such as Base64 or MIME encoding [RFC2045—RFC2049] are commonly used. Similarly, content originally contained one text block. MIME formatting permits subdivisions of the content to allow a single email to contain multiple attachments.

2. SMTP Command Structure

SMTP defines a command and response environment for communicating between a *mail user agent* (MUA—mail client) and *message transport agent* (MTA—mail server). The MUA connects to the MTA and issues a set of commands based on the information in the data's email headers. Each of these commands (Table 22.1) is well defined and must be supported by an SMTP-compatible system.

TABLE 22.1 SMTP Commands

Command	Example	Purpose
HELO <i>domain</i>	HELO roach.1an	Tells the MTA the domain for sender's information
EHLO <i>domain</i>	EHLO roach.1an	Similar to HELO but specifies ESMTP support
MAIL FROM: <i>address</i>	MAIL FROM: <u>santa@pole.org</u>	Specifies the email sender address
RCPT TO: <i>address</i>	RCPT TO: <u>grinch@xmas.mil</u>	Specifies one recipient for the email

Command	Example	Purpose
DATA	DATA	Specifies the start of the email content
QUIT	QUIT	Ends the SMTP/ESMTP connection

Some commands are intended to be used in a specific order. For example, **HELO** (or **EHLO**) must be the first command. And there can be only one **MAIL FROM** per email. Other commands may be used repeatedly; **RCPT TO** may be repeatedly provided for delivery to many users.

The **DATA** command is the only multiline command. After issuing a **DATA** command, all subsequent bytes form the SMTP meta header and content. A period (.) on a line by itself indicates the end of the data.

22.2.2.1 MTA Return Codes

Each MUA sends a command to the server, and the server responds with an acknowledgement. The return codes follow a format originally defined for FTP (RFC114). Each code has a three-digit identifier. The first digit tells the class of code. The second identifies the response category (Table 22.2), and the remaining digit tells the exact code type. A text string that describes the code in a human-readable form follows each code. For example, after the MUA sends a **HELO**, the MTA may respond with **250 hostname Hello host [192.168.1.5], pleased to meet you**. The code 250 indicates a positive completion and no syntax issues.

TABLE 22.2 MTA Return Codes Classes

Code	Purpose
1xx	Positive Preliminary reply
2xx	Positive Completion reply
3xx	Positive Intermediate reply
4xx	Transient Negative Completion reply
5xx	Permanent Negative Completion reply
x0x	Syntax issues
x1x	Informational response
x2x	Connection-oriented reply

Code	Purpose
x3x	Unused
x4x	Unused
x5x	Mail system

22.2.2.2 Sending Email

To send email, the MUA must connect to the MTA and transmit a series of commands. After each command, the MTA provides a status reply. Figure 22.1 shows a sample SMTP command and response session. In this example, Telnet is used as the MUA. Figure 22.2 shows the email generated by this example.

```
% telnet rudo1f 25
Trying...
Connected to rudo1f.npole.org.
Escape character is '^]'.
220 rudo1f.npole.org ESMTp Sendmail 8.8.6 (PHNE_17135)/8.7.3 SMKit7.1.1
hp hp; Thu, 25 Apr 2002 09:17:17 -0600 (MDT)
helo ranch.npole.org
250 rudo1f.npole.org Hello ranch.npole.org [10.2.241.27], pleased to
meet you
mail from: santa@npole.org
250 santa@npole.org... Sender ok
rcpt to: grinch@xmas.mil
250 grinch@xmas.mil... Recipient ok
data
354 Enter mail, end with "." on a line by itself
Subject: Ho ho ho

I know you've been a bad boy.
.
250 JAA19237 Message accepted for delivery
quit
221 rudo1f.npole.org closing connection
Connection closed by foreign host.
```

FIGURE 22.1 Sample SMTP session with MUA information (in bold).

Many of the SMTP headers are specified by default. If the MUA does not provide the headers in the `DATA`, then the values are filled in by the MTA. These include the `Date` and `Message-ID` headers. However, the MUA may specify these values. Just

```

Received: from exchange2.npole.org ([10.8.16.2]) by exchange.npole.org
with
    SMTP (Microsoft Exchange Internet Mail Service Version 5.5.2653.13)
    id J2DYC3G1; Thu, 25 Apr 2002 08:19:32 -0700
Received: from rudolf.npole.org (rudolf.npole.org [10.2.23.20])
    by exchange2.npole.org (Postfix) with ESMTMP id 97BB3C0093F
    for <grinch@xmas.mil>; Thu, 25 Apr 2002 08:19:32 -0700 (PDT)
Received: from ranch.npole.org (IDENT:santa@npole.org [10.2.241.27])
    by rudolf.npole.org with SMTP (8.8.6 (PHNE_17135)/8.7.3 SMKit7.1.1
    hp hp) id JAA19237 for grinch@xmas.mil;
    Thu, 25 Apr 2002 09:17:31 -0600 (MDT)
Date: Thu, 25 Apr 2002 09:17:31 -0600 (MDT)
From: santa@npole.org
Subject: Ho ho ho
Message-Id: <200204251517.JAA19237@rudolf.npole.org>
To: grinch@xmas.mil

```

I know you've been a bad boy.

FIGURE 22.2 Sample email from SMTPsession.

as the MUA specifies the `subject`, other fields such as the `Date` and `Message-ID` may also be provided.

22.2.2.3 Command Exploitation

Many of the MTA commands can leak information, and attackers can use this information to identify potential weaknesses. For example, upon connecting to the MTA, an initial return code is displayed. Many MTAs include the exact version number and patch level of the email server. An attacker can view this information and readily identify possible vulnerabilities.

Some commands are required, and others are optional. For example, `MAIL FROM`, `RCPT TO`, and `DATA` are required for sending email, but the `HELP` command is optional. `HELP` is commonly used to display the list of available commands. Just as the initial connection lists version information, the `HELP` command may also include the MTA version and patch level. Even when the version information is not available, the list of available commands may sometimes identify the type of server (Figure 22.3).

Other optional commands, such as `VERFY` and `EXPN`, are used to verify and expand email addresses. Attackers can use these to scan the host for a list of email addresses. Between 1995 and 2003, many unsolicited mailing lists were generated by repeatedly calling `VERFY` and `EXPN` with common account names. The replies were used to generate lists of valid email addresses.

```

$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 rudolf.npole.org ESMTP Sendmail 8.11.6/8.11.6; Sun, 15 Jan 2006
09:51:22 -0700
HELP
214-2.0.0 This is sendmail version 8.11.6
214-2.0.0 Topics:
214-2.0.0      HELO      EHLO      MAIL      RCPT      DATA
214-2.0.0      RSET      NOOP      QUIT      HELP      VRFY
214-2.0.0      EXPN      VERB      ETRN      DSN      AUTH
214-2.0.0      STARTTTL
214-2.0.0 For more info use "HELP <topic>".
214-2.0.0 To report bugs in the implementation send email to
214-2.0.0      sendmail-bugs@sendmail.org.
214-2.0.0 For local information send email to Postmaster at your site.
214 2.0.0 End of HELP info

```

FIGURE 22.3 MTA version information in the initial connection and HELP menu.

Secure mail systems are usually modified to not list version information at the initial connections or in the `HELP` reply. Most production MTAs do not provide any of the optional commands.



Some MTAs are intentionally configured to list incorrect mailer versions. This usually requires modification of the MTA configuration files. The belief is that false information is more damaging than no information. An attacker who observes false information is likely to believe it is real.

22.3 COMMON SERVERS

There are hundreds of MTA providers and implementations. Some handle general mail delivery, whereas others specialize in mailing lists, network types, or additional features (e.g., calendars and shared workgroup space). Although there are many MTA options, most mail servers on the Internet either use Sendmail, Microsoft Exchange, Qmail, or Postfix [Timme2004, Credentia2003]. Each of these mailer options has tradeoffs with regards to performance, features, and security.

1. Sendmail

The Sendmail MTA (<http://www.sendmail.org/>) is the single most common mail transport agent. It is open source and provided as the default MTA distribution for most Unix systems. Sendmail is a master email daemon; it receives, processes, forwards, and delivers email.

Sendmail is also one of the oldest MTA implementations—it dates back to 1979 as the program `delivermail`. Being old and widely used is a strong benefit for any application. It implies durability, extendibility, supportability, and overall usefulness. However, Sendmail does have significant limitations. For example, Sendmail consistently benchmarks as one of the slowest MTAs. The configuration file is very complex, and Sendmail is historically known as the second most insecure network applications in the Internet. (Sendmail is second only to DNS, and Sendmail uses DNS.) Between 1993 and 2003, there were 65 software releases, averaging one release every 2 months. Many of these releases address security issues.

Sendmail is readily identifiable by the SMTP welcome message, `HELP` file identifier, and version information placed in the email header. Attackers can readily identify Sendmail, including the version and patch level. This allows an attacker to identify all unpatched risks. For mitigation, administrators should consider changing or removing the Sendmail identification strings and regularly check for updates and patches.

2. Microsoft Exchange

Similar to Sendmail, the Microsoft Exchange server is a monolithic server designed as an alternative to SMTP. Within the MTA, the system uses the proprietary *Exchange* data format, but it also supports communicating with standard SMTP MTAs. From a security viewpoint, Exchange is not as vulnerable as Sendmail. The total number of exploits for Exchange is significantly less than Sendmail, but risks to Sendmail are patched rapidly—usually in days. In contrast, Exchange is patched much less frequently and Microsoft sometimes downplays risks. Some vulnerabilities with Exchange have been unpatched for months (or longer), and most patches are released several months after the vulnerability's initial discovery.

3. Qmail

Qmail (<http://www.qmail.org/>) is intended as a replacement for Sendmail. It was designed for performance, reliability, simplicity, and modularity. As a result, Qmail is very fast and relatively easy to configure and maintain. Because of its impressive performance, it is widely used by systems that manage large mailing lists and high volume traffic.

Security was a primary design goal for Qmail. The MTA defines a clear separation between files and programs, limits code that runs as root (Unix administrative privileges), and implements defense-in-depth. As a result, there have only been two known exploits for Qmail since 1997, and both were patched shortly after identification.

22.3.4 Postfix

Postfix (<http://www.postfix.org/>) began as an alternative to Sendmail. In 1998, Web-based application servers were modularly designed to handle high loads and limit the impact from security exploits. Wietse Venema adapted these Web-based techniques to email delivery. The result is Postfix: a fast, efficient, flexible, modular, and secure MTA.

Although fewer systems use Postfix than Sendmail or Microsoft Exchange, Postfix is the fastest growing MTA. In 2003, less than 3 percent of MTAs were using Postfix. By 2005, Postfix had grown to over 20 percent. Postfix is currently included by default on some Linux and BSD distributions.

Postfix employs dozens of special-purpose applications but maintains the same command-line options as Sendmail. This makes it a viable drop-in replacement for the more common Sendmail system. Postfix does not have the same security issues as Sendmail. Instead, it implements many security-oriented concepts:

Least privileged: Each module runs with the minimum permissions necessary.

Insulation: Each module performs one task independently from other modules.

Controlled environment: No modules operate as other users.

No SUID: No Postfix module runs with “`setuser id`” access. In Unix, SUID is used to change access privileges.

Trust: The concept of trust between modules is well defined and constantly validated.

Large Inputs: Application memory is properly handled to mitigate risks from overflows.

From a security perspective, Postfix is secure upon installation. It does not disclose version information and consciously implements concepts for secure applications. As a result, Postfix has only had one reported exploit in eight years, and it was a DoS attack.

3 **Security**

In This Chapter

- Threat Models
- Concepts
- Common Mitigation Methods

2. THREAT MODELS

Determining likely attack vectors requires understanding the probable threat models. Although attacks may come from anywhere, the environment dictates likely directions that an attacker may explore. Attacks may be internal or external to a system, and intentional or accidental. In addition, different attackers have different motivations. By understanding the likely vectors and possible motivations, an acceptable solution can be identified to address the security needs, and resources will not be wasted on unlikely scenarios.

1. Internal versus External



When people think of computer security, they usually think of external attackers—people (or processes) that do not belong in the environment. They gain entry through portals that link to the “outside” and compromise systems by turning internal information into external information. Unfortunately, both internal and external attackers equally target systems. Surveys conducted by the CSI Institute and FBI have shown that the risk from internal attackers (insiders) is nearly equal to the risk from external attackers. In dollar value, however, the damage caused by insiders is significantly greater than the damage from most external attacks. Internal people know your system’s inner workings and how to best attack it.

1.2.1.1 Internal Attacker Motivation

Hostile insiders are usually disgruntled employees or corporate spies. Although occasionally a competitor or organized gang plants an employee into a company, more often the employee is already present and becomes motivated to attack the company from the inside. Although the motivational factors are incident specific, they usually include one of the following:

Personal issues: Personal problems may provide motivation. A disagreement with a boss or coworker, or general frustration, may trigger an attack.

Unfair disadvantage: The employee may feel mistreated by the company and view his insider access as a way to fight back.

Greed: An employee may see value in selling insider access to an interested external party. For example, in 2004, an insider at America Online (AOL) sold 92 million customer e-mail addresses to spammers [Krim2004].

Curiosity: Although not necessarily malicious, the employee's curiosity and exploration of the company's internals may create problems.

Ignorance: The employee may not be aware that specific information should be confidential.

Each of these motivating factors may cause significant damage. Although corporate policies set the groundwork for legal reactions and employee education, they are generally not a deterrent to insiders. For example, an employee that feels mistreated is probably not concerned about being fired. And user education is unlikely to prevent an intern from placing sensitive information in their résumé.

The "No Touch" Audit

In 2005, a large corporation hired an external auditor to scan for networking weaknesses. This style of auditing begins with identifying and mapping the corporate network. Unfortunately, the hiring department was not the same as the computer support division. This developed into an internal department war within the company. The result was an audit restriction: conduct the network survey without touching any system owned by the corporation. This restriction even forbade the auditors from accessing the company's public Web site.

Fortunately for the auditors, the company's employees and business partners leaked a significant amount of public information. Through the use of press releases, public white papers released by the corporation's technology partners, employee online résumés, and postings in public newsgroups, the auditors created a very detailed map of the company's network. The map

included the network architecture, router specifications, type and quantity of servers, operating system versions, and specific applications used by the company. Even network policies such as scheduled backups and downtimes were identified. The audit demonstrated the severity of innocent information leaks from insiders: the corporate network was vulnerable.

1.2.1.2 Internal Attacker Damage

The damage caused by insiders can range from theft to vandalism and from information leakage to destruction. Theft and vandalism are generally associated with the corporate catch phrases “we trust people here” and “people get fired for that.” But, there is a deeper level of importance. By the time the problem is caught, the damage is done: expensive hardware is missing or destroyed, data and privacy is lost, and time must be spent identifying and recreating damaged information. In the case of industrial espionage, risk assessments must occur, and project goals usually require significant adjustments.

The simplest way to protect against theft and vandalism is to take basic precautions: monitor the systems for abnormal downtime, lock down expensive equipment, place main servers behind locked doors, and use video cameras. Even with these precautions, a determined thief will find a way to access data and hardware. Store valuable data in redundant locations, lock away backup tapes and CD-ROMs, and maintain redundant or compatible equipment as short-term emergency replacements.



Keeping all of this in mind, there are few things that can stop a determined insider with a screwdriver. Detection is as important as prevention, if not more important.

Information leakage and destruction are more difficult to control. Backup and usage policies may limit any long-term impact, but they do not provide short-term solutions. Preventative measures such as access restrictions and internal IDSs limit the overall potential impact. Access restrictions reduce the likelihood of an attack, whereas an internal IDS provides rapid notification of a problem.

The security of the corporation begins with the security of the network. Although not a complete solution, securing the corporate network can mitigate many of the risks posed by insiders. Security options include designing secure solutions, hardening systems and networks, and monitoring for potential risks.

1.2.1.3 External Attacker Motivation

External attackers generally have many more motivating factors than internal attackers. Although external attacks share many of the same motivations as internal attackers, there may be other driving forces:

Political: The attack may be used to make a statement.

Status: An attacker may use the attack as bragging rights or to demonstrate his skills.

Power: An attacker may use the attack to show his technical superiority.

Any list of motivating factors for external attackers is only partial; anything may be a driving factor. In some cases, the attacker may want “a system” —the selection may be arbitrary based on general criteria such as features or accessibility. In other cases, the attack may be specific; the attacker doesn’t want *a* system, he wants *your* system.

The primary mitigation options for external attackers revolve around network security. By preventing unauthorized access to network systems, a company can reduce the likelihood of a successful attack. Networks with adequate redundancy and fail-over detection can withstand even large-scale network attacks.

More Power!

The need for computing resources can be a significant driving factor. An attacker may have heard of a new exploit but lack the resources to test the exploit. Without access to the necessary resources, the attacker turns to someone else that has the resources. This type of “show me” attack generally happens shortly after a new exploit is made public.

The infamous distributed denial-of-service (DDoS), where many computers attack a single computer on the network, was likely developed by rival users in online chat rooms (IRC—Internet relay chat). The motivation is direct: more computers yield more resources for the attack. In February 2001, a 15-year-old Canadian called “Mafiaboy” conducted DDoS attacks against online powerhouses including Yahoo!, eBay, Amazon, CNN, ZDNet, and e*trade.

Although exploits and network attacks are generally malicious, resources may also serve nonmalicious purposes. For example, in 1998, William Aaron Blosser used spare computer cycles at US West to compute prime numbers [Blosser1998]. Although not malicious, this did result in an abuse of network resources.

1.2.1.4 Bridging Internal and External Attackers

In some cases, external attackers solicit information or access from internal sources. When this partnership happens, the result can be disastrous. In May 2005, insiders at Bank of America and Sumimoto Bank assisted external attackers in compromising bank accounts and stealing money. Bank of America identified 670,000 compromised bank accounts [Reconnex2005], and the thieves stole £220 million from Sumitomo Bank [Scott2005].

The combined problem of internal and external attackers can only be mitigated through a combined defense against internal and external attackers. In this situation, the network is one of the few common venues.

1.2.2 Intentional versus Accidental

Although intentional attacks receive significant attention, accidental attacks are very common. Router misconfigurations, system oversights, incompatibility, and human error all add to accidental attacks. Although accidental application problems, such as deleting a critical file (or an entire database) are significant, accidents concerning the network generally impact a wider group. For example, a backhoe may knock out networking access for a neighborhood, or a misconfigured router may create a routing loop, congesting a network used by an entire company.

An Attack, or an Interesting Event?

Most users dread system attacks, whereas most system administrators generally overlook "interesting events." So how can someone distinguish an attack from an innocent event or user error? A good rule of thumb is "if it happens once, it's a fluke; if it happens twice, it's a coincidence; if it happens three times, it's an attack." This means, for example, a problematic application that generates one error, one time is not an attack. But, if this same application causes the same problem many times, then it is a threat to the system's security.

1.3 CONCEPTS

Five basic concepts form the foundation of risk management: confidentiality, authentication, authorization, integrity, and repudiation. A system can be strong in one area and weak in another, which leads to a potential vulnerability. For example, bank ATM systems use PIN numbers to identify customers and grant access. Although the PIN supplies authorization, it may not supply integrity (the PIN may be stolen) or confidentiality (anyone can see you at the ATM).

1. Confidentiality and Privacy

Confidentiality is the ability to operate in private. Systems that provide confidentiality mitigate the risks from an eavesdropper or attacker. The level of required confidentiality varies with needs of an environment. For example, email is transmitted in plain text. Any person that can intercept the email may read it, including mail relay systems. An encrypted email ensures that the content cannot be read, but the sender and recipient may still be disclosed; at minimum, an attacker can see that an email was sent. By embedding a hidden message within an email (steganography) or generating many fake emails (chaffing), a sender may lower the likelihood of an attacker identifying the communication and ensure a higher degree of privacy.

2. Authentication

Authentication permits one system to determine the origin of another system. This becomes essential in an online community, where two systems are usually not directly connected. Anyone can attempt to impersonate anyone else. Authentication systems provide a means to identify a system or data as authentic.

3. Authorization and Access Control

Not everyone (nor everything) is equal. Based on authentication, systems, processes, and users are offered different levels of access. *Authorization* is the level of access control that is permitted. For example, anyone may dial a phone number to an office building—there is no access control restricting who may dial. But not everyone may enter the building—access is restricted to authorized staff, and the authorization is based on an ID, badge, or key authentication.

4. Integrity

When transmitting information, a recipient should be able to validate that the information was not modified in transit. Information tampering or modification changes the *integrity* of the information. A system with a high degree of integrity should be difficult to tamper.

5. Nonrepudiation

Authentication ensures that the sender is who he says he is. Integrity ensures that a message is not tampered with. But what links the message to the originator? The ability to *repute*, or deny, originating a message is key to security. If an attacker falsifies a document, the forged originator should be able to repute the document. *Nonrepudiation* ensures that an originator cannot falsely repute information. A system that includes authentication, integrity, and nonrepudiation should be able to

detect tampered information and prevent valid information from being falsely rejected.

4. **COMMON MITIGATION METHODS**

Regardless of the environment—hardware, software, network, and physical—there are a few basic approaches for mitigating security risks. These general approaches work by proactively limiting the impact from a security breach.

1. **Compartmentalize**

Despite our best efforts, systems have bugs. Although these unplanned limitations can impact functionality, large multifunctional systems are more prone to problems. In particular, a flaw in a multifunctional system likely impacts the entire system. For example, a stereo with a built-in CD player may not work if either the stereo or CD player breaks. Similarly, an all-in-one network device that includes a firewall, virus scanner, and IDS may perform none of these tasks if one of these functions is misconfigured or vulnerable to an exploit.

The simplest way to mitigate the risk from a multifunction system failure is to separate out the functionality. Tasks and functions are divided into units containing limited functionality. These units can then be integrated into a larger system. Although the failure of one component may reduce the functionality of the larger system, the failure should not negate the functionality from other units; the scope of a failure is limited to the defective unit.

Compartmentalized functionality has an additional benefit: mix and match. Independent units can be integrated with many other units. With multifunction systems, components are tightly integrated—it becomes difficult to use one function independently. A multifunction network device is unlikely to permit using one function (e.g., an integrated virus scanner) with a different product (e.g., an external IDS from a different vendor). Compartmentalized units lead to additional security: a failure by one vendor does not imply insecurity within every unit.

Independent units do have a few limitations. Each independent unit requires its own configuration; there may not be a system for configuring all units at once. Tightly integrated multifunctional systems can be very desirable with respect to maintenance.

2. **Secure Fail**

Unexpected events happen, from an intentional attacker to a lightning strike. When a failure occurs, the impacted units should securely fail and not increase the threat level. For example, a firewall that fails should block all access. The alternative, a fire-

wall that fails and permits all traffic, may not be immediately noticed and may allow an attacker inside the network.

3. Defense-in-Depth

A single security precaution prevents a single attack. Other attack methods may not be prevented and may bypass the security option altogether. More security options and more varieties of security options yield a system that is more difficult to exploit. For example, a castle in Europe may employ a high wall to deter attacks. But they also use other security options: a moat, drawbridge, and portcullis. If attackers still manage to storm the castle, narrow hallways and twisting stairwells provide advantages to defenders.

This same concept, with layers of security providing *defense-in-depth*, works well for computer networks. A single firewall is good; multiple firewalls are better. Antivirus scanners on computers are good, but also using an antivirus scanner on network traffic and on the email server may prevent widespread outbreaks.

Using tools and services from multiple vendors generally yields more security. For example, a company that only uses Cisco firewalls faces a risk from a Cisco-specific compromise. In contrast, a home user that employs both SMC and Linksys firewalls is unlikely to be breached by a vendor-specific attack. A few companies take variety to an extreme. For example, backups may be stored in triplicate, using three different backup systems on three different computer platforms with three different operating systems. The benefit is critical: a single failure from any one system is not likely to impact the other two systems.

The primary limitations from a defense-in-depth approach are complexity and compatibility. A network with three virus scanners may prevent 99 percent of computer viruses, but the maintenance cost is three times greater. Moreover, the different scanners may be incompatible on the same system.

4. Security-by-Obscurity

A provably secure solution means that an attacker, knowing all the implementation details, will be unable to realistically bypass a security precaution. Many cryptographic systems fall into this category; an attacker who has the encrypted data and algorithm cannot decrypt the data within a meaningful timeframe. (Given 200 years and 1,000 computers, anything is possible.) In contrast, many security precautions become weaker when implementation details are provided. For example, protecting a wireless network by not telling anyone that it exists is a viable security option as long as nobody discovers the network signal. *Security-by-obscurity* denotes any mechanism that is only secure due to a lack of knowledge.

Security-by-obscurity is a viable addition to defense-in-depth, but it should not be the only line of defense. Examples of commonly practiced security-by-obscurity

solutions include password management (don't tell anyone your password) and server details. It is hard enough to compromise a server, but it becomes nearly impossible to compromise a specific system when (1) the type of host is unknown, (2) the location of the host is unknown, and (3) the defenses protecting the host are unknown.

Passwords

Login passwords are an example of security-by-obscurity. Passwords are only effective as long as they remain secret. An ongoing debate concerns the use of passwords. Should passwords be long or short, complicated or simple, static or changed regularly?

Many companies and institutions have strict password policies. These policies include length (e.g., at least eight characters) as well as selection (letters, numbers, no repetition, must include at least one symbol, etc.). Although these passwords are less vulnerable to brute-force guessing and dictionary attacks, they are more likely to be written down or forgotten. The alternative are weak passwords that can be easily remembered but are vulnerable to brute-force guessing.

Whereas many secure organizations require complicated passwords, the banking industry has taken a different tact. Bank account PIN codes are a type of password. PINs rarely change, use a small character set (10 digits: 0 - 9), and usually consist of only 4 characters—a total of 10,000 different combinations. The type and length of a PIN were chosen to reduce the need to write down the combination. Multiple login failures block the account and prevent brute-force attacks, and the main security comes from the bank's closed network—preventing access to the password file.

1.4.5 Security and Usability

There is a common misconception within the software development community of a tradeoff between security and usability: the more secure the system, the less usable it becomes. Actually, this is only true in extreme cases. For example, most software exploits today rely on buffer overflows and unvalidated parameters. Correcting an overflow condition and validating parameters does not impact the usability, but does improve the system's security. In the case of network security, a home user that does not run any external network servers will not be impacted by a firewall that blocks remote users from accessing the system. A corporate server is no less usable if the system is hardened, with all unnecessary services and applications disabled.

In extreme cases, there are usability tradeoffs. For example, systems that require multiple authentication tokens (besides a username and password) may be difficult to use. And a firewall that constantly prompts for user confirmation is likely a bigger annoyance than a security measure. In general, these issues are due to implementation and design faults, rather than security theory.

Usability does not strictly refer to restrictive functionality. Usability includes cost and performance. A secure solution that costs more than the data it protects is likely restrictive—most home users will not pay \$10,000 for a firewall that protects a \$1,000 computer. Similarly, a mitigation option that severely impacts performance may not be a viable option. Encrypted tunnel technologies may provide a secure network connection, but the overhead of the encryption may make some protocols, such as H.323 video conferencing, sluggish to the point of being unusable.

3 Network Theory

In This Chapter

- Standards Bodies
- Network Stacks
- Layers and Protocols
- Common Tools

3.1 STANDARDS BODIES

Standards allow compatibility between systems created by different manufacturers. Whereas some standards are accepted through common usage (de facto standards include many RFC documents), others are approved by governing bodies. The field of computer networking contains a wide variety of groups providing official standards. Although some of these organizations cover unique niches, others overlap in jurisdiction. Table 3.1 lists the most common organizations. Government organizations, corporations, niche associations, and informal groups may also specify standards.

TABLE 3.1 Common Standards Organizations for Networking

Abbreviation	Standards Body/Purpose
ANSI	American National Standards Institute Communication standards
IEEE-SA	Institute of Electrical and Electronics Engineers, Standards Association Low-layer network standards

Abbreviation	Standards Body/Purpose
ISO	International Standards Organization Information technology standards
IEC	International Electrotechnical Commission Electrical, electronic, and related technical standards
IETF	Internet Engineering Task Force Governing body over the Request For Comments (RFC) proposals for standards; network protocol standards



By adhering to standards, separate systems can be ensured to interact in a compatible fashion. Although new approaches may be faster, more efficient, or less expensive, they may not be compatible with existing systems. In general, “standard” is better than “better.”

3.1.1 Standards

Network standards come from a variety of sources. The most common include the following:

Request For Comments (RFC): RFCs provide de facto standard information. They provide a forum for detailing network functionality and compatibility information. Some RFCs are provided for informational insight, whereas others define actual network standards.

Best Current Practices (BCP): BCPs are a subset of RFCs endorsed by the IETF.

For Your Information (FYI): As with BCP, the FYI documents are a subset of RFCs; however, BCP documents are technical, and FYI documents are informational.

Standards (STD): STD is a subset of RFCs that deals with networking standards.

Internet Experiment Note (IEN): The Internet Working Group developed the IEN as a set of technical documents, similar to RFCs. This group of documents merged with the RFC in 1982.

Many of the RFC documents are cross-referenced with BCP, FYI, STD, and IEN numeric identifiers.

3.1.2 RFC

In 1969, a series of documents called *Request For Comments* (RFC) were created. The RFCs were intended to record the unofficial development notes of the ARPANET (which later became the Internet). These documents quickly became the de facto standards used for developing network protocols. Today, RFC documents are treated much more as a standard than true “request for comments”; few RFCs are debated after becoming accepted.

The submission process for RFCs has evolved over the years, but the basics have not changed [RFC1000, RFC2026]. First, a standard is proposed. Anyone can propose a standard. If the proposal has merit, then the submitter creates a draft of the standard. The draft is opened to the community for review and comment, and may undergo many revisions. Eventually, the draft is either dropped as unacceptable or becomes accepted and assigned an RFC numeric identifier.

There are four main risks to the RFC process. First, many RFCs are informational and not proposed standards. Even though they do not define standards, many RFCs are still implemented as standards (see Chapter 22). This means that some RFCs, which have not been scrutinized for implementation issues, may be widely adopted and implemented.

The second risk from RFCs comes from a lack of security. Very few RFCs discuss potential security issues, and some RFCs mention security concerns but offer no mitigation options. Moreover, the few RFCs that discuss security issues are not necessarily vetted by security personnel and do not detail known security risks. For example, RFC2786 discusses the Diffie-Hellman cryptographic key exchange. But this RFC does not detail the risks associated with the algorithm nor mention any experts that evaluated the protocol. Without these details, a noncryptographer could implement the protocol and introduce significant vulnerabilities.

Special interests pose one of the largest risks to the RFC process. Many companies propose parts of standards, but keep other parts proprietary. For example, Microsoft proposed RFC2361, which defines the video/vnd.avi and audio/vnd.wave meta-tags but does not discuss the proprietary AVI format. Similarly, Macromedia's Real Time Streaming Protocol (RTSP—RFC2326) discusses the download mechanism but not the proprietary media formats used in the examples. Both of these are in sharp contrast to Kerberos, PPP, and other RFCs, where all related details are also defined as open standards.

The final risk to the RFC process comes from missing standards. Many peer-to-peer (P2P) and Instant Messaging (IM) formats are widely used but not defined by RFC standards. Similarly, Voice-over-IP (VoIP) and Internet telephony encase open standards within proprietary formats. A VoIP client from one vendor will not work with VoIP servers from other vendors. This is very different from the Web standards defined by RFCs; a browser from one company will work with a Web server from any other company.

Get the Message?

Instant Messaging is partially defined by a few RFCs. RFC2778 and RFC2779 are informational, whereas RFC3859 through RFC3862 are destined to become standards. Each of these describes session management but not details concerning implementation.

Although each of the IM RFCs discusses high-level interactions, they do not describe any consistent packet format. This is a sharp contrast with other protocols defined by RFCs. Telnet, FTP, SMTP, HTTP, DNS, and others protocol standards include high-level descriptions as well as implementation-specific details. RFCs are intended to assist the implementation of interoperable network devices. Without detailed information, there is no compatibility. As such, IM solutions from different vendors are not compatible.

2. NETWORK STACKS

Each system on the Internet uses a network *stack* to manage network data. The stack is divided into *layers* and each layer includes network protocols for specific functionality. Network layers are conceptual, not physical or a byproduct of implementation. Different stacks can still be compatible across networks.

There are many ways to represent network stacks. The two most common are the ISO OSI and TCP/IP stacks. Both define specific layers for functionality. Within each layer are protocols, standards, and conventions for communicating information.

In addition to the OSI and TCP/IP stacks, multiple stacks may be used sequentially or nested, where a single layer in a stack expands to reveal an entire stack. This provides additional functionality when a single stack is not sufficient.

1. Network Stack Design

Network stacks are designed to be modular and flexible, allowing information to flow between diverse environments. The modular design allows new hardware, software, network configurations, and network protocols to communicate without redesigning the entire network. These design goals permit layers to operate independently and provide the opportunity for reliable data transfers.

1. Layer Independence

The different layers within a stack simplify the implementation requirements by segmenting functionality. In theory, a single layer may be replaced without impacting the entire stack. Changing a protocol used by one layer does not require changing the protocols used in any other layer.

For example, two computers may use different hardware, different operating systems, and different software. But as long as the network layers implement compatible protocols, the two computers can communicate across a network. This independence allows a cell phone with a Web browser to access a distant supercomputer. The two network stacks need to be compatible, but they do not need to be identical. Functionality found in one stack, layer, or protocol may not be required on a different system.



Although the conceptual design calls for independence between layers, actual implementations are often dependent. Commonly, a protocol is implemented at one layer with a specific protocol requirement at another layer. But even dependent protocols usually provide some degree of flexibility and are not hardware- and operating system-specific; entire stacks are usually flexible even when specific protocols are not.

3.2.1.2 Network Reliability

Most network layers provide a degree of reliability. Depending on the protocol, transmission errors may be detected or corrected. The simplest standards use checksums to validate data transfers. More complex protocols rely on cryptographic algorithms. By dividing the network stack into layers, the architecture provides many opportunities for identifying and addressing transmission errors.

Many protocols in a network stack provide error detection. Although this can appear redundant, the duplication provides additional reliability and modularity. Whereas *error-free reliability* identifies complex errors, *error-free modularity* ensures that protocols are robust:

Error-Free Reliability: Different protocols in different layers detect different types of errors. For example, a simple checksum may identify one wrong bit, but two wrong bits are undetectable. Whereas a simple checksum may be acceptable for one protocol, a different protocol may be impacted by more complex errors. Because layers operate independently, each layer must validate its own information.

Error-Free Modularity: Each layer is intended to operate independently. This means that one layer cannot assume that other layers performed any error checking. If the data transfer must be reliable, then protocols cannot assume that another layer validated the information.

3.2.1.3 Types of Errors

In general, there are three types of transmission errors: spurious, burst, and intentional. A *spurious* error occurs when a single bit (or byte) changes value. This may occur due to a short amount of transmission interference. For example, if the transmitted data is 01010101 then a spurious error could appear as 01011101.

A *burst* error appears as a sequential segment of incorrect data. Most commonly, this appears as a sequence of identical bits (e.g., 00000 or 11111). The repeated bits could be due to a defective logic gate or a sustained level of interference. A long burst of variable interference may generate “random” data. Although it is possible for spurious or burst errors to pass an error-detection algorithm, most detection algorithms are chosen to limit the number of undetected errors.

The final error category is intentional. *Intentional* errors occur when data is systematically modified. Although these may appear as spurious or burst errors, intentional errors are frequently designed to bypass error detection. When critical network data becomes intentionally corrupted, the result is a denial of service. In contrast, undetected data modifications can lead to a system compromise.

The network stack provides multiple opportunities for error detection. This means an attacker faces a significant challenge when transmitting intentionally modified data that can bypass all error-detection systems. Even when every layer’s protocol uses a simple error-detection algorithm, such as a checksum, the requirement to bypass multiple checksums increases the difficulty.



Simple checksums, even from multiple layers, are usually not a significant deterrent against intentionally modified data that is intended to appear error-free. In contrast, a single cryptographic error-detection system can be a very successful deterrent. But because layers operate independently, a network application has no guarantee that a sufficiently complex detection algorithm is employed within the stack.

2. ISO OSI

In 1983, the International Standards Organization (ISO) released specifications for the Open Systems Interconnection (OSI) network model [ISO7498]. The OSI model defines a seven-layer stack, with each layer providing a specific network function (Table 3.2).

1. Lower OSI Layers

The lower four layers form the network services and communicate data across the network. The first layer, the *physical layer*, focuses on transporting data across a medium to another networked system. This layer ensures that data transmitted

TABLE 3.2 ISO OSI Network Model

Layer	Name	Purpose	Book Section
7	Application	End-user applications	Part VIII
6	Presentation	Data conversion	Part VII
5	Session	Extended duration connection management	Part VI
4	Transport	Data collection and aggregation	Part V
3	Network	Remote network routing and addressing	Part IV
2	Data Link	Local network routing and addressing	Part III
1	Physical	Data transport media	Part II

matches the data received, but it does not protect from data collisions—where two nodes transmit at the same time—or prevent external noise from appearing as transmitted data. The physical layer defines the medium and medium specifications. Examples of layer 1 standards include 10Base-2, 100Base-T, ISDN, and modem protocols such as V.33 and V.42bis. Part II discusses the physical layer, including a detailed look at common wired protocols, common wireless protocols, and the security risks involved in different network configurations.

The physical layer does not address noise or transmission collisions. Instead, the second layer manages this type of error detection. The *data link layer* ensures that data received from the physical layer is intended for the host and is complete, that is, not a fractional data transfer. This layer provides transmission management and simple error detection from collisions and external interference. Additionally, the data link layer manages node addressing on a network with more than two hosts. The most common example of a data link protocol is the IEEE 802 standard, including CSMA/CD and LLC (Chapter 10). Other sample data link protocols include PPP and SLIP (Chapter 9). Part III details the security impacts from simplified data link protocols in point-to-point networks and complex multinode network MAC addressing, including the threats from ARP attacks such as poisoning and flooding.

The third layer, the *network layer*, routes data between adjacent networks and permits connectivity between vastly different data link and physical specifications. Examples of network layer protocols include IP, BGP, IGRP, IPX, and X.25. Part IV covers the general security risks and options that relate to the network layer. The Internet Protocol (IP) is examined in detail, including many different types of IP at-

tacks (Chapter 12). The network layer permits connections between distant networks, which leads to fundamental concerns regarding anonymity (Chapter 13).

The *transport layer* defines ports, permitting different network services to be enabled at the same time. The network and data link layers may transmit data along a variety of paths; the recipient may receive data out-of-sequence. The transport layer addresses data sequencing by placing blocks of data in the correct order. Example transport layer protocols include TCP, UDP, NetBIOS-DG, and many AppleTalk protocols such as ATP, ADSP, and AEP. Part V examples the transport layer, TCP (in particular), and related security implications.

3.2.2.2 Upper OSI Layers

The upper three layers form the end-user layers and assist application management. The fifth layer, the *session layer*, provides support for connections that may exist even when the lower network infrastructure changes. For example, a user may log in to a remote server. The lower OSI layers can drop, reset, or change the network connection, but the user remains logged in. This is commonly used for database and naming services such as DNS and LDAP. The session layer also provides support for remote services such as RPC and RTP. Part VI covers the risks associated with the session layer with a particular focus on one of the most crucial, yet insecure, protocols on the Internet: DNS.

The sixth layer, the *presentation layer*, was originally designed to handle data conversion, such as from ASCII to EBCDIC, but today is commonly used to provide transparent compression and encryption support. Few stacks implement protocols in the presentation layer—this layer is usually transparent. A few examples of presentation layer protocols include LPP, SSH, and SSL. Part VII looks at the presentation layer, with SSL and SSH as examples.

At the top of the stack is the *application layer*. This seventh layer provides application-specific support, and acts as the final interface between network and non-network components of an application. Most user applications implement application layer protocols. There is a handful of presentation layer protocols, but there are thousands of application layer protocols. More common examples include Telnet, FTP, HTTP (Web), SMTP (email), SNMP, X-Windows, NTP, BootP, DHCP, NFS, and SMB. Although each of these protocols have specific security concerns, most share general risks and have common mitigation options. Part VIII looks at general application layer issues and solutions and offers SMTP and HTTP as specific examples.

Almost Consistent

The seven layers of the OSI model are distinct, but their purposes are sometimes interpreted inconsistently. For example, the transport layer is sometimes associated with the upper layers rather than the lower layers. Whereas the lower layers provide network addressing and routing, the upper layers supply service and support. The transport layer provides high-level control for service addressing and low-level control for application management. Thus, the transport layer can belong either or both categories.

The original layer definitions predate the use of cryptography or security methodologies. As such, any layer may implement any cryptographic or security-oriented protocol.

Although the OSI stack contains seven layers, it is sometimes described as having an eighth layer. Layer 0, or the physical medium, is sometimes separated from layer 1, the physical layer. This is commonly done for clarification when the same physical medium can be used with differing signal protocols. Similarly, an imaginary layer 8 is sometimes associated with anything beyond the standard stack. Jokingly called “OSI layer 8,” the human layer (or user layer) is one common example.

3.2.3 DoD TCP/IP Stack

Before the ISO OSI network model, there was TCP/IP. In 1969, the U.S. Department of Defense (DoD) created the TCP/IP stack. This four-layer stack defines the functionality needed for connection-oriented communication. Although the distinct layers in the OSI model are beneficial as teaching aids, the TCP/IP stack is most commonly implemented. Both of these network models are compatible; the functionality in one can be mapped into the functionality of the other (Figure 3.1).

The lowest layer of the TCP/IP stack is the *network interface*. This single layer is a combination of the OSI’s physical and data link layers. The network interface layer defines the communication medium, flow control, error detection, and local network addressing. In contrast to the OSI physical layer, the TCP/IP model’s network interface layer does not define the physical medium—it only defines the communication across the medium.

The TCP/IP model’s Internet layer performs the same duties as the OSI network layer. This layer manages internetwork addressing and routing.

The TCP/IP transport layer is almost identical to the OSI transport layer. The minor exception concerns terminology. In the OSI model, network connections may be defined as connection-oriented or connection-less, indicating whether transmission confirmation is required or not. In the TCP/IP model, there is no distinction within the terminology, although both types of traffic are supported.

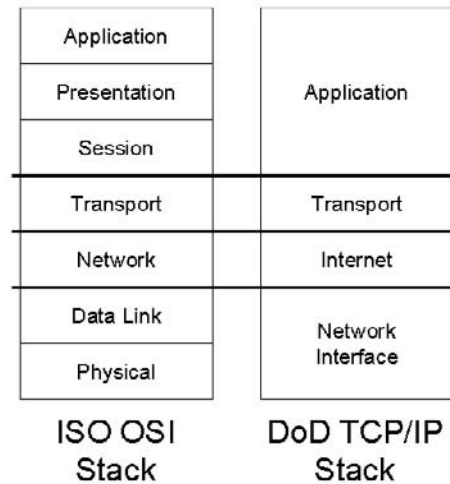


FIGURE 3.1 The ISO OSI and DoD TCP/IP stacks.

The final TCP/IP layer is equivalent to all of the upper layers in the OSI model. The TCP/IP application layer provides session, presentation, and application-specific support. Unlike the OSI model, the TCP/IP model does not distinguish session and presentation from application.

Was That Four Layers or Five?

The TCP/IP network stack is sometimes described as having five layers. The fifth layer, hardware, separates the physical medium definition from the local network addressing and flow control. In the five-layer TCP/IP model, the hardware layer is equivalent to the OSI physical layer.

3.2.4 OSI versus TCP/IP

Although any networking protocol can be placed in the OSI or TCP/IP models, each model has a distinct purpose. The TCP/IP model requires each application to manage the end-to-end communications. Only the basic networking functions are defined. In contrast, the OSI model uses a strict layered hierarchy that contains both networking and networking-related functionality.

Although the TCP/IP model is more simplistic than the OSI model, it is also much older and the de facto standard. The Internet is built on TCP/IP. The OSI

model, with its well-defined segmented layers, is preferable as a teaching tool, but not every layer may fit cleanly in the OSI model. (See Sections 3.4.2 and 3.4.3.)

3.2.5 Other Stacks

Although the OSI and TCP/IP models are the most common stacks, other stacks exist. For example, the Data Over Cable Service Interface Specification (DOCSIS) stack is commonly used with cable and DSL modems (Figure 3.2). DOCSIS defines a multilayer stack, but the layers do not correspond with the ISO OSI layers. The OSI physical layer corresponds with the lowest DOCSIS layer. The first layer, the DOCSIS physical layer, is split into three components: the actual medium, the modulation layer that manages the radio or optical signals, and the transmission layer that handles the actual data transfer and receipt. The layer is also divided between uplink and downlink directions. The DOCSIS data link layer is similar to the OSI data link layer except that it includes MPEG encoding and compression. The upper layers of the DOCSIS stack mirror the OSI stack, with one addition: DOCSIS includes control messages along with the upper layers.

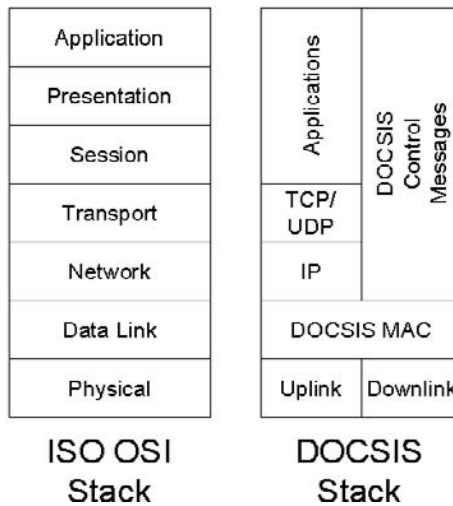


FIGURE 3.2 Aligning the DOCSIS and OSI network stacks.

Other stacks are not as common as OSI or TCP/IP, but there is generally a mapping between different models.

3.2.6 User Layers

The network stacks only cover network-oriented protocols and standards. For example, the application layer provides network support to user-level applications. Programs (or functions) that do not use the network are not in the application layer and not part of the network stack. Outside the network stack are the user, system, and kernel spaces. Commonly called the *user layers*, these spaces host nonnetwork-oriented applications such as games, word processors, and databases.

3. MULTIPLE STACKS

In both the OSI and TCP/IP models, each network stack is intended as a linear transaction: transmissions flow down the stack, and received data flows up. There are no loops or jumps within a stack. Although some layers may perform no function, these layers still exist as empty layers. When looping functionality is required, a second stack is introduced. For example, an initial connection may use one network stack to perform authentication and a second stack to manage the authenticated data. An *authentication stack* is an example of two stacks in sequence. Similarly, a virtual network may use an established network connection (one stack) as a virtual medium for another stack. These stacks within stacks permit the creation of virtual private networks (VPNs). Finally, adjacent stacks permit the conversion from one protocol to another.

1. Sequential Stacks

Stacks may be placed in sequence, so that the application layer of one stack connects to the physical layer of the next (Figure 3.3). Sequential stacks are commonly used for authentication. As an authentication stack, the first stack blocks the initial network connection. The application associated with the stack manages the transfer and authentication of credentials. When the credentials are validated, the next stack handles the connection.

2. Stacks Within Stacks

Each layer of a stack may modify the data. The only condition is that a modification during sending must be reversible during receipt. Modifications may include encapsulation (adding headers and trailers around the data) or encoding, where the data is transformed into another representation. Encoding examples include SLIP's simple character replacements (Chapter 9) and the encryption and compression provided by Secure Shell (Chapter 20).

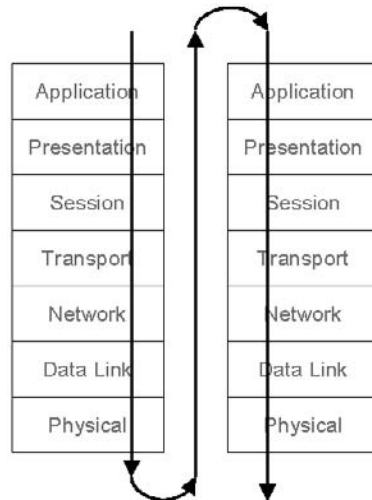


FIGURE 3.3 Two sequential stacks.

Because any data that passes down a sending stack appears the same after it passes back up the receiving stack, stacks can be nested. In a nested stack, a protocol in a specific layer may contain all the elements, from application to physical layer, of another stack. Nesting is commonly used when a network-based application fills the role of a single network protocol. A Sockets proxy (Socks) is an example of a stack within a stack (Figure 3.4). Socks is used to proxy connections through a remote system. Socks manages this by intercepting and modifying the IP headers of a data transmission. Rather than modifying all IP applications to use Socks, most Socks implementations intercept the network stack. After the data passes through the network layer, the Socks protocol redirects the data to a new stack rather than the remaining data link and physical layers. The new stack modifies the data for use with the Socks protocol, and then passes the data through the Socks data link and physical layers—the lower layer for the Socks stack may differ from the original lower layers.

Upon receipt by the Socks server, the external stack is peeled away, the IP layer is replaced, and the data is transmitted using the proxy's data link and physical layers.

3.3.3 VPN

A virtual private network (VPN) is a common example of nested stacks. In a VPN, the physical layer of one stack is actually an application for the next stack. Secure Shell (SSH) and `stunnel` are examples of protocols that can tunnel other protocols. The basic concept redirects the physical medium to an application that contains an

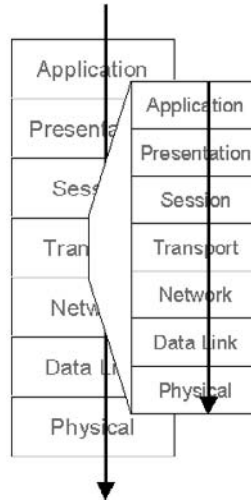


FIGURE 3.4 Example of a stack within a stack.

established connection. (This is discussed in detail in Chapters 9, 19, and 20.) The data for the virtual connection is tunneled through the application.

Tunneling is only part of a VPN. The second part requires network routing to be directed through the tunnel. This redirection can make two remote networks appear adjacent. VPNs are commonly used to connect two private networks over a public connection. The connection is done in three steps:

1. **Establish the public connection.** A dual-homed host, existing on both a private network and a public network, is used to connect to another dual-homed host. The connection is across a public network.
2. **Create the tunnel.** An application, such as `ssh` or `stunnel`, is used to tunnel a connection between the two hosts.
3. **Create the network.** A networking protocol, such as PPP, is initiated through the tunnel. This allows network traffic to flow between the private networks.

After establishing the VPN connection, network traffic is routed across the virtual network. Nodes using the virtual network may experience a latency related to the tunneling, but the two networks will appear adjacent.

4. **LAYERS AND PROTOCOLS**

Each layer in a stack may contain one or more protocols. A protocol implements a specific functionality associated with the layer. Although there are only two widely used stacks, and only a few layers to define each stack, there are literally thousands of standardized protocols, and even more that are considered nonstandard. Whereas stacks and layers are conceptual, protocols are specific definitions for the layer concepts. Two nodes on a network may use very different software to implement a particular protocol, but two implementations of a protocol should be compatible.

Because protocols apply the layer concepts, they may not fit perfectly into a single layer. Misaligned mappings and additional functionality can lead to an imperfect association between a protocol and a particular layer or stack.

1. **Mapping Protocols to Layers**

In general, the convention is to design and implement protocols so that they map directly into one layer. A single protocol should not perform tasks found in multiple layers. Unfortunately, due to poor design, ambiguous layer functionality, and technical constraints, many protocols are tightly associated with multiple layers. Moreover, a protocol may be designed for one stack, such as the TCP/IP stack, and therefore not have a perfect mapping into another stack like the ISO OSI model. For example, the IEEE 802 standard is designed for the TCP/IP stack and defines the network interface layer. Subelements of the standard, such as IEEE 802.3 are closely associated with the OSI data link layer, whereas IEEE 802.3z describes the OSI physical layer for Gigabit Ethernet. The set of IEEE 802 standards fit well into the TCP/IP model but not into the OSI model.

In addition to matching multiple layers, some standards and protocols define portions of layers. The IEEE 802.2 standard defines a subsection of the OSI data link layer, and resides above the IEEE 802.3 definition (see Chapters 5 and 7).

2. **Misaligned Mappings**

Protocols are often tightly associated between layers. Although this usually happens within a single layer, such as the association between SLIP and CSLIP—a compressed version of SLIP—the association can occur between layers. For example, IP resides in the network layer. But IP uses the Internet Control Message Protocol (ICMP) for routing and traffic management. Some of the ICMP functionality clearly resides in the network layer, as a sublayer above IP. But other ICMP functions appear to cover functionality found in the transport layer. For example, ICMP manages the IP flow control, which is network layer functionality. But ICMP can also define data to be transported, such as the content of an ICMP Echo (ping)

packet. The capability to define data for the network layer occurs in the transport layer. Thus, ICMP contains both network and transport layer attributes.

Because protocols represent a specific implementation and not an abstract concept, and because protocols may be designed for an alternate stack, many protocols do not map properly into a specific stack model. Although there may not be a perfect mapping, there remains a general alignment.

3.4.3 Different Layers, Different Views

The implementation of a specific protocol or standard may incorporate the functionality from multiple layers. As such, different people may associate protocols with different layers. A very common example is the Secure Shell (SSH) executable, `ssh`. This executable is primarily used to implement an encrypted tunnel. Encryption is a modification of the data and usually associated with the OSI presentation layer; however, `ssh` implements a few other functions:

Login Functionality: SSH provides user authentication. The authentication covers the entire protocol's connection and permits an extended connection. Extended connection support is usually associated with the OSI session layer.

Remote Procedures: The SSH daemon, `sshd`, can provide specific services, or functions. This is reminiscent of remote procedure calls (RPC), a function usually associated with the OSI session layer.

Command Prompt: When `ssh` is used by itself, it may run a remote program or start a login shell. Both remote program execution and providing a login shell are OSI application layer functions.

SSH fits well within the application layer of the DoD TCP/IP stack, but it does not fit well within the ISO OSI model. Although SSH is best described as an OSI presentation layer protocol, it is commonly included in the session or application layers.

Another example of an ill-fitted protocol is SAMBA, the open source implementation of Microsoft's Server Message Block (SMB) protocol. SMB is used to share resources, such as disk drives or printers, between computers. SMB is an application layer protocol that is dependent on the NetBIOS protocol. NetBIOS is sometimes classified within the transport layer, but also provides named pipe support—session layer functionality. The implementation of the SAMBA client includes SMB, NetBIOS, named pipes, and even an FTP-style interface—an application layer function that interfaces with the SMB application layer. The SAMBA client includes OSI transport, session, and application layer functionality.

5. COMMON TOOLS

Network maintenance and security requires tools for querying, collecting, assessing, and analyzing network traffic and connected nodes. Although there are a large variety of tools and many are open source, only a few regularly exist on most operating systems. Some of the more popular tools may need to be downloaded and installed. The common tools perform general functions, and the specific tools test explicit functions.

1. Querying Tools

Querying tools permit an administrator to test network connectivity and the functionality of specific protocols. These tools test connectivity to a particular network protocol. A successful query indicates network connectivity from the local host to the queried protocol on the remote host, but does not indicate full connectivity within the stack. Common query tools include the following:

Arp: A tool that lists the local ARP table (data link layer, see Chapter 10), modifies the ARP table, and more importantly, generates an ARP network request. If the `arp` command cannot identify a host on the local network, then the host is not on the local network (or not online).

Ping: A simple tool that generates an ICMP echo request. A successful `ping` indicates connectivity along the physical, data link, and network layers.

Netstat: The `netstat` command is available on nearly all networked operating systems, although the parameters and output format may vary. This tool provides a variety of network related status reports, including current network connections, routing tables, and number of bytes transmitted and received. Metrics can be displayed by network interface or protocol.

Telnet: Although originally designed for establishing a remote shell connection, `telnet` is commonly used to test connections to remote TCP services. Telnetting to a specific port on a remote host can test the connectivity of the entire remote network stack. The `netcat` application (`nc`) is a common alternative to `telnet`.

Nmap: Nmap is an extremely powerful tool for identifying remote hosts. Nmap is capable of identifying active hosts (e.g., an ICMP ping scan) and transport layer ports (TCP and UDP). In addition, it contains fingerprinting capabilities. Nmap not only identifies available TCP network services on a remote host, but it can attempt to identify the type of services (Web, email, etc.) and identify remote operating systems.

p0f: Similar to `nmap`'s operating system fingerprinting service, `p0f` estimates the type of remote operating system based on packet sequencing and related attributes.

2. Collection Tools

Collection tools are used to gather network traffic for analysis:

TCPdump: TCPdump interfaces with the data link layer and collects all packets received by the local host. Although the application's name is `TCPdump`, it can collect all network traffic that reaches the data link layer, including ARP, IP, TCP, UDP, and higher protocols. TCPdump is the de facto standard for network data collection; collected data is usually stored in the `tcpdump` format.

Snort: Snort is a very sophisticated alternative to `tcpdump`. Snort permits complex packet filtering and alerts based on specific matches. This application is the basis of many open source IDSs.

3. Assessment Tools

Vulnerability assessment tools analyze network interfaces for known and potential risks. Any segment of the network stack may be analyzed, although most tools primarily focus on the OSI session, presentation, and application layers.

Three common assessment tools are Nessus, SATAN, and Saint. These tools analyze services on remote hosts for known risks. Analysis methods include port scans that identify potential network services, packet-based host profiling, and specific service assessment. After identifying the remote network service's port, the service is queried. Frequently, the tools can identify not only the type of service but also the exact version and patch level of the service. The identified service is then compared to a list of known service exploits.

Although these tools offer very powerful ways for administrators to quickly check their own networks for known risks, attackers can use these same tools to scan for vulnerable hosts. An attacker may use Saint to scan a subnet for particular vulnerabilities and later return with appropriate exploits.

Not all of the scans performed by these tools are harmless. For example, many of the scan options in Nessus are explicitly hostile. These scans test for overflow and write-enabled vulnerabilities. In addition, many services do not handle network scans well. Nessus scans that are not denoted as hostile may still hang, reboot, or otherwise impair the target system.

4. Analysis Tools

Analysis tools are used to view and interpret collected packets, and to generate explicit test cases. Ethereal is an example of a powerful analysis tool that can reconstruct network connections and decode hundreds of different network protocols.

Wireshark (formerly called Ethereal) has one specific limitation: it can only passively collect and analyze data. Wireshark has no way to inject packets. Without this functionality, it cannot generate specific test cases. A separate analysis tool, `hping`, offers a simple command-line interface for generating custom ICMP, UDP, and TCP packets (as well as other protocols).

3 Basic Cryptography

In This Chapter

- Securing Information
- Authentication and Keys
- Cryptography and Randomness
- Hashes
- Ciphers
- Encryption
- Steganography

1. SECURING INFORMATION

There are three basic approaches to securing information: prevention, restriction, and cryptography. Access to information can be prevented. If an attacker cannot access information, then the information is safe from the attacker. For network security, isolated networks and restrictive architectures are generally sufficient deterrents. For example, many government offices have two computers that are not networked to each other. One computer accesses the public Internet, and the other only operates on a classified network. Without a connection between the two computers, it becomes difficult for an attacker on the unclassified network to attack the classified network. A classified network security breach indicates a conscious effort and not an accidental configuration error.

When networks cannot be isolated, access can still be restricted. Most remote login systems require a username and password. Different types of authentication exist for different security needs. Although authentication restricts access, it generally does not hinder eavesdropping-related attacks.

Cryptography is the most common approach for securing networks. Cryptography encodes data so that only the intended recipients can decode the message. Cryptographic systems include random number generators, hashes, ciphers, and encryption algorithms. *Steganography* (hiding information) is commonly associated with cryptographic algorithms.

4.3 AUTHENTICATION AND KEYS

When a user remotely connects to a computer, he is identified with the system. If the user is authenticated, then access is provided. Unidentified users are blocked. The many types of authentication mechanisms generally fall into one of three classifications:

Something you know: Passwords and PIN numbers are types of information used to access systems. The authentication assumes that knowledge of these secrets is sufficient to authenticate the user.

Something you have: Room keys and badges are physical items that provide authentication. Physical keys and badges can be copied, however, digital badges and keys can deter counterfeiting efforts.

Something you are: Biometrics, such as fingerprints, iris and retina patterns, and even handwriting patterns, are distinctive. Unlike the other two classifications, these do not require memorization and cannot be easily passed to other people.

Simple authentication systems may use only one type of authentication, such as a password. Single authentication methods are called *one-part authentication*. Authentication systems that require many elements from one category are also called one-part authentication. In general, a system that requires two types of memorized information is just as secure as a system that requires one larger piece of memorized information. Needing two digital keys to enter a system is no safer than needing one digital key.

More complex systems use *two-part* or *three-part authentication*, indicating elements from two or three authentication classes. For example, requiring a password and fingerprint is much stronger than requiring either alone.

In cryptography, keys are used to access data. The keys may come from a variety of elements, including passwords, random numbers, and *hashes* (digital summaries) of provided information.

Something You Are... Missing

Each of the three authentication classifications has distinct weaknesses. The most common type of authentication relies on passwords. Users are expected to memorize these bits of information, but they often forget them. Up to 80 percent of system administration help calls concern forgotten passwords [Hayday2003]. In addition, attackers can steal passwords—they only need to observe the password to compromise it. Users that access many systems are left with a dilemma: they can use many passwords but risk forgetting them, or they can use one password and potentially compromise all of their systems. In addition, users commonly write down passwords, allowing another vector for password theft.

Physical keys, such as digital tokens, badges, and USB dongles (devices that plug into USB sockets) can be misplaced or stolen. Damage is also common—most digital keys are pocket size and do not survive washing machine rinse cycles.

Biometric authentication systems are often touted as secure because the information cannot be duplicated, stolen, or misplaced. But these solutions have their own limitations. For example, biometric devices that require fingerprint identification block access to people who are missing fingers or hands. Similarly, a variety of eye diseases can impact iris and retina identification. Even voice identification can be deterred by the common cold. Although biometrics are very accurate, they are also very restrictive and influenced by temporary and permanent physical changes.

4.3.1 Key Management Risks

Regardless of the key type, key data must be stored on the computer system for use in a cryptographic system. Passwords, physical key identifiers, and biometric information are stored in files and databases. An attacker who can compromise a database of fingerprints can generate authoritative fingerprint information—without having access to the physical finger. There are three main approaches for protecting password information. The first approach simply restricts direct access to the key database. Although the database must be accessible, this does not necessarily mean direct access. Restrictive filters and programming interfaces can mitigate the need for direct access.

The second approach encrypts the key information within the repository. The *Pretty Good Privacy* system (PGP), for example, uses a cryptographic key to encode

and decode data. A PGP key must be stored on the local system. To protect the key from theft, it is encrypted using a password. Only a user that provides the correct password can decrypt and use the key. This effectively wraps one-part authentication (something you have, the PGP key) with one-part authentication (something you know, the PGP key password); however, this aspect of PGP does not offer two-part authentication.

Instead of storing the actual key, the third approach stores a hash, or *digest*, of the key. Any key that generates the same hash value is considered to be acceptable. Many password systems use this approach, including Unix `/etc/passwd` entries and LDAP SHA1 passwords. To reverse, or *crack*, these password systems, an attacker must identify a password combination that generates the same hash. Tools such as John The Ripper (<http://www.openwall.com/john/>) and Rainbow Crack (<http://www.antsight.com/zsl/rainbowcrack/>) are used to guess passwords and identify hash matches.

That's a Password?

The Unix `/etc/passwd` file stores username and account information. Traditionally, it also stores a hash of the user's password. If the password were stored in plaintext, any user on the system could see all other passwords. Instead, the password is encoded. Older Unix systems used an algorithm called *crypt*. *Crypt* uses a modified version of the DES encryption algorithm—modified to be a hash instead of an algorithm that supports decryption. Later versions of Unix switched to MD5. (OpenBSD uses a variant of the Blowfish encryption algorithm.)

People commonly use the same password on multiple systems. By using a hash function, these systems ensure that the password cannot be recovered. Even if an administrator's account becomes compromised, the passwords remain safe.

In addition to using strong hash functions, most Unix systems have moved the passwords out of the `/etc/passwd` file. The `/etc/shadow` file stores passwords, whereas `/etc/passwd` stores user information. The shadow file is only accessible by an administrator. Regular users on the system cannot view the hashed passwords. This prevents dictionary attacks against the known hash values. Finally, `/etc/shadow` is not exported from the system. Remote attackers see access denial, local attackers see access restriction, and compromised administration accounts are faced with information that cannot be decoded.

2. Keys and Automated Systems

When an automated system uses encryption, the key must be made available to the system. This means that any attacker with access to the system will have access to the key. Although many complicated systems attempt to encrypt, encode, or hide keys that are used by automated systems, these are all instances of security-by-obscurity. Automated systems generally do not wait for information from manual sources; the data must be available when the system needs it. If the key is available to automated systems, then the key is potentially available to all users, friend and foe.

3. Symmetrical versus Asymmetrical Keys

The two primary types of keys are symmetrical and asymmetrical. *Symmetrical keys* means the same key is used for encryption and decryption. When using a network-based encryption system, both ends of the network require the same key. Risks are associated with the initial transfer of the key (it can be intercepted), but after the transfer, it can be used to safely encrypt data. DES, Blowfish, and AES are examples of encryption algorithms that use symmetrical keys.

Asymmetrical algorithms, such as PGP, use one key to encrypt and a different key to decrypt. The key pairs, called the *public* and *private keys*, allow key exchanges without risk. Both systems exchange public keys, but do not reveal their private keys. The public key is used for encryption and generates ciphertext—any user can encrypt the data—but only the private key can decrypt the ciphertext. In this scenario, an interceptor that sees the asymmetrical key exchange cannot crack the algorithm.

The public key is not always used for encryption. The private key can also create ciphertext. In this situation, anyone with the public key can decode the ciphertext. The benefit of this approach comes from nonrepudiation; the sender cannot claim that he did not encode a message, and all recipients know that the message is authentic. This is commonly referred to as a *cryptographic signature*. A sender may use private and public keys for encoding a message; the recipient's public key ensures message privacy, and the sender's private key signs the encryption, which ensures nonrepudiation.

In general, symmetrical algorithms are faster than asymmetrical. Many systems initiate the communication flow with asymmetrical keys. Then a symmetrical key is transmitted using the asymmetrical encoding. When both ends have the symmetrical key, they switch over to the faster symmetrical algorithm. This type of handshake ensures the safe transfer of the symmetrical key between two authenticated parties and provides the speed benefits of a symmetrical algorithm.

4.3.4 Key Exchange

Symmetrical and asymmetrical keys require the keys to be shared before use. In 1976, Whitfield Diffie and Martin Hellman developed an algorithm to exchange a secret key in a public forum without compromising the key [Diffie1976]. Called the *Diffie-Hellman key exchange* (DH), this algorithm is based on a mathematical property: factorization is more difficult than multiplication. DH is commonly used in network protocols because keys are not always preshared. IPsec, IPv6, SCTP, SSH, and SSL all support the DH key exchange.



The file dh.c on the companion CD-ROM contains sample code that performs a Diffie-Hellman key exchange.



In the basic DH algorithm, two parties share a predetermined prime number and primitive root. A *primitive root* (r) is any number where the modulus of its exponents covers every value (v) up to the prime number (p).

$$\text{For each } v=1\dots p; \text{ there exists some } k; r^k \bmod p = v \quad (4.5)$$

For example, if the prime number is 7,919, then 3 is a primitive root, but 107 is not. The primitive root is used to generate shared secret values. By covering all possible values (v) up to the prime, the primitive root ensures that an attacker must scan up to 7,919 values to identify the key.



This simple example used 7,919, which is too small to be a secure prime value. Most implementations use much longer prime numbers, where the time needed to compute each r^k becomes prohibitive. Good prime numbers can be 300 digits or longer.

The prime number and root can be publicly known and traded. Disclosing this information does not impact the security of the algorithm. Each party also selects a private key. The private keys a and b are combined with the primitive root and prime numbers to generate public keys, A and B

$$A = r^a \bmod p \quad (4.6)$$

$$B = r^b \bmod p \quad (4.7)$$

The public keys are exchanged and used to determine the shared secret (s):

$$s = r^A \bmod p = r^B \bmod p \quad (4.8)$$

The shared secret (s) is known to both parties but was never transmitted over the network. An attacker who observes the key exchange knows the public keys (A and B), prime number (p), and primitive root (r), however, to derive the shared secret, the attacker must also determine both private keys. If the prime number has 300 digits (requiring 1024 bits, or 128 bytes, of storage), then it can take a very long time (decades or centuries) to identify the shared secret. Alternatively, the attacker could simply try all of the possible shared secret values—an equally daunting task.

5. Certificates and Certificate Authorities

The Diffie-Hellman key exchange provides the means to generate a shared secret value, but it does not authenticate either party. Although a man-in-the-middle (MitM) attack cannot compromise an observed key exchange, the MitM can impersonate either end of the key exchange. If an attacker wants to compromise the key exchange between a client and server, he can intercept the initial key exchange. This allows the attacker to exchange keys with the client and server independently and then relay data between them.

Certificates and certificate authorities provide the means to authenticate the ends of a key exchange. A *certificate* is a unique data sequence associated with a host. A *certificate authority* (CA) is a trusted third party that can validate a certificate. When the client and server exchange public keys, they each query the CA with the public certificates. Only authenticated certificates are accepted.

One of the most widely used certificate systems is the Secure Sockets Layer (SSL) protocol. SSL is a presentation layer protocol commonly used to provide security to Web connections—HTTPS is HTTP with SSL. Although certificates do provide authentication, they are not always used in a secure manner. The main risks concern automation, bidirectional trust, and the trustworthiness of the CA server—each of these are detailed in Chapter 21.

6. Kerberos

The *Kerberos* session layer protocol is an example of a strong certificate-based system. Kerberos uses a *key distribution center* (KDC) for initially granting a certificate. The client must authenticate with the KDC. The authentication requires two parts: a password or digital token, and a client-side certificate. The Kerberos *Authentication Server* (AS) validates the certificate. The result from the KDC and AS exchange is an authenticated client, with the client holding an authentication key.

The KDC and AS allows the client to communicate with a *Ticket-Granting Service* (TGS). Each network service requires a ticket, or service-specific certification, for granting access. The authenticated client requests the service and duration, and is provided a *Ticket-Granting Ticket* (TGT). The client may use the service as long as the TGT is valid.

For example, if a user wants to access a Web server using Kerberos, then:

1. The user logs in to the system, validating himself with the KDC and AS.
2. The user's Web browser uses the key to request a TGT from the TGS for accessing the Web server.
3. The browser then provides the TGT to the Web server, granting access to the server.

With Kerberos, all keys have durations. The TGT may expire, requiring a new TGT to be requested from the TGS. Session may also expire or require renewal.

In addition to authenticating connections and users, Kerberos provides a foundation for encryption. Each key and certificate encrypts traffic. As a result, an attacker cannot view information transfers, replay packets, or hijack sessions. If an attacker does manage to crack a session, it will happen long after the session has expired. An attacker is better off targeting an end user or server system than attacking the network traffic.

Kerberos is an example of a secure network protocol. It properly employs authentication and privacy through the use of encryption and key management. In addition, Kerberos is an open standard (RFC4120), with source code available from MIT (<http://web.mit.edu/kerberos/>). Systems that use Kerberos include IBM's Distributed Computing Environment (DCE), Sun RPC, Microsoft Windows 2000 and XP, and some versions of X-Windows.

Unfortunately, Kerberos does have limitations. The overhead and management of Kerberos limits its practical usage. In addition, variations of the implementation are not compatible. For example, the MIT Kerberos is not compatible with the Kerberos security in Windows 2000. Finally, applications must be modified to use Kerberos—the protocol is not transparently supported.

4.4 CRYPTOGRAPHY AND RANDOMNESS

An ideal cryptographic system is not predictable. A particular plaintext value generates a specific ciphertext value, but the ciphertext is unknown without applying the algorithm. This prevents an attacker from readily determining the plaintext from the ciphertext. Three common methods are used to obscure, or randomize, the plaintext-ciphertext relationship: random number generators, confusion, and diffusion. These methods combine to form cryptographic *substitution boxes* (S-box). An S-box is used to obscure the relationship between the plaintext and ciphertext.

4.4.1 Random Numbers

Most cryptographic algorithms are based on a controlled random number generator. Given an input, the generator creates an output. But one input and output pair cannot be used to determine a different input and output pair. For example, a simple linear congruence pseudo-random number generator [RFC1750] bases the output on the previous value:

$$v_{i+1} = (v_i \bar{a} + b) \bmod c \quad (4.9)$$

Given the same inputs (v_i , \bar{a} , b , and c), it generates the same output (v_{i+1}); however, the output value is not immediately predictable without applying the algorithm. Most random systems store the previous value. For example, in ANSI-C, the `srand()` function initializes (*seeds*) the first value (v_0). All subsequent calls to the `rand()` function update the previous value.

In cryptography, the random algorithm is seeded using the key. The random values are combined with the plaintext to create ciphertext. In decryption, the random sequence is re-seeded, and the random elements are removed, leaving the plaintext.

More complicated random number generators use a variety of mathematical operators, including addition, multiplication, bit shifting, and logical AND, OR, and XOR. In addition, some use external data sources for generating *entropy* (information) to mix into the random algorithm. In general, external entropy is not desirable for cryptographic systems because the random sequence cannot be re-generated to decode the ciphertext.



Encryption with external entropy creates a different ciphertext each time. But the difference may not always impact the decoding. As long as the external entropy can be added and not change the decoded plaintext, it can be used to deter cryptanalysis.

4.4.2 Confusion and Diffusion

In 1948, Claude Shannon described two properties for obscuring plaintext contents: confusion and diffusion [Shannon1949]. *Confusion* refers to data substitution. For example, a Caesar cipher performs letter-by-letter substitutions (e.g., replace all occurrences of \bar{a} with \bar{w}). Similarly, Rot13 rotates each letter by 13 characters (\bar{a} becomes \bar{n} , \bar{b} becomes \bar{o} , etc.). More complex systems may perform substitutions across bit patterns or blocks of data.



Ciphers that strictly rely on confusion use the replacement character sequence as the cryptographic key. With Rot13, the key is known, but other ciphers may require statistical analysis or brute-force guessing to determine the key.

Even though confusion impairs readability, it remains vulnerable to cryptanalysis. In English, the letter *e* is the most common letter. Caesar and Rot13 ciphers do not remove this property—the most common letter from an encoded English paragraph is likely *e*. Complex substitutions may require additional analysis, but they are still vulnerable to frequency and pattern recognition.

Diffusion spreads data. This can be a simple bitwise rotation or a more complex *weaving*. For example, the bits **110011** may be woven by rotation (**100111** or **111001**) or by interlacing (e.g., patterns such as odd and then even positions: **101101**). When diffusion is used across a large block of data, individual byte frequencies can be obscured.

4.4.3 S-Box

S-boxes combine elements from random generators, confusion, and diffusion. For example, the *Data Encryption Standard* (DES) uses 8 different S-boxes to mix blocks of 64 bits. Sixty-four bits of plaintext are fed into the S-boxes to generate 64 bits of ciphertext. This first pass through all 8 S-boxes is called a *round*. DES then feeds the output into the S-boxes again. In total, DES uses 16 rounds to generate the final ciphertext.

The basic concept of combining S-boxes and rounds exists with most cryptographic systems. The *Advanced Encryption Standard* (AES) uses more complicated S-boxes than DES but fewer rounds. The MD5 cryptographic checksum uses 4 small S-boxes, 16 times each per round, through 4 rounds.

4.5 HASHES

Secure systems validate data integrity, checking for intentional information tampering as well as accidental corruption. The degree of validation varies based on the needs of the environment. For example, SLIP (Chapter 9) operates over a point-to-point serial cable. SLIP offers no validation or integrity checking because a direct serial connection is generally black and white with regards to operation: it is either functional or it is not. In contrast, the SSH (Chapter 20) network protocol has packets that traverse many networks, and the SSH contents are intended to be private. SSH uses a very sophisticated integrity detection system.

Most integrity validation systems use hash functions. A hash function maps a large set of input into a few bits of data that describe the input. Although it is likely

that many different inputs generate the same hash value, identifying the cases where the input does not match the hash value can validate integrity. Data tampering and corruption usually results in mismatched hash values.

Five main types of hash functions are used with network protocols: parity, checksum, CRC, cryptographic, and signed cryptographic. Although most network protocols use simple parity, checksum, or CRC hashes, a few protocols use more secure cryptographic hashes.

1. Parity

The simplest type of hash function is a parity check. A *parity check* sets a single bit to 0 or 1, depending on the number of 1 bits in the input sequence. For example, if the input is 00110101, then the 1-bit parity is 0. An even number of bits is assigned 1, and an odd number is assigned 0. A reverse parity check swaps the value assignment (0 indicates odd). A 1-bit parity check can detect single bit errors, but multiple-bit errors that change an even number of bits retain the same parity value. Although parity checks are common for physical and data link layer protocols [RFC935], weak error detection makes parity checks undesirable for higher-layer network protocols.

2. Checksum

A *checksum* extends the parity concept by adding bits. For example, an 8-bit checksum will segment the input into 8-bit sections and add each of the sections (ignoring any bit overflows). The resulting 8-bit number is the checksum. An 8-bit checksum can detect multiple bit errors, as long as the total difference is not a multiple of 256. Similarly, a 16-bit checksum adds each set of 16 bits. The resulting checksum can detect that error totals do not differ by 65,536.

Checksums are commonly used by protocols such as IP, TCP, and UDP. The Internet Checksum, defined in RFC1071, uses a *one's complement*. In effect, the checksum is reversed and represented as a 16-bit value.

3. CRC

A *cyclic redundancy check* (CRC) uses a polynomial function as an extension to a checksum. The polynomial function acts as an S-box, generating pseudo-uniqueness. Each piece of data is modified using the S-box and then combined in a similar fashion as a checksum. There are many different CRC functions—each with different polynomials and combination methods (Table 4.1). Although any polynomial can work as a CRC, choosing one that lowers the degree of repetition can be difficult.

TABLE 4.1 Different CRC Polynomial Functions

Name	CRC (bits)	Polynomial	Equation
ATM HEC	8	0x107	$x^8 + x^2 + x + 1$
CRC-10	10	0x633	$x^{10} + x^9 + x^5 + x^4 + x + 1$
CRC-12	12	0x180F	$x^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-CCITT	16	0x1021	$x^{16} + x^{12} + x^5 + 1$
CRC-DNP	16	0x3D65	$x^{16} + x^{13} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^2 + 1$
CRC-16	16	0x8005	$x^{16} + x^{15} + x^2 + 1$
CRC-32	32	0x4C11DB7	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
CRC-64-ISO	64	0x1B	$x^{64} + x^4 + x^3 + x + 1$

Rather than computing each CRC value every time, precomputed tables are used. The tables are commonly based on a particular factor of the polynomial. In addition, many polynomials match bit patterns—computing the actual polynomial is not necessary. For example, CRC-CCITT matches the bit-wise polynomial 0001 0000 0010 0001 ($2^{12} + 2^5 + 2^0$ or 0x1021)—the 2^{16} term is not needed because it is a 16-bit checksum. Listing 4.1 is a function that precomputes the CRC-CCITT table, and Listing 4.2 shows the full table.

LISTING 4.1 CRC-CCITT Initialization

```

/*****
CRC_CCITT_Init(): Initialize the CRC-CCITT table.
This is a 16-bit CRC.
*****/
u_int *   CRC_CCITT_Init ()
{
    static u_int CRC_table[256];
    u_int i;
    u_int Bits;
    u_int CRC;

    for(i=0; i<256; i++)
    {
        CRC = (i << 8);
        for(Bits=0; Bits<8; Bits++)
        {
            CRC = CRC * 2;
            if (CRC & 0x10000) CRC = CRC ^ 0x1021;
        }
    }
}

```

```

    }
    CRC_table[i] = (CRC & 0xffff);
  }

  return(CRC_table);
} /* CRC_CCITT_Init() */

```

LISTING 4.2 The CRC-CCITT Precomputed Table

00 - 07:	0000	1021	2042	3063	4084	50a5	60c6	70e7
08 - 0F:	8108	9129	a14a	b16b	c18c	d1ad	e1ce	f1ef
10 - 17:	1231	0210	3273	2252	52b5	4294	72f7	62d6
18 - 1F:	9339	8318	b37b	a35a	d3bd	c39c	f3ff	e3de
20 - 27:	2462	3443	0420	1401	64e6	74c7	44a4	5485
28 - 2F:	a56a	b54b	8528	9509	e5ee	f5cf	c5ac	d58d
30 - 37:	3653	2672	1611	0630	76d7	66f6	5695	46b4
38 - 3F:	b75b	a77a	9719	8738	f7df	e7fe	d79d	c7bc
40 - 47:	48c4	58e5	6886	78a7	0840	1861	2802	3823
48 - 4F:	c9cc	d9ed	e98e	f9af	8948	9969	a90a	b92b
50 - 57:	5af5	4ad4	7ab7	6a96	1a71	0a50	3a33	2a12
58 - 5F:	dbfd	cbdc	fbfb	eb9e	9b79	8b58	bb3b	ab1a
60 - 67:	6ca6	7c87	4ce4	5cc5	2c22	3c03	0c60	1c41
68 - 6F:	edae	fd8f	cdec	ddcd	ad2a	bd0b	8d68	9d49
70 - 77:	7e97	6eb6	5ed5	4ef4	3e13	2e32	1e51	0e70
78 - 7F:	ff9f	efbe	dfdd	cfcc	bf1b	af3a	9f59	8f78
80 - 87:	9188	81a9	b1ca	a1eb	d10c	c12d	f14e	e16f
88 - 8F:	1080	00a1	30c2	20e3	5004	4025	7046	6067
90 - 97:	83b9	9398	a3fb	b3da	c33d	d31c	e37f	f35e
98 - 9F:	02b1	1290	22f3	32d2	4235	5214	6277	7256
A0 - A7:	b5ea	a5cb	95a8	8589	f56e	e54f	d52c	c50d
A8 - AF:	34e2	24c3	14a0	0481	7466	6447	5424	4405
B0 - B7:	a7db	b7fa	8799	97b8	e75f	f77e	c71d	d73c
B8 - BF:	26d3	36f2	0691	16b0	6657	7676	4615	5634
C0 - C7:	d94c	c96d	f90e	e92f	99c8	89e9	b98a	a9ab
C8 - CF:	5844	4865	7806	6827	18c0	08e1	3882	28a3
D0 - D7:	cb7d	db5c	eb3f	fb1e	8bf9	9bd8	abbb	bb9a
D8 - DF:	4a75	5a54	6a37	7a16	0af1	1ad0	2ab3	3a92
E0 - E7:	fd2e	ed0f	dd6c	cd4d	bdaa	ad8b	9de8	8dc9
E8 - EF:	7c26	6c07	5c64	4c45	3ca2	2c83	1ce0	0cc1
F0 - F7:	ef1f	ff3e	cf5d	df7c	af9b	bfba	8fd9	9ff8
F8 - FF:	6e17	7e36	4e55	5e74	2e93	3eb2	0ed1	1ef0

For the CRC-CCITT algorithm, each byte is combined with the previous CRC value and the index table:

$$\text{CRC}[i+1] = ((\text{CRC}[i] \ll 8) \wedge \text{Table}[(((\text{CRC}[i] \gg 8) \wedge \text{Byte}) \& 0xff)]) \& 0xffff)$$

The resulting CRC value indicates the hash value for use with the integrity check. Different CRC functions are used with different protocols. For example, CRC-CCITT is used with the X.25 network protocol, and CRC-16 is used by LHA compression. CRC-32 is used by the Zip compression system as well as Ethernet and FDDI data link protocols.

4.5.4 Cryptographic Hash Functions

Parity, checksum, and CRC do provide integrity checking, but they also have a high probability for generating hash collisions. *Cryptographic hash functions* use much larger bit spaces. Rather than relying on complex polynomials, these functions rely on S-boxes to encode the data. Cryptographic hash functions generate a *digest*, or fingerprint, from the data. Although collisions are possible, different data generally have different hash values. Unlike the simpler integrity functions, the impact from a single bit change is not identifiable without running the cryptographic hash.

MD5 and SHA1 are examples of cryptographic hash functions. The *Message Digest algorithm #5* (MD5) generates a 128-bit digest [RFC1321]. The US Secure Hash Algorithm #1 (SHA1) generates a 160-bit digest and was intended to replace MD5 [RFC3174]. MD5 and SHA1 are very common for security-oriented systems. Other hash functions, such as MD2, MD4, Tiger, SNEFRU, HAVAL, and RIPEMD-160 are less common.

Hash and Crash Collisions

Cryptographic hash functions are intended to be nonpredictive. Although two sets of data can generate the same digest, finding two sets of data is supposed to be nondeterministic. An attacker should not be able to easily modify data and create a specific hash value. For this reason, many protocols rely on MD5 and SHA1 to validate information. If the information does not match the digest, then the data cannot be validated.

In August 2004, Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu published a paper that discussed how to compute collisions for many cryptographic hash functions, including MD5 [Wang2004]. The attack reduces the number of permutations needed to create a specific hash value. In theory, an attacker can compromise a data set and modify the data in such a way that the MD5 digest does not change. This type of modification would not be detected. In December 2004, Dan Kaminsky released a proof-of-concept program that did just that: it modified a file, creating two distinct files that generate the same MD5 digest [Kaminsky2004].

5. Sparse Hash Mappings

The largest risk to cryptographic hash functions comes from sparse hash mappings. Algorithms such as MD5 and SHA1 work very well for large data sets but offer little protection for small data sets. For example, if the data set is known to consist of five lowercase letters, then there are only 11,881,376 combinations (a relatively small number). Given the hash value, the data can be determined by testing all the data combinations. In any situation where the data set is smaller than the size of the digest, the hash mapping becomes *sparse*—few of the possible hash values are ever used. Because SHA1 generates a 160-bit digest (20 bytes), caution should be used when using SHA1 to validate less than 20 bytes of data. Similarly, if the data set is limited (e.g., only containing letters or numbers), then it simplifies the search process for an attacker.

Ultimately Weak

Internet Relay Chat (IRC) is an application layer protocol that allows real-time text communication. (IRC was the predecessor to Instant Messaging.) Using IRC, individuals and groups can talk to each other in real time, but, they can also see each other's network address. Many denial-of-service attacks were originally developed as a way to disable or inconvenience other people on IRC channels.

There are many different IRC servers. Some offer cloaking capabilities, where the user's network address is hidden from plain sight. UltimateIRCd 3.0.1 (<http://freshmeat.net/projects/ultimateircd/>) is one such IRC server. Rather than seeing the IP address 10.1.5.6, users see the cloaked value fbb8a3c.16b4fba6.11de379d.20846cb9. A regular user is unable to identify the IP address from the cloaked value; the cloak protects users from being targeted by denial-of-service attacks.

UltimateIRCd uses MD5 to perform the cloaking. The IP address a.b.c.d becomes MD5(d).MD5(c).MD5(b).MD5(a). Normally, MD5 is a strong cryptographic hash function. In this case, however, it is used to protect sparse data. IP addresses only use 256 digits. An attacker can precompute all 256 possible MD5 hash values and then look up any address.



On the CD-ROM is a copy of UnUltimate.c—an uncloaking program that reverses the UltimateIRCd 3.0.1 server's MD5 cloak. The server code is also provided.

6. Signed Cryptographic Hashes and HMAC

Cryptographic hash functions such as MD5 and SHA1 are well defined. Any recipient of the data can validate the contents. A *message authentication code* (MAC) is

any cryptographic hash function that uses a key to generate a permutation. When using a MAC, only recipients with the key can validate the data with the digest.

There are many ways to implement a MAC. The simplest method encrypts the hash with a key. Only key holders can decrypt the data and validate the hash. Using this approach, a parity, checksum, or CRC that is encrypted can be good enough to validate the data.

An alternate approach uses a *hashed message authentication code* (HMAC) [RFC2104]. This approach mixes the key with two cryptographic hash calls:

```
Hash(Key XOR outterpad, Hash(Key XOR innerpad, Data))
```

The inner pads and outer pads (`innerpad` and `outterpad`) ensure that two different key values are used. Also, passing one digest into a second hash function increases the variability. Listing 4.3 provides an example using MD5 as the hash function.

LISTING 4.3 HMAC Using MD5

```
void Hmac_MD5 (char Key[64], char Digest[64],
               char *Data, int DataLen)
{
    MD5_CTX context;
    char innerpad[64], outterpad[64];

    /* XOR key with innerpad and outterpad values */
    for(i=0; i<64; i++)
    {
        innerpad[i] = Key[i] ^ 0x36;
        outterpad[i] = Key[i] ^ 0x5c;
    }

    MD5Init(&context); /* initialize the inner context */
    MD5Update(&context,innerpad,64) /* MD5 the inner key */
    MD5Update(&context,Data,DataLen); /* MD5 the data */
    MD5Final(Digest, &context); /* Compute digest */

    MD5Init(&context); /* initalize the outer context */
    MD5Update(&context,outterpad,64); /* MD5 outer key */
    MD5Update(&context,digest, 16); /* MD5 with previous digest */
    MD5Final(digest, &context); /* Compute new digest */
} /* Hmac_MD5 */
```

For most network protocols, the integrity check is passed along with the packet. Protocols that depend on parity, checksum, CRC, and basic cryptographic hash algorithms are vulnerable to modification. Because the algorithms are well known, an attacker can modify the data and send a valid hash value along with the data. By using an HMAC, tampered data will not be validated; the attacker cannot create a

valid HMAC without also having the key. HMAC is used by some of the most secure network protocols, including IPsec, IPv6, Secure Shell (SSH), SSL, and Kerberos.

6. CIPHERS

In cryptography, there is a clear distinction between encryption and encoding. *Encoding* is a translation from one format to another. Although encodings may not be immediately readable, they offer no protection with regards to information tampering or authentication. Common encoding systems include XOR, Rot13, Mime (or Base64), and UUencoding. More complicated systems include monoalphabetic and polyalphabetic ciphers, such as the Caesar cipher.

In contrast to encoding, *encryption* provides options for privacy and authentication. An encryption algorithm, such as DES or AES, is used to encode data in such a way that the contents are protected from unintended recipients.

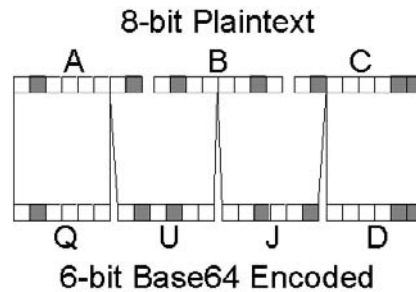
Encoding and encryption algorithms are *ciphers*. A cipher refers to any system that transforms data. Ciphers contain encoding and decoding components. Encryption and decryption functions provide the encoding and decoding functionality. In contrast, hashes are not ciphers because there is no decoding component.

1. Simple Ciphers

The simplest ciphers use a fixed algorithm for encoding and decoding data. For example, the *logical exclusive-or operator* (XOR) can perform a simple encoding. Each letter is encoded by performing a bit-wise XOR with a key. If the key is **ABC**, then the string **YES** becomes the ASCII characters **0+8**. If the string is larger than the key, then the key is repeated to cover the string.

Mime, or *Base64*, encoding is commonly used in email attachments. Email is not designed for transferring binary files. To prevent data corruption, binary files are Base64 encoded [RFC1421]. The Base64 algorithm converts every 3 binary bytes into 4 printable characters that can be safely transferred via email. The encoding algorithm converts three 8-bit blocks into four 6-bit blocks (Figure 4.1). Because 6 bits can hold 64 different combinations, they are mapped to letters (26 lowercase + 26 uppercase), numbers, and two symbols (+ and /). If the plaintext does not end on a 3-byte boundary, then equal signs are used at the end of the encoding. The decoding algorithm reverses this process.

XOR and Base64 are not secure ciphers. Each transforms data but can be readily reversed and provides neither privacy nor data validation. But, on occasion, they are found in places where security is desirable. For example, Web pages that use Basic Authentication for login access transmit a Base64-encoded username and password. Any eavesdropper can readily decipher the login credentials.



Base64 Encoding Table

0='A'	26='a'	52='0'
1='B'	27='b'	53='1'
2='C'	28='c'	54='2'
...
23='X'	49='x'	61='9'
24='Y'	50='y'	62='+'
25='Z'	51='z'	63='/'
Undefined is '='		

FIGURE 4.1 Base64 encoding.

2. Monoalphabetic and Polyalphabetic Ciphers

More complex ciphers perform character substitutions. Rot13 substitutes each letter with the letter offset by 13 characters. This makes Rot13 a shift substitution cipher—it shifts letters by 13 characters. The Caesar cipher, used by Julius Caesar to communicate with his generals, is another example of a shift substitution cipher. In a Caesar cipher, the number of characters shifted may vary between encodings.

The Vigenere cipher, proposed by Blaise de Vigenere in the sixteenth century, replaces letters with other letters but not by shifting the alphabet. For example, 'A' may map to 'G' and 'B' maps to 'W'. The letter mappings are consistent but not sequential.

Rot13, Caesar, and Vigenere ciphers are examples of *monoalphabetic* ciphers. Each replaces one *alphabet* (set of characters) with a different alphabet. A *polyalphabetic* cipher replaces one alphabet with multiple alphabets. For example, a polyalphabetic shift substitution cipher may use two alphabets. The original letters are alternately replaced using the different alphabets.

3. One-Time Pads

Monoalphabetic and polyalphabetic ciphers are vulnerable to character frequency attacks [FM34-40-2]. For example, the letter 'e' is the most common English

letter. A ciphertext encoding of English text will contain a character that repeats with the same frequency. One-time pads address this issue.

A *one-time pad* is a sequence of random characters that does not repeat. Each random character is combined with each plaintext character, resulting in a keyed ciphertext. Only a recipient with the same one-time pad can generate the same sequence of random characters and decode the message.

In the physical world, a one-time pad is literally a pad of paper with random characters and each sheet is used once. Only the intended recipients have identical paper pads.

In the digital world, a secret key can be used to seed a random number generator. Each character generated is combined using XOR (or a similar function) to encode and decode the ciphertext. Although XOR is normally insecure, the lack of key repetition makes a one-time pad more secure than other strong-encryption algorithms. The only weaknesses are the key size used to seed the random number generator and the “randomness” of the generated sequence. If the generator only takes a 16-bit seed, then there are only 65,536 different combinations. Similarly, if the generator does not create very random data, then the sequence may be vulnerable to a frequency analysis.

4.6.4 Book Ciphers

Book ciphers extend the one-time pad concept to a lookup table (*book*). The book represents the key and environment for encoding the data. For example, the ciphertext “1832 Z” cannot be decoded without using the proper book. In this case, the cipher indicates word offsets from the first paragraph of this chapter (decoding as “do not use”). Book ciphers cannot be decoded unless the recipient has a copy of the correct book.

Many compression algorithms operate like book ciphers, replacing a short ciphertext value with a longer plaintext value. In most cases, compressed files include the lookup table or information needed to recreate the table. But some systems use predetermined books that are not known to external attackers.

The simplest book ciphers use predetermined code lists. The lists determine the type of information that can be encoded. Any system that uses many abbreviations or numeric shortcuts is effectively a book cipher to outsiders.

4.7 ENCRYPTION

Encryption ciphers use keys to create permutations in the ciphertext. Encrypting a plaintext message with two different keys generates two different ciphertexts. Unlike cryptographic hash functions, encryption algorithms are intentionally reversible. Given a ciphertext and the correct key, the original plaintext can be derived.

The many different encryption algorithms are primarily divided into streaming and block ciphers, with block ciphers being more common. Block ciphers have a few limitations that are mitigated by different block cipher modes. Together, ciphers and modes provide a wide range of algorithms for selection. When comparing algorithms, key size and speed are the primary factors.

4.7.1 Streaming and Block Ciphers

Encryption algorithms fall under two categories: streaming and block. *Streaming ciphers* encode plaintext on a byte-by-byte basis—they operate on data streams. A one-time pad is an example of a streaming cipher. Another example is Rivest Cipher #4 (RC4). RC4 is a symmetrical key streaming cipher commonly used by wireless networks for WEP encryption (Chapter 7) and by SSL (Chapter 21). In each of these cases, the amount of data to encrypt varies, so a stream cipher's byte-by-byte encryption adds little to the transmitted packet size.

Block ciphers operate on data segments. For example, DES applies a 56-bit key to a 64-bit data block. If the plaintext size is smaller than 64 bits, then it is padded. But the algorithm itself only operates on 64-bit blocks. Table 4.2 lists common ciphers with their respective block and key sizes.

TABLE 4.2 Common Block Ciphers

Name	Key Size (bits)	Block Size (bits)
DES	56	64
IDEA	128	64
CAST-128	variable, up to 128	64
RC2	variable, up to 128	64
3DES (Triple-DES)	168	64
Blowfish	variable, 32 to 448	64
SEED	128	128
Serpent	128, 192, or 256	128
Twofish	128, 192, or 256	128
AES (Rijndael)	128, 192, or 256	128, 192, or 256

4.7.2 Block Cipher Modes

Block ciphers have a fundamental weakness: each block is encrypted independently. If a 64-bit sequence appears twice in the plaintext, then a 64-bit block cipher will

generate two identical 64-bit ciphertexts. To mitigate this issue, *modes* are used to add variability between blocks.

A variety of modes can be used to seed one block with the results of the previous block. This chaining allows a block cipher to emulate a streaming cipher and reduces the likelihood of a redundant ciphertext block. Common modes include the following:

Electronic Codebook Mode (ECB): ECB splits the plaintext into blocks, and encrypts blocks separately. There is no linking between blocks; each ECB ciphertext block is vulnerable to attack.

Triple ECB Mode: To strengthen weak keys, multiple keys can be used. In 3ECB, the plaintext is encrypted with the first key, decrypted with the second, and re-encrypted with the third.

Cipher Block Chaining Mode (CBC): The ciphertext from the previous block is XORed with the next plaintext block. This deters block-based attacks, but one error in a ciphertext block will corrupt the remainder of the message.

Cipher Feedback Mode (CFB): For the first block, an *initialization vector* (IV) is XORed with the plaintext before encryption. The IV is simply a large binary sequence (number, string, or array) that changes between each ciphertext encoding. In CFB mode, subsequent blocks use the previous ciphertext block in place of the IV. New IVs can be periodically selected to prevent data corruption from impacting an entire message. CFB is effective when the IV does not repeat with the plaintext. But, if the IV repeats with a duplicate initial plaintext block, then the ciphertext also duplicates.

Output Feedback Mode (OFB): For the first block, OFB encrypts the IV with the key. This is used to generate a sequence that is XORed with the first plaintext block (similar to a one-time pad). Subsequent blocks use the previous ciphertext block in place of the IV.

The variety of ciphers, key sizes, block sizes, and block modes allow encryption algorithms to be selected based on a best-fit model. Algorithms can be weighed based on speed, complexity, key sizes, corruption recovery, and other aspects.

Is Bigger Always Better?

Cryptographic attacks attempt to reduce the effective key strength. If some bits are used in relation to other bits, then finding one set of bits can be used to determine the related set. For example, *Triple-DES* (3DES) uses 3ECB to increase the key strength. Because the Data Encryption Standard (DES) uses one 56-bit key, 3DES uses three keys for a total of 168 bits. But 3ECB does

not always increase the key strength linearly. Although 3DES uses three keys, the effective key strength is only 112 bits. The attack method requires 7,000 TB (terabytes)—or 2^{56} bytes—of memory. Because this attack is currently not practical for most attackers, 3DES is applicable with 168-bit key strength and theoretically at 112bits.

The American Encryption Standard (AES) was introduced in 2000 as a replacement for DES. AES supports large keys and block sizes, and uses many S-boxes with few rounds. Although AES is large, it is relatively fast, requiring a low processing overhead. In contrast, 3DES uses a smaller key and block size with fewer S-boxes but takes much more processing time to encrypt and decrypt messages.

Similar to 3DES, AES was found to be vulnerable to a potential attack. In 2002, an attack was found that reduced the effective key size from 2^{256} to around 2^{100} [Courtois2002, Schneier2002]. Although AES does not appear to be as strong as originally expected, these theoretical attacks make AES about as effective as 3DES while retaining the performance improvements. (This was a huge finding in the cryptanalysis community, but it has little impact on practical usage.)

4.8 STEGANOGRAPHY

Although encryption is designed to provide authentication and privacy, it does not prevent attackers from observing traffic. An attacker may not know the contents of a data transfer but can see that a data transfer occurred. *Steganography* addresses the visibility-related risks from a data transfer. Encryption protects data by preventing readability—the data can be observed but not understood. Steganography prevents data from being seen. In steganographic encoding environments, the heuristics encode ciphertext, obscuring detectionefforts.

Steganography applies camouflage to information. Although forum-specific information is expected, subtle modifications can be used to convey an entirely different message. For example, data may be hidden in images so that the image does not appear noticeably different. Image steganography may encode data within a few pixels. A casual observer will not notice any difference to the image. More complicated algorithms store data in unused bits within the file, or modify image compression rates, index tables, and internal data structures.

Beyond images, steganography can be used to store data in any format, including text. Text-based steganography may alter character spacing, punctuation, line breaks, and even word choice. Strong steganographic systems are undetectable by casual observers. For example, the first paragraph of this section (Section 4.8) ends

with a steganographic text message. In the last sentence of that first paragraph, the first letter of each word spells out a hidden message.

It is difficult to identify steganographic messages, but they can be found. Covert channels that use steganography have been observed and validated in forums such as newsgroups (NNTP), email and spam (SMTP), Web pages, and IRC. Secret messages have been seen hidden in images, text, HTML, and even in packet headers for IP, TCP, and UDP. Steganography may use any part of the environment for camouflage. For example, the message may not be in the network packets but rather in the timing between packets. Steganography can even use keys to create permutations within the selection of hiding places. When steganography is combined with cryptography, the resulting ciphertext becomes very difficult to observe, and even if it is seen, it remains undecipherable.

The largest limitation to steganography is the camouflage size. If a 10 KB image is modified to store a few bytes of data, then few people will notice. But if the same image is modified to store 500 KB of information, then the change will likely be noticed. This is similar to the encryption problem when the key is significantly smaller than the data. Without taking special precautions, an encryption algorithm may fall victim to a pattern repetition attack. Similarly, too little camouflage allows detection of the steganographic message.

CHAPTER 2

Data Encryption Techniques

2.1 INTRODUCTION

In today's world nothing is secure. Nobody gives you the guarantee of 100% security. So, there is a need to protect our computer and data from the attackers. Due to rapid increase in the use of internet, every individual as well as his/her information is having a threat from the attackers. Some information which is beneficial to an individual or a group may be used against them by the attacker. In the companies or business organisations due to huge competition, the security measures are very important to protect the data/information. These security measures include authentications, encryptions, access control, confidentiality, etc.

Encryption is the process of converting the original information which is in meaningful and readable form (in cryptography we called it as plaintext) into unreadable form (in cryptography we called it as ciphertext). Encryption process requires a key for this conversion. The process of converting the ciphertext into plaintext is called *decryption*. Decryption is the reverse process of encryption. Decryption process also uses a key for conversion. There are a number of algorithms available for encryption. Depending upon the number of key/keys used encryption is divided into two types:

1. Symmetric encryption
2. Asymmetric encryption

We will discuss these methods in detail in the next section. A model used for encryption and decryption process is called a *cryptosystem*. The area of study in which one can study various techniques of encryption is known as *cryptography*. There are various techniques available to derive the plaintext or decrypt the ciphertext without much knowledge about the key and plaintext. This process is called *cryptanalysis* or *breaking the code*. The areas of cryptography and cryptanalysis together are called *cryptology*. Figure 2.1 explains encryption and decryption process.

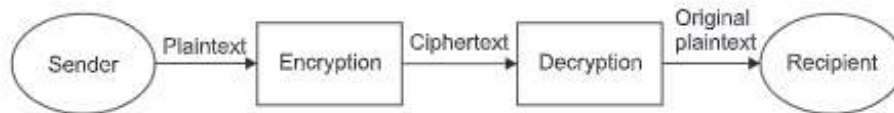


Figure 2.1 Encryption and decryption.

2.2 ENCRYPTION METHODS

The encryption algorithms are comparatively simple. The same encryption algorithm can be used for decryption but the subkeys should be used in reverse order. Encryption algorithms are classified into two types:

1. Symmetric encryption
2. Asymmetric encryption also called public key cryptography

2.2.1 Symmetric Encryption

An encryption technique in which only one key is required for encryption as well as decryption is called *symmetric encryption*. It is also called conventional encryption technique. As the name indicates, symmetric encryption algorithms use same key for encryption as well as for decryption.

For example, two friends A and B want to communicate with each other. They agreed on a symmetric encryption algorithm and a secret key. Friend A first encrypt the message by using the encryption algorithm and a secret key. Then he sends this encrypted message to B. The recipient B uses the same key and algorithm to decrypt the message. The detail graphical representation of this procedure is shown in Figure 2.2.

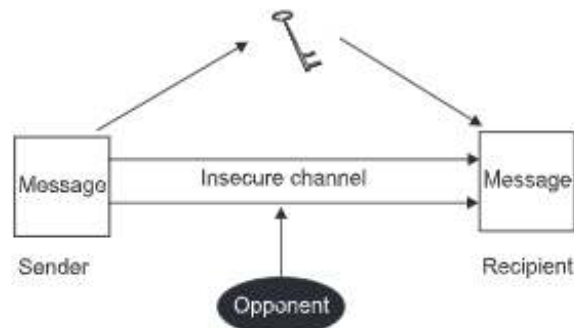


Figure 2.2 Symmetric encryption and decryption.

Most of the people use locks to keep the home secure. Similarly, symmetric encryption technique is used to provide the security to our message. To provide the security to our home we use a lock of some company to close the door. The same key is required to open and lock the door. If we use another key of the same company to open or close the door, it cannot work for that lock. Similarly, in symmetric key encryption,

the same key is required for encryption as well as decryption of the message. A few well-examined encryption algorithms that everyone could use just like the model of the lock may be the same, but the keys are different. The security of the home depends upon the quality of the lock, in the same way the security of our encryption algorithm depends upon the key.

Symmetric encryption and decryption techniques have various components as shown in Figure 2.3. The components of symmetric encryptions are:

1. **Plaintext:** The original message written or created by the sender is called plaintext. It is used as input for the encryption algorithm.
2. **Encryption algorithm:** There are various algorithms available for encryption. Using one of the algorithms we can encrypt the message, i.e., convert the plaintext to ciphertext.

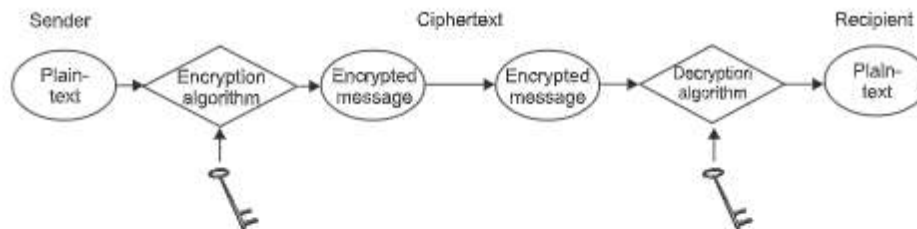


Figure 2.3 Components of symmetric encryption and decryption.

3. **Key:** Key is nothing but the pattern of alphabets/numbers used to convert the plaintext into ciphertext. In symmetric encryption, the same key is used for encryption and decryption. The security of any encryption algorithm depends upon the key. To provide more security, use new key for every new message.
4. **Ciphertext:** The encryption algorithm converts the plaintext into unreadable form using the key. This output is called *ciphertext*. The same key always produces the same cipher for the same plaintext. Whereas different keys will produce different ciphertexts for the same plaintext.
5. **Decryption algorithm:** The algorithm used to convert the ciphertext into plaintext is called the *decryption algorithm*. It uses the same key for symmetric encryption. Same encryption algorithm can be used for decryption but in reverse order.

2.2.2 Asymmetric Encryption

In asymmetric encryption, two different keys are used, one key for encryption and other key for decryption. These keys are mathematically related to each other. Sometime in asymmetric encryption, each user uses two keys called public key and private key. Public key is publically available so that not only sender and recipient but any body may know the key. Another key is private key which is a secrete key and known to the origimator (owner) of the key. Asymmetric cryptography is also known as public key cryptography. Asymmetric encryption algorithms cannot be decrypy easily. For asymmetric encryption algorithms key distribution is not required as each user have

their own keys. Therefore, public key cryptography provides more security as compared to symmetric encryption. We will see more detail about this in section 7.2.

2.3 CRYPTOGRAPHY

Cryptography is the practise of mathematical scrambling of word. Different encryption techniques are used for this purpose.

Parameters used by cryptographic systems are:

- *Operations used:* Encryption algorithms use various operations to convert plaintext into ciphertext. These include substitution, transpositions, etc. In substitution operation, one element in the plaintext is replaced by another element. In transposition operation, the order of rearrangement of the element is done. Most of the encryption algorithms use substitutions and transpositions.
- *Key:* Symmetric encryption technique use only one (same key) key for encryption and decryption. Whereas asymmetric encryption, two keys are required. Asymmetric encryption is also called *public key encryption*.
- *Types of processing:* Encryption techniques are classified into two types, stream cipher and block cipher, depending on processing. In *block cipher* the input plaintext is divided into a number of blocks. Each block having fixed number of elements. Then at a time one block is processed and the ciphertext is generated as a block having the same number of elements. In *stream cipher*, plaintext is processed one bit at a time. One bit of plaintext is converted into one bit of ciphertext at a time.

2.4 SUBSTITUTION CIPHERS

Classical encryption techniques are divided into two basic types: substitution ciphers and transposition ciphers.

In the substitution ciphers, one element of plaintext is substituted by other element. These ciphers are also called *monoalphabetic ciphers*. Example of this cipher is Caesar cipher. In some ciphers, the group of bits are replaced by another group of bits. These ciphers are also called *polygraphic substitution ciphers*. Examples of this cipher are Hill cipher and Playfair cipher.

In the following section, we discuss different monoalphabetic and polygraphic substitution ciphers.

2.4.1 The Caesar Cipher

The Caesar cipher is the oldest and the simplest substitution cipher. In this cipher, the ciphertext is generated by shifting each letter from the plaintext by same distance. It was first proposed by Julius Caesar so known as *Caesar cipher*. He used this cipher for his private communication. He used to replace each element in the plaintext by a shift of 3, so the plaintext letter PT_i is enciphered as ciphertext letter CT_i such as:

$$CT_i = E(PT_i) = (PT_i + 3) \bmod 26$$

In this cipher, each alphabet is numbered such as $a = 0$; $b = 1$ $z = 25$. As there are total 26 letters mod 28 is used to convert the last three letters such as x , y and z into a , b and c respectively.

The conversion of each element of plaintext into ciphertext using Caesar cipher is given below.

Plaintext	a	b	c	d	e	f	g	h	i	j	k	l	m
Ciphertext	D	E	F	G	H	I	J	K	L	M	N	O	P
Plaintext	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Using this encryption, the message **work patiently** would be encoded as:

Plaintext	w	o	r	k	p	a	t	i	e	n	t	l	y
Ciphertext	Z	R	U	N	S	D	W	L	H	Q	W	O	B

Advantages

- This cipher (encryption algorithm) is easy to implement.
- This cipher is very simple.

Disadvantages

- Brute force attack is easily possible.
- Its observable pattern helps the attacker to find out plaintext easily.
- Maximum number of keyspace (total number of keys) are 25 which can easily find out.

2.4.2 Monoalphabetic Ciphers

The monoalphabetic cipher is also known as a *cryptogram*. The KEY for this cipher is generated by doing the rearrangement of the alphabets. These different alphabets are then substituted for the alphabets in the plaintext. The result is a ciphertext. The same KEY is used to generate the plaintext from the ciphertext. The monoalphabetic cipher can be a permutation of the 26 alphabetic characters. So there are only $26!$ or greater than 4×10^{26} possible keys. This large number of keys help to eliminate the brute force attack. Suppose for each alphabet we assign the key as shown below:

Plaintext	a	b	c	d	e	f	g	h	i	j	k	l	m
Key	P	H	V	E	S	J	B	Y	O	T	G	N	X
Plaintext	n	o	p	q	r	s	t	u	v	w	x	y	z
Key	C	R	I	Z	L	W	M	A	F	K	Q	D	U

Now, suppose the plaintext is "we are the best" then, the ciphertext is as below:

Plaintext	w	e	a	r	e	t	h	e	b	e	s	t
Ciphertext	K	S	P	L	S	M	Y	S	H	S	W	M

Here the ciphertext is "KSPLSMYSHSWM". It is meaningless and also very difficult to the attacker to break as compared to Caesar cipher.

Cryptanalysis of Monoalphabetic Ciphers

The cryptanalysis of the monoalphabetic ciphers is easy as the ciphertext reflects the frequency count of the original message. In the above example, letter "S" is occurring 4 times as compared to other letters. If we study the frequency analysis of the English letters we observe that some letters such as "e, a, I" occurs more than letters like "j, z, x". This frequency analysis of the English language is used to perform the cryptanalysis. So monoalphabetic cipher is not more secure.

2.4.3 Playfair Cipher

Playfair cipher is the well-known encryption algorithm. It divided the plaintext into a group of two letters each. Each group is treated as a single unit. Using the key, groups of plaintext corresponding ciphertext groups are generated. The key generation procedure is discussed below. As the Playfair cipher uses groups of two letters to generate the ciphertext, it is a block cipher of block size two. The total encryption process is divided into three parts:

1. Preparing the Plaintext
2. Preparing the Key
3. Encryption

Preparing the Plaintext

The message is first converted into lower case, remove the punctuations and then split it into a group of two letters each. If any group has the same letters, then split that group by adding extra letters like x between these two letters. For example, if some group having the letters ll, then we can split it like lx and second letter l is paired with the first letter of the next pair and then all the letters are shifted by one position to the right. If j is present all j's are replaced with i's. This particular example contains no j's. We will illustrate this in more detail from the following example.

Message: We live in a world full of beauty.

Step 1 Convert this message into lowercase letters and remove punctuations.

weliveinaworldfullofbeauty

Step 2 Split the text into a pair of two.

we li ve in aw or ld fu **ll** of be au ty

If the last group is having only one letter, then append any one letter in that group to make a pair. If both the letters in a pair are same, then split this pair by adding any letter in between the letters and rearrange the groups. In this example, one of the pairs having same letters "ll" (shown in bold). Add letter "x" in between the letters, so the group is "lx". But the group should be of two letters, so shift the last letters of this group to the right by one position and rewrite the groups again.

we li ve in aw or ld fu lx lo fb ea ut y

Here the last group is having only one letter, so append one more letter to complete the pair. Here we append "z" with the last letter "y" as shown below:

we li ve in aw or ld fu lx lo fb ea ut yz

Step 3 Now, write the groups such that in one row 6 pairs are there as shown below:

we	li	ve	in	aw
or	ld	fu	lx	lo
fb	ea	ut	yz	

Now, the plaintext is ready to encrypt.

Preparing the Key

Read the key having any number of letters. Remove the duplicate letters from the key if any. Convert all the letters of the key into uppercase letters. To prepare the key, 5×5 matrix is constructed. We illustrate the key preparing procedure as shown below:

Suppose the key is "another"

Step 1 Convert the key into uppercase letters, the key becomes

ANOTHER

Step 2 Write the letters in the 5×5 matrix form, i.e., 5 letters in one row as shown below:

A	N	O	T	H
E	R	R		

Step 3 The remaining letters of the alphabet which are not present in the key are filled in the alphabetical order as shown below:

A	N	O	T	H
E	R	B	C	D
F	G	I/J	K	L
M	P	Q	S	U
V	W	X	Y	Z

As we have to form 5×5 matrix and there are total 26 alphabets, we have to omit one alphabet. Generally we select such alphabet which occur less in the language, i.e., the frequency of that letter is less. Here we omit j. Still if j occurs in the plaintext, i is the replacement for j in this particular example. If the key length is long, the message is more secure. But cryptanalysis of the Playfair is easy as compared to modern ciphers.

Encryption

The next step is encryption where the two letters (a pair) of the plaintext is encrypted at a time. Take any pair of letters from the plaintext. The letters in the pair compare with the 5×5 key matrix. The letters in a pair may be in the same row, in the same column, or in different rows and columns of the key matrix. The encryption procedure is illustrated using the following steps:

Step 1 Read a pair of letters from the prepared plaintext. If both the letters of pair are on the same row, then each letter of a pair is replaced by the letter to

the right of that letter. If the letter in a pair is the last letter (rightmost) on the row, then replace it with the first letter of the same row.

Suppose the plaintext pair is "nt", then corresponding ciphertext is "OH". Here n and t are on the same row, so we select the right side letters from the key matrix. The right side letter of n is O and the t is H. Therefore "OH" is the ciphertext for "nt". Similarly if the plaintext is "yw", then y becomes Z and w becomes X, so "ZX" is the ciphertext for "yw". If the plaintext pair is "rd", then r is replaced by B and d is replaced by E (as there is no letter to the right of d, the first letter from the same row, i.e., E, is selected). The ciphertext for "rd" is "BE".

Step 2 If both the letters of pair are in the same column, then each letter is replaced by the next letter (i.e., letter below the plaintext letter) in the same column. If the letter in a pair is the last letter in the column, then replace it with the first letter of the same column.

Suppose the plaintext pair is "em", then corresponding ciphertext is "FV". Here e and m are in the same column, so we select the next letter in the same column from the key matrix. The next letter of e is F and the m is V. Therefore "FV" is the ciphertext for "em". If the plaintext pair is "cy", then c is replaced by K and y is replaced by T (as there is no letter below y in the same column, the first letter from the same row, i.e., T, is selected). The ciphertext for "cy" is "KT".

Step 3 If both the letters of pair are neither in the same row nor in the same column, then the substitution for plaintext pair is based upon their intersection in the key matrix. Take the first letter from the plaintext pair. Locate its position in the key matrix. Then move across the row, i.e., left or right until it is lined up with the second letter in a pair. Then start with the second letter and move up and down the column until it is lined up with the first letter. The letters at the intersections are the ciphertext for the said pair.

Suppose the plaintext pair is "gs", then corresponding ciphertext is "KP". We have to apply above steps on all the pairs of the plaintext. The ciphertext for the above plaintext is:

VRFKAFGONVNBULLMIZIHIEFESHZY

In this cipher, there are $26 \times 26 = 676$ diagrams. The identification of individual diagrams is more difficult, and also the frequencies of individual letters have greater ranges which provide more security to this cipher.

Decryption

For decryption, the reverse process we have to follow.

Step 1 Break the ciphertext into pairs of letters:

VR	FK	AF	GO	NV
NB	UL	LM	IZ	IH
IE	FE	SH	ZY	

Step 2 Same as encryption, write down the alphabet square with the key “ANOTHER”:

A	N	O	T	H
E	R	B	C	D
F	G	LJ	K	L
M	P	Q	S	U
V	W	X	Y	Z

Step 3 Read a pair of letters from the prepared ciphertext. If both the letters of pair are on the same row, then each letter of a pair is replaced by the letter to the left of that letter. If the letter in a pair is the first letter (leftmost) on the row, then replace it with the last letter of the same row.

Step 4 If both the letters of pair are in the same column, then each letter is replaced by the previous letter (i.e., letter above the plaintext letter) in the same column. If the letter in a pair is the first letter in the column, then replace it with the last letter of the same column.

Step 5 If both the letters of pair are neither in the same row nor in the same column, then the substitution for plaintext pair is based upon their intersection in the key matrix. Take the first letter from the plaintext pair. Locate its position in the key matrix. Then move across the row, i.e., right or left until it is lined up with the second letter in a pair. Then start with the second letter and move down and up the column until it is lined up with the first letter. The letters at the intersections are the ciphertext for the said pair.

Transform the pairs of letters in the opposite direction from that used for encryption:

WE	LI	VE	IN	AW
OR	LD	FU	LX	LO
FB	EA	UT	YZ	

This message is now readable, although removing the extra spaces and substitutions for double letters makes it more readable:

We live in a world full of beauty.

Cryptanalysis of the Playfair cipher is easy, as for the same pair of letters always converted into a same pair of ciphertext.

2.4.4 The Hill Cipher

The next classical cipher for encryption is the Hill cipher. It is a polygraphic substitution cipher. The Hill cipher is based on linear algebra. Lester S. Hill was invented this cipher in 1929. It was the first cipher in which more than three symbols operate at a time.

Working

The Hill cipher uses the basic matrix multiplication. So, the alphabets are converted into numbers. Each letter from A to Z is assigned a digit from 0 to 25 such as A = 0, B = 1, C = 2, and Z = 25. As there are total 26 letters, the base is used as 26.

The total encryption process is divided into three parts:

1. Preparing the Plaintext
2. Preparing the Key
3. Encryption

Preparing the Plaintext: First each letter in the message is converted into numbers such as a = 0, b = 1 and so on. Then the numbers should be written in columnar form. The number of letters in each column depends on the key matrix size. Suppose the key matrix is 2×2 , then each column of plaintext having two elements only. For 3×3 size, then each column having 3 numbers. If the last column contains less elements then append necessary numbers to complete the last column. For example, for a key size of 3×3 , the total number of elements in the plaintext should be multiple of 3, otherwise append necessary elements to make it multiple of 3. Suppose there are 19 elements in the plaintext, then in each column there are 3 elements but in the last column there is only one element. So append 2 more elements in the last column.

Consider the message 'COE'. Since 'C' is 2, 'O' is 14 and 'E' is 4, the plaintext is:

$$P = \begin{bmatrix} 2 \\ 14 \\ 4 \end{bmatrix}$$

Preparing the Key: Every letter in the key is also assigning the number like message. Then the numbers should be written in row wise. The number of letters in each row depends on the key matrix size. Suppose, the key matrix is 2×2 , then each row having two elements only. The key matrix is always square matrix.

Consider the key "ANOTHERBZ": which is converted into numbers as: 0 13 14 19 6 4 17 1 25. Then write these numbers in matrix form as:

$$K = \begin{bmatrix} 0 & 13 & 14 \\ 19 & 6 & 4 \\ 17 & 1 & 25 \end{bmatrix}$$

Encryption: The encryption is the multiplication of key matrix and plaintext matrix. The number used for the letters are base 26. So mod 26 is used to generate the ciphertext.

$$\text{Ciphertext} = \text{Key} \times \text{Plaintext} \text{ mod } 26$$

$$C = KP \text{ mod } 26$$

Thus, the encryption is:

$$C = \begin{bmatrix} 0 & 13 & 14 \\ 19 & 6 & 4 \\ 17 & 1 & 25 \end{bmatrix} \begin{bmatrix} 2 \\ 14 \\ 4 \end{bmatrix} = \begin{bmatrix} 238 \\ 138 \\ 148 \end{bmatrix} \text{ mod } 26$$

$$C = \begin{bmatrix} 4 \\ 8 \\ 18 \end{bmatrix}$$

Here the numbers are reconvert into the letters, so, 4 = E, 8 = I, 18 = S. So the ciphertext is 'EIS'.

Decryption

To convert the ciphertext into plaintext again we have to perform matrix multiplication. The inverse of key matrix is multiplied by the ciphertext matrix to generate the plaintext matrix. To calculate the inverse of the key matrix, we use standard methods with extended Euclidean algorithm. The extended Euclidean will be discussed in Chapter 6. The decryption is shown below:

$$P = K^{-1} \times C \text{ MOD } 26$$

$$K^{-1} = \frac{1}{6453} \begin{bmatrix} -146 & 311 & 32 \\ 407 & 238 & -266 \\ 83 & -211 & 247 \end{bmatrix} \text{ mod } 26$$

$$K^{-1} = \frac{1}{6453} \begin{bmatrix} -16 & 25 & 6 \\ 17 & 4 & -6 \\ 5 & -3 & 13 \end{bmatrix} \text{ mod } 26$$

Here first compute $6453 \text{ mod } 26 = 5$, then find the multiplicative inverse of 5 such that $5d \text{ mod } 26 = 1$, where d is the multiplicative inverse of 5.

$$K^{-1} = \frac{1}{5} \begin{bmatrix} -16 & 25 & 6 \\ 17 & 4 & -6 \\ 5 & -3 & 13 \end{bmatrix} \text{ mod } 26$$

$$26 = 5(5) + 1$$

$$1 = 26 - 5 \times (5) = 26 + (-5)(5)$$

Therefore, the multiplicative inverse of 5 is -5 or 21.

Now,

$$K^{-1} = 21 \begin{bmatrix} -16 & 25 & 6 \\ 17 & 4 & -6 \\ 5 & -3 & 13 \end{bmatrix} \text{ mod } 26$$

$$K^{-1} = \begin{bmatrix} -24 & 5 & 22 \\ 19 & 6 & -22 \\ 1 & -11 & 13 \end{bmatrix} \text{ mod } 26$$

$$K^{-1} = \begin{bmatrix} 2 & 5 & 22 \\ 19 & 6 & 4 \\ 1 & 15 & 13 \end{bmatrix} \text{ mod } 26$$

Now, our ciphertext "EIS" is multiplied by this new key, we get:

$$P = \begin{bmatrix} 2 & 5 & 22 \\ 19 & 6 & 4 \\ 1 & 15 & 13 \end{bmatrix} \begin{bmatrix} 4 \\ 8 \\ 18 \end{bmatrix} \text{ mod } 26 = \begin{bmatrix} 2 \\ 14 \\ 4 \end{bmatrix}$$

The plaintext we get back is 'COE'.

One should take precaution while the selection of the key as every matrix does not have an inverse. We can find out this directly. If the determinant of the matrix is 0 then the inverse of that matrix is not exist. If the matrix has common factors with the modulus then the inverse of that matrix is not exist. Here we use 26 as modulus so the matrix having common factors of 2 and 13 cannot be used. So discard such matrix and select new key.

Security

The Hill cipher use matrix multiplication, therefore the ciphertext letter generated for a letter in plaintext is not dependent upon only a single plaintext letter but it is a combination of many letters. This helps to avoid the letter frequency problem. It is therefore difficult for cryptanalysis and provides more security. But this cipher uses linear algebra, which makes easy for a known plaintext attack.

Cryptanalysis of the Hill Cipher: We know that the Hill cipher is vulnerable to chosen plaintext attack. We will see how the key can be determined by considering Example 2.1.

EXAMPLE 2.1 Suppose the plaintext is P , ciphertext C , therefore $P = C = Z(2/4)$ is given. The plaintext be "ATTACK", and the message is encrypted using $a = 0$, $t = 1$, $c = 2$, and $k = 3$. The ciphertext is "CATKAC". Determine the key used to encrypt the plaintext "ATTACK".

Solution We know that $P = C = Z_4^2$. It indicates that there are two rows in the matrices and the base is 4. So, the letters in the message are numbered from 0 to 3.

Assume that the key may be $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and

the plaintext matrix may be $\begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \end{bmatrix}$ and

the ciphertext matrix may be $\begin{bmatrix} 2 & 1 & 0 \\ 0 & 3 & 2 \end{bmatrix}$

For Hill cipher, $C = K \times P \text{ mod (base)}$,

Here using the above matrices we get:

$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & 3 & 2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \end{bmatrix}$$

Let us consider the first two letters of the message, i.e., "AT". The first column of plaintext matrix $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ represents these two letters. The second pair of letters is

“TA” = $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and the third pair is $\begin{bmatrix} 2 \\ 3 \end{bmatrix}$. Thus we try to find out the key by solving for one pair of letters at a time. We solve this for the above data:

$$K \times P = C$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

Therefore, $b = 2$ and $d = 2$

$$\text{Similarly, } \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

Therefore, $a = 1$ and $c = 3$.

$$\text{From the above, key matrix is } \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

2.4.5 Polyalphabetic Ciphers

Above ciphers are simple substitution ciphers. They are not secure as the cryptanalysis of these ciphers are easy due to frequency analysis. If the ciphertext is sufficiently large, cryptanalysis can be made. Given a sufficiently large ciphertext, it can easily be broken. For large ciphertext, it is possible to find out the letter frequencies easily. Therefore, to provide more security and solve the problem of frequency analysis, there is a need to design a new cipher. Polyalphabetic cipher solves this problem. In the polyalphabetic cipher, a single letter of plaintext can be converted to several different letters of the ciphertext instead of just one letter.

The well-known polyalphabetic substitution cipher is Vigenere cipher. In this cipher, a set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers. The Caesar cipher uses the shift of 3 places whereas Vigenere cipher uses shifts of 0 to 25. The key is the permutations of the alphabet called a Vigenere square. In this cipher, there are 25 rows and 25 columns. 25 rows of this square can be used as keys, where the row number gives the amount it is shifted.

There are two different methods to form a polyalphabetic cipher from Vigenere ciphers. In the first method during encryption, select all the 25 rows of the Vigenere square one by one. Therefore, every 25th letter is encrypted with the same key. In the second method, a key is created which gives the order of the rows to be selected. This means only selected rows are used instead of all 25 rows of the Vigenere square. For example, the key is created could be $K = (5, 2, 16)$ and then repetition of these three rows is done. It means every third letter is encrypted with the same key. In this cipher each single letter of plaintext is encrypted using only one key. Therefore, it works like monoalphabetic ciphers.

The Vigenere cipher, proposed by Blaise de Vigenere in the 16th century, is a polyalphabetic substitution cipher based on Table 2.1.

Table 2.1 The Vigenere table

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

There are 25 rows and 25 columns in the Vigenere square. Each row of the table corresponds to a Caesar cipher. There is a shift of 0 positions in the first row, a shift of 1 position in the second row, a shift of $n-1$ position in the n^{th} row. For example, the message is:

She is very happy and beautiful girl.

And the key is 'another'.

To encrypt this message, first write the key. The letters of the key is repeated as many times as the length of the plaintext. Write the plaintext below the key so that one letter of the plaintext is below the one letter of the plaintext.

Keyword:	anoth	erano	thera	nothe	ranot	heran
Plaintext:	sheis	veryh	appya	ndbea	utifu	lgirl

The ciphertext is generated using the Vigenere square. Key letters indicate the row and plaintext letters indicate the column. First find the intersection of the row and column using each letter of the key with corresponding letter in the plaintext. Note down all the intersections. This gives you the ciphertext.

The ciphertext generated using Vigenere cipher is shown below:

Keyword:	anoth	erano	thera	nothe	ranot	heran
Plaintext:	sheis	veryh	appya	ndbea	utifu	lgirl
Ciphertext:	SUSBZ	ZVRLV	TWTPA	ARULE	LTVTN	SKZRY

The decryption also followed the same procedure. First write the letters of the key in the same way as we write for encryption. Then write the ciphertext below the letters of key as shown below:

Keyword:	anoth	erano	thera	nothe	ranot	heran
Ciphertext:	SUSBZ	ZVRLV	TWTPA	ARULE	LTVTN	SKZRY

For decryption, the key letters indicate the column and ciphertext letter indicate row. Select the column for a key letter and then select the row correspond to the plaintext letter. The intersection is the plaintext for the corresponding ciphertext letter. The plaintext generated is shown below:

Keyword:	anoth	erano	thera	nothe	ranot	heran
Ciphertext:	SUSBZ	ZVRLV	TWTPA	ARULE	LTVTN	SKZRY
Plaintext:	SHEIS	VERYH	APPYA	NDBEA	UTIFU	LGIRL

In Vigenere cipher, for the same plaintext there are multiple ciphertext. This helps to avoid the frequency analysis of the cipher and makes the cipher secure. For example, in the above plaintext there are 3 e's that they have been encrypted by 'S,' 'V,' 'L', respectively. This helps to hide the count of occurrence of e in the plaintext. So, it makes frequency analysis of the letters in the plaintext difficult. The implementation of this cipher is easy.

Cryptanalysis of the Vigenere Cipher

The Vigenere cipher is secure from the attack using frequency analysis. But it is not completely secure cipher. If the attacker is able to find out the length of the key, then frequency analysis is possible. The chosen-plaintext attack is possible against this cipher.

2.4.6 One-time Pad or Vernam Cipher

In 1918, Gilbert Verman developed a cipher called as *one time pad*. It is the most secure cryptographic algorithm. In this cipher, the key is a set of random numbers generated by pseudo-random number generator. This generator is used only once to encrypt a message. One-time pad and key is used for decryption. Mauborgne developed a method of one-time pad. A one-time pad is a very simple *symmetric* cipher. The key is selected randomly so that every time new key is used for encryption. Therefore, for the same message next time different ciphertext is generated. So, it is difficult to break this cipher. For decryption, same key is used, so secure key transmission is the problem.

Properties of One-time Pad

1. The number of possible keys is equal to the number of possible plaintexts.
2. The key is selected at random.
3. Key should be used only once.

Working of the Cipher

The key or one-time pad is a string of characters or numbers. User should note that no part of the key be reused again. Length of the one-time pad is equal to the length of the message. Then perform XOR operation between the plaintext (message) and one-time pad (key). The result is ciphertext.

$$\begin{aligned}\text{Encryption: } C_i &= P_i \oplus K_i & i &= 1, 2, 3, \dots \\ \text{Decryption: } P_i &= C_i \oplus K_i & i &= 1, 2, 3, \dots\end{aligned}$$

where

P_i : plaintext bits
 K_i : key (key-stream) bits
 C_i : ciphertext bits

EXAMPLE 2.2 Encrypt the following message using one-time pad and then decrypt the encrypted message.

Message: WE LIVE IN A WORLD FULL OF BEAUTY

The key is given as:

Key: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Solution:

PLAINTEXT	W	E	L	I	V	E	I	N	A	W	O	R	L	D	F	U	L	L	O	F	B	E	A	U	T	Y
	22	04	11	8	21	4	8	13	0	22	14	17	11	3	5	20	11	11	14	5	1	4	0	20	19	24
OTP KEY	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
RESULT	22	5	13	11	25	9	14	20	8	31	24	28	23	16	19	35	27	28	32	24	21	25	23	43	43	49
MOD 26	22	5	13	11	25	9	14	20	8	5	24	3	23	18	19	9	1	2	6	24	21	25	23	17	17	23
CIPHERTEXT	W	F	N	L	Z	J	O	U	I	F	Y	C	X	Q	T	J	B	C	G	Y	V	Z	W	R	R	X

The ciphertext is “WFNLZJOUIFYCXQTJBCGYVZWRRX”

Decryption of the above is derived as shown below:

CIPHERTEXT	W	F	N	L	Z	J	O	U	I	F	Y	C	X	Q	T	J	B	C	G	Y	V	Z	W	R	R	X
	22	5	13	11	25	9	14	20	8	5	24	2	23	16	19	9	1	2	6	24	21	25	23	17	17	23
OTP KEY	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
RESULT	22	4	11	8	21	4	8	13	0	-4	14	-9	11	3	5	-6	-15	-15	-12	5	1	4	0	-6	-6	-2
MOD 26	22	4	11	8	21	4	8	13	0	22	14	17	11	3	5	20	11	11	14	5	1	4	0	20	20	24
PLAINTEXT	W	E	L	I	V	E	I	N	A	W	O	R	L	D	F	U	L	L	O	F	B	E	A	U	T	Y

Number of bits in the one-time pad and the plaintext are same. Therefore, the length of the key is large. This is the drawback of this cipher because it is difficult to remember or make the transmission of key securely. But it also makes this cipher more secure.

2.5 TRANSPOSITION CIPHERS

One more classical encryption cipher is transposition ciphers. In transposition cipher, the letters are written in a row under the key and then arrange the column as per alphabetical order. There are two types of transposition ciphers: single columnar and double columnar transposition ciphers.

2.5.1 Single Columnar Transposition

Single columnar transposition cipher is the simple cipher. Read the key, and numbered each letter of the key as per their appearance in the alphabet. The total encryption process is divided into three parts:

1. Preparing the Key
2. Preparing the Plaintext
3. Encryption

Preparing the Key: Suppose the key is 'another'. We can assign the number to each letter in this key as shown below:

a	n	o	t	h	e	r
1	4	5	7	3	2	6

That is, the first letter 'a' is numbered 1. There are no B's or C's, so the next letter to be numbered is the 'e'. So e is numbered 2, followed by h, and so on. In the key if the same letter has occurred more than one time, it should be numbered 1, 2, 3, etc. from left to write. For example, the key is "heaven". Here 'e' is occurred two times. So first 'e' from left hand side is numbered as 2, whereas second 'e' is numbered as 3.

h	e	a	v	e	n
4	2	1	6	3	5

Preparing the Plaintext: The letters from the message is written in rows under the numbered letters of the key. One letter from message is to be written under each letter of the key. Let us say that the message is "we are the best". We can write it as shown below:

h	e	a	v	e	n
4	2	1	6	3	5
W	E	A	R	E	T
H	E	B	E	S	T

Encryption: Now, arrange the above message written in rows under the numbered letters of the key as per ascending order of the numbers at the top of the plaintext letters.

a	e	e	h	n	v
1	2	3	4	5	6
A	E	E	W	T	R
B	E	S	H	T	E

Then the letters are copied down column wise from top to bottom. The result is ciphertext, i.e.,

ABEEESWHTTRE

For decryption, first calculate the number of letters present in the ciphertext. Using the number of letters in the key, we can calculate the number of letters present in the last row. As it can be seen above, all the columns contain only two letters and this is important. In the above example, there are 12 letters and the key having 6 letters, so there are two rows and the last row have 6 letters. This gives us the idea about number of rows and number of letters in each column. Here there are two rows and each row having two ciphertext letters. For decryption, the key is prepared as for encryption. Then write the first two letters below the column number '1'.

```

h   e   a   v   e   n
4   2   1   6   3   5
      A
      B

```

Next two letters below column number two.

```

h   e   a   v   e   n
4   2   1   6   3   5
      E   A
      E   B

```

Next two letters below column number 3 and so on. In this way write all the letters from ciphertext. It will look like this:

```

h   e   a   v   e   n
4   2   1   6   3   5
W   E   A   R   E   T
H   E   B   E   S   T

```

Now, write down the letters in row wise, the result is the plaintext as below:

WEARETHEBEST

Separate the words by spaces, we will get the message, i.e.,

WE ARE THE BEST

2.5.2 Double Columnar Transposition

The single columnar transposition cipher is not much secure. To provide stronger transposition cipher, double columnar transposition is used. The working of double columnar cipher is similar to the single columnar transposition, but the process is repeated twice. Here we can use either the same key both times or two different keys. Suppose the plaintext is "we are the best", and the keys are "heaven" and "another".

```

h   e   a   v   e   n
4   2   1   6   3   5
W   E   A   R   E   T
H   E   B   E   S   T

```


This first encryption gives: ABEEESWHTTRE. These letters are written under the second key, thus we get:

a	n	o	t	h	e	r
1	4	5	7	3	2	6
A	B	E	E	E	S	W
H	T	T	R	E		

If the last row like the above example, having less letters than the first row, then we can add some more letters to complete the row. But this reduces the security of the cipher. So, one can encrypt the plaintext by not adding any dummy letters in the last row. The ciphertext is as below:

AHSEEBTETWER

The double columnar transposition cipher uses two keys so it is stronger than the single columnar transposition. The cryptanalysis of double columnar is difficult as compared to that of single columnar cipher.

2.6 CRYPTANALYSIS

The study of methods of breaking the ciphers is called *cryptanalysis*. In cryptanalysis, the cryptanalyst tries to search for flaws or loopholes in the design of the ciphers. The cryptanalyst guesses the key and a very large number of guesses for the key were incorrect, we can ensure that the cipher is stronger and not easily breakable. If the message length is long and the key size is small, cryptanalysis is easy.

In the transposition cipher, the order of letters in the plaintext is shuffled around. However, if the key length is short, then it is possible to break the ciphers. In a columnar transposition cipher, the message is written row wise. The number of letters in a row is fixed and equal to the length of a key. The ciphertext is generated by noting down the letters column wise form a block of letters in each column. Then permutation is performed on the letters in each block. This gives the key and helps to break the cipher.

2.6.1 Enumerate All Short Keywords

The cryptanalysis is started by trying all possible short keys. For a key of length up to 9 characters/letters we can do this very fast. For permutation of every key we find out same text. Select the most meaningful text as our plaintext. The possible number of rearrangements for a key of length N is $N!$ (N factorial). If the key length increases then the number of rearrangements increases very fast. For various key lengths the possible keys and its numbers are shown in Table 2.2.

Up to key length 6, it is easy to test all possible combinations of the keys. If the key having length more than 6 letters, it is difficult to try all possible combinations to find out the key. So for better security, the key length should be large enough.

Table 2.2 Number of possible keys for various length keywords

Key length	Possible keys	No. of permutations
2	AB, BA	2
3	ABC, BAC, CBA, ...	6
4	ABCD, ABDC, ACBD, ...	24
5	ABCDE, ABCED, ...	120
6	ABCDEF, ABCDFE, ...	720
7	ABCDEFG, ABCDFEG, ...	5,040
8	ABCDEFGH, ...	40,320
9	ABCDEFGHI, ...	362,880
10	ABCDEFGHIJ, ...	3,628,800
11	ABCDEFGHIJK, ...	39,916,800
12	ABCDEFGHIJKL, ...	479,001,600

2.6.2 Dictionary Attacks

If the above cryptanalysis method failed, we now try for another method. Generally the columnar transposition cipher uses a key with a word or a phrase. So instead of checking all possible keys, we may only go to check only common words. For this a dictionary should be prepared. It must consist of a large list of words such as place names, famous people names, mythological names, historical names, etc. From this dictionary we generate a text file of possible keys. Select the words having length greater than 6 letters as a key, as the keys having length up to 6 is checked by above method. Now use these words to decrypt the ciphertext. Note down the key with meaningful plaintext. This method can work if the key was one of the words that are present in the dictionary, but it fails to work if the key is something like 'THECOLDINOCTOMBER'. It is not possible to include all the places or phrases in the dictionary.

Ciphertext attack is not that simple for transposition cipher. Suppose we obtain the ciphertext as:

Ciphertext: GSMOEVMTFEFMTYPPEIRSPIOEVIIEOMP

We assume that this is encrypted by columnar transposition using a key. Cryptanalyst tries to find out the key and the plaintext by following the steps given below:

- Step 1** Count the number of letters in the ciphertext. There are 30 letters in the ciphertext.
- Step 2** Assuming the possible dimension of an array. The array could have any of the following dimensions: 6×5 or 5×6 or 10×3 or 3×10 . Suppose that we first try a 6×5 array. This helps us to guess the key length. For this array, the key length is 6. Then the ciphertext array is as shown in Table 2.3.
- Step 3** Now, observe the top row of the array in Table 2.3 and try to find the meaningful word. As per this word, identify the columns and permute these columns as shown in Table 2.4. After permutation, we observe the word

GIVE in the first row of Table 2.4. Also there are words or partial words in the other rows of Table 2.4. If this gives meaning statement then the key is almost recovered.

In this example, the encryption key is 152634 and the plaintext is **GIVE ME SOME TIME TO PROVE MYSELF**.

Table 2.3 Ciphertext array

1	2	3	4	5	6
G	V	M	E	I	E
S	M	T	I	O	E
M	T	P	R	E	O
O	E	Y	S	V	M
E	F	P	P	I	P

Table 2.4 Permuted ciphertext array

1	5	2	6	3	4
G	I	V	E	M	E
S	O	M	E	T	I
M	E	T	O	P	R
O	V	E	M	Y	S
E	L	F	P	P	P

Therefore, the plaintext is “GIVEMESOMETIMETOPROVEMYSELFPPP”. Here last three characters are “PPP” with no meaning. So we remove “PPP” and the plaintext is “**GIVE ME SOME TIME TO PROVE MYSELF**”.

2.7 STEGANOGRAPHY

The art and science of hiding information by embedding message within another message is called *steganography*. Steganography helps to keep the message secret. In this technique, the useless or unused bits are replaced with bits of the information which we want to hide. This can be done on the text/graphics files, encrypted messages or images. Steganography can be used to hide text or images. It is used to support the encryption.

2.7.1 Applications

Steganography can be used for legitimate as well as illegitimate purposes.

Legitimate Purposes

1. For copyright protection, one can insert watermark into the original image.
2. To tag notes to online images.
3. To maintain the confidentiality of valuable information.
4. To protect the data from unauthorised access.

Illegitimate purposes

1. For stealing the data.
2. Militants use this technique to send their message.

2.7.2 Limitations

Steganography also have certain drawbacks:

1. A lot of overhead is required to hide a relatively few bits of information.
2. Once the attacker knows the system, it becomes virtually worthless.

SOLVED PROBLEMS

- 2.1** Find the key for decryption using Hill cipher if the key for encryption is "DIMENSION".

Solution:

- Step 1** First convert the letters in the key in number form such as a = 0, b = 1, and so on.

Write the key in 3×3 matrix form in row wise.

$$K = \begin{bmatrix} d & i & m \\ e & n & s \\ i & o & n \end{bmatrix} = \begin{bmatrix} 3 & 8 & 12 \\ 4 & 13 & 18 \\ 8 & 14 & 13 \end{bmatrix}$$

- Step 2** Calculate the determinant of K .

$$\begin{aligned} \text{Det}(K) &= 3(13 \times 13 - 14 \times 18) - 8(4 \times 13 - 8 \times 18) + 12(4 \times 14 - 8 \times 13) \\ &= 3(169 - 252) - 8(52 - 144) + 12(56 - 104) \\ &= 3(-83) + 8(92) + 12(-48) \\ &= -249 + 736 - 576 \\ &= -89 \end{aligned}$$

- Step 3** Calculate the adjoint of K .

$$\text{Adj}(K) = \begin{bmatrix} -83 & 64 & -12 \\ 92 & -57 & -6 \\ -48 & 22 & 7 \end{bmatrix}$$

- Step 4** Find out the multiplicative inverse of 89 mod 26.

To find the multiplicative inverse, we use two methods:

1. $ab \text{ mod } 26 = 1$ where $a = -89$ and we have to find out the possible value of b by putting one by one value. $\text{GCD}(a, b) = 1$. We know that $-89 \text{ mod } 26 = 11 \text{ mod } 26$. Therefore, $a = 11$ and possible value of $b = 19$.

Modular multiplicative inverse of $-89 \text{ mod } 26 = 19$

2. Extended Euclidean algorithm for multiplicative inverse (refer Chapter 6 for more detail).

Step 5 Find the mod of adjoint matrix.

$$K^{-1} = \begin{bmatrix} -83 & 64 & -12 \\ 92 & -57 & -6 \\ -48 & 22 & 7 \end{bmatrix} \text{mod } 26$$

Step 6 Find the inverse of the matrix.

$$K^{-1} = \frac{1}{\det(K)} \text{adj}(K)$$

$$K^{-1} = \left(\frac{1}{-89}\right) \begin{bmatrix} -83 & 64 & -12 \\ 92 & -57 & -6 \\ -48 & 22 & 7 \end{bmatrix} \text{mod } 26$$

$$K^{-1} = \left(\frac{1}{89}\right) \begin{bmatrix} 83 & -64 & 12 \\ -92 & 57 & 6 \\ 48 & -22 & -7 \end{bmatrix} \text{mod } 26$$

$$K^{-1} = 19 \begin{bmatrix} 83 & -64 & 12 \\ -92 & 57 & 6 \\ 48 & -22 & -7 \end{bmatrix} \text{mod } 26$$

$$K^{-1} = (19) \begin{bmatrix} 5 & 14 & 12 \\ 12 & 5 & 6 \\ 22 & 4 & 19 \end{bmatrix} \text{mod } 26$$

$$K^{-1} = \begin{bmatrix} 95 & 266 & 228 \\ 228 & 95 & 114 \\ 418 & 76 & 361 \end{bmatrix} \text{mod } 26$$

Therefore, the key for decryption is,

$$K^{-1} = \begin{bmatrix} 17 & 6 & 20 \\ 20 & 17 & 10 \\ 2 & 24 & 23 \end{bmatrix}$$

- 2.2 Use Hill cipher to encrypt and decrypt the message, "I CANT DO IT". The key for encryption is "DIMENSION".

Solution:

Encryption: First convert the letters in the key and the plaintext in number form such as a = 0, b = 1, and so on.

$$K = \begin{bmatrix} \text{d} & \text{i} & \text{m} \\ \text{e} & \text{n} & \text{s} \\ \text{i} & \text{o} & \text{n} \end{bmatrix} = \begin{bmatrix} 3 & 8 & 12 \\ 4 & 13 & 18 \\ 8 & 14 & 13 \end{bmatrix}$$

$$\text{Plaintext } P = \begin{bmatrix} 8 & 13 & 14 \\ 2 & 19 & 8 \\ 0 & 3 & 19 \end{bmatrix}$$

$$C \text{ (Ciphertext)} = K \text{ (Key)} \times P \text{ (Plaintext)}$$

$$C = \begin{bmatrix} 3 & 8 & 12 \\ 4 & 13 & 18 \\ 8 & 14 & 13 \end{bmatrix} \begin{bmatrix} 8 & 13 & 14 \\ 2 & 19 & 8 \\ 0 & 3 & 19 \end{bmatrix}$$

$$C = \begin{bmatrix} 40 & 227 & 334 \\ 58 & 353 & 502 \\ 92 & 409 & 471 \end{bmatrix} \pmod{26}$$

$$C = \begin{bmatrix} 14 & 19 & 22 \\ 6 & 15 & 8 \\ 14 & 19 & 3 \end{bmatrix}$$

Convert the numbers into alphabets.

$$= \begin{bmatrix} O & T & W \\ G & P & I \\ O & T & D \end{bmatrix}$$

Read the characters from the matrix in column wise vertically downward direction. The ciphertext is: OGOTPTWID

Decryption: The key for encryption is DIMENSION. So the key for decryption is as below: (For detail refer Problem 2.1)

$$P = K^{-1} \times C$$

$$P = \begin{bmatrix} 17 & 6 & 20 \\ 20 & 17 & 10 \\ 2 & 24 & 23 \end{bmatrix} \begin{bmatrix} 14 & 19 & 22 \\ 6 & 15 & 8 \\ 14 & 19 & 3 \end{bmatrix}$$

$$P = \begin{bmatrix} 554 & 793 & 482 \\ 522 & 825 & 606 \\ 494 & 835 & 305 \end{bmatrix} \pmod{26}$$

$$P = \begin{bmatrix} 8 & 13 & 14 \\ 2 & 19 & 8 \\ 0 & 3 & 19 \end{bmatrix}$$

$$P = \begin{bmatrix} I & N & O \\ C & T & I \\ A & D & T \end{bmatrix}$$

Read the matrix column wise, the plaintext is: "ICANTDOIT"

- 2.3 Use Hill cipher to encrypt and decrypt the message "ESSENTIAL". The key for encryption is "ANOTHERBZ".

Solution: First convert the letters in the key and the plaintext in number form such as a = 0, b = 1, and so on.

The key matrix is,
$$K = \begin{bmatrix} 0 & 13 & 14 \\ 19 & 6 & 4 \\ 17 & 1 & 25 \end{bmatrix}$$

The plaintext matrix is,
$$P = \begin{bmatrix} 4 & 4 & 8 \\ 18 & 13 & 0 \\ 18 & 19 & 11 \end{bmatrix}$$

Encryption:

Ciphertext matrix is, $C = K \times P \pmod{26}$

$$C = \begin{bmatrix} 0 & 13 & 14 \\ 19 & 6 & 4 \\ 17 & 1 & 25 \end{bmatrix} \begin{bmatrix} 4 & 4 & 8 \\ 18 & 13 & 0 \\ 18 & 19 & 11 \end{bmatrix} \pmod{26}$$

$$C = \begin{bmatrix} 486 & 435 & 154 \\ 256 & 230 & 196 \\ 536 & 556 & 411 \end{bmatrix} \pmod{26}$$

$$C = \begin{bmatrix} 18 & 19 & 24 \\ 22 & 22 & 14 \\ 16 & 10 & 21 \end{bmatrix} \pmod{26}$$

$$C = \begin{bmatrix} S & T & Y \\ W & W & O \\ Q & K & V \end{bmatrix} \pmod{26}$$

Read the characters from the matrix in column wise vertically downward direction. The ciphertext is: "SWQTIWKYOV"

Decryption:

Plaintext matrix is, $P = K^{-1} \times C \pmod{26}$

First we find out the key for decryption which is K^{-1} .

$$K^{-1} = \frac{1}{6453} \begin{bmatrix} -146 & 311 & 32 \\ 407 & 238 & -266 \\ 83 & -211 & 247 \end{bmatrix} \pmod{26}$$

$$K^{-1} = \begin{bmatrix} 2 & 5 & 22 \\ 19 & 6 & 4 \\ 1 & 13 & 13 \end{bmatrix}$$

$$\begin{aligned}
&= \begin{bmatrix} 2 & 5 & 22 \\ 19 & 6 & 4 \\ 1 & 13 & 13 \end{bmatrix} \begin{bmatrix} 18 & 19 & 24 \\ 22 & 22 & 14 \\ 16 & 10 & 21 \end{bmatrix} \pmod{26} \\
&= \begin{bmatrix} 498 & 368 & 580 \\ 538 & 533 & 624 \\ 512 & 435 & 479 \end{bmatrix} \pmod{26} \\
&= \begin{bmatrix} 4 & 4 & 8 \\ 18 & 13 & 0 \\ 18 & 19 & 11 \end{bmatrix} \\
&= \begin{bmatrix} \text{E} & \text{E} & \text{I} \\ \text{S} & \text{N} & \text{A} \\ \text{S} & \text{T} & \text{L} \end{bmatrix}
\end{aligned}$$

Read the characters from the matrix in column wise vertically downward direction. The plaintext is "ESSENTIAL"

- 2.4 The chosen plaintext attack is on Hill Cipher with $P = C = Z(2^7)$. Suppose the plaintext be "ESSENTIALA" and the message is encoded using: E = 0, S = 1, N = 2, T = 3, I = 4, A = 5 and L = 6. The ciphertext is "TNSLIHALEF". Find the key used in this cipher.

Solution: Assume the key be $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and

the plaintext matrix is $\begin{bmatrix} 0 & 1 & 2 & 4 & 6 \\ 1 & 0 & 3 & 5 & 5 \end{bmatrix}$ and

the ciphertext matrix is $\begin{bmatrix} 3 & 1 & 4 & 5 & 0 \\ 2 & 6 & 4 & 6 & 4 \end{bmatrix}$

We know that $C = K \times P$

$$\begin{bmatrix} 3 & 1 & 4 & 5 & 0 \\ 2 & 6 & 4 & 6 & 4 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 & 4 & 6 \\ 1 & 0 & 3 & 5 & 5 \end{bmatrix}$$

Let's consider the first two letters of the message, i.e., "ES". The first column

of plaintext matrix $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ represents these two letters. The second pair of letters

is "SE" = $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, the third pair of letters is "NT" = $\begin{bmatrix} 4 \\ 4 \end{bmatrix}$, the fourth pair of letters

is "IA" = $\begin{bmatrix} 4 \\ 5 \end{bmatrix}$, and the last pair of letters is "LA" = $\begin{bmatrix} 6 \\ 5 \end{bmatrix}$. Thus, we try to find

out the key by solving for one pair of letters at a time. We solve this for above data:

$$K \times P = C$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Solving this we get, $b = 3$ and $a = 1$

$$\text{Similarly } \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 6 \end{bmatrix}$$

Therefore $d = 2$ and $c = 6$.

From above the key matrix is $\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 6 & 2 \end{bmatrix}$

- 2.5 Use transposition cipher to encrypt and decrypt the message "MEET ME AT BOAT CLUB CANTEEN" using the key "EXAMPLE".

Solution:

E	X	A	M	P	L	E
M	E	E	T	M	E	A
T	B	O	A	T	C	L
U	B	C	A	N	T	E
E	N					

Here the last row is not fully filled, so we add any letter (but same letter in all columns) in the unfilled columns of the last row. Suppose we use letter "X" for this purpose. The matrix is now as below:

E	X	A	M	P	L	E
M	E	E	T	M	E	A
T	B	O	A	T	C	L
U	B	C	A	N	T	E
E	N	X	X	X	X	X

Give the numbering to the columns as per their alphabetical order.

E	X	A	M	P	L	E
2	7	1	5	6	4	3
M	E	E	T	M	E	A
T	B	O	A	T	C	L
U	B	C	A	N	T	E
E	N	X	X	X	X	X

Next arrange the table as per ascending order of the numbers given to each column.

A	E	E	L	M	P	X
1	2	3	4	5	6	7
E	M	A	E	T	M	E
O	T	L	C	A	T	B
C	U	E	T	A	N	B
X	E	X	X	X	X	N

Now note down the letters column wise. This is the ciphertext as below:

EOCXMTUEALEXECTXTAAXMTNXEBBN

For decryption, first calculate the number of letters present in the ciphertext. Using the number of letters in the key, we can calculate the number of letters present in the last row. As can be seen above, all the columns contain only four letters and this is important. In the above example there are 28 letters and the key having 7 letters, so there are four rows and each row has 7 letters. For decryption, the key is prepared as for encryption. Then write the first four letters below the column number '1'.

E	X	A	M	P	L	E
		1				
		E				
		O				
		C				
		X				

In the same way, fill all the columns one by one.

E	X	A	M	P	L	E
2		1				
M		E				
T		O				
U		C				
E		X				

and we get the output is as below:

E	X	A	M	P	L	E
2	7	1	5	6	4	3
M	E	E	T	M	E	A
T	B	O	A	T	C	L
U	B	C	A	N	T	E
E	N	X	X	X	X	X

Note down the letters row wise, we get the plaintext as:

MEETMEATBOATCLUBCANTEENXXX

From above text, the last three characters are XXX. This indicates that these characters are added just for padding and not the part of message. So remove these letters and insert the spaces in between the meaningful words. We get the final plaintext is as below:

MEET ME AT BOAT CLUB CANTEEN

- 2.6 Apply cryptanalysis on the following ciphertext which was encrypted using single columnar transposition.

Ciphertext: EOCKMTUEALEXECTXTAAXMTNXEBBN

Solution: We assume that this is encrypted by columnar transposition using a key. Cryptanalyst tries to find out the key and the plaintext by following the steps given below:

- Step 1** Count the number of letters in the ciphertext. There are 28 letters in the ciphertext.
- Step 2** Assuming the possible dimension of an array. The array could have any of the following dimensions: 7×4 or 4×7 or 14×2 or 2×14 . Suppose that we first try a 7×4 array. This helps us to guess the key length. For this array, the key length is 7. Then the ciphertext array is as shown in table given below:

1	2	3	4	5	6	7
E	M	A	E	T	M	E
O	T	L	C	A	T	B
C	U	E	T	A	N	B
X	E	X	X	X	X	N

2	1	4	5	6	7	3
M	E	E	T	M	E	A
T	O	C	A	T	B	L
U	C	T	A	N	B	E
E	X	X	X	X	N	X

- Step 3** Now, observe the top row of the array in the above table and try to find the meaningful word. As per this word, identify the columns and permute these columns as shown in table given below.

2	7	1	5	6	4	3
M	E	E	T	M	E	A
T	B	O	A	T	C	L
U	B	C	A	N	T	E
E	N	X	X	X	X	X

After permutation, we observe the word MEET in the first row of the table. Also there are words or partial words in the other rows of the table. If this gives meaning statement then, the key is almost recovered.

We start our cryptanalysis from column 2 using word MEET in the first row. But using this sequence we cannot get the meaningful message. So, again some rearrangement of the columns is needed. Therefore, we rearrange the sequence as per word "BOAT" from second row as shown in table given below:

Now we get the meaningful words and the encryption key 2715643. The plaintext is MEETMEATBOATCLUBXXX. Last three letters are XXX and have no meaning so delete those letters and the final plaintext is:

MEET ME AT BOAT CLUB.

2.7 The ciphertext given below was encrypted with substitution cipher:

AIIXGILHCHA

The rule for encryption is given as:

$$C = (P + K) \bmod 26$$

where C is the ciphertext, P is the plaintext and K is the key. Assume that the plaintext is in English. If the first plaintext letter is G, find the key and the plaintext.

Solution: We know that mod 26 is used means the alphabets are numbered as $a = 0, b = 1, c = 2, d = 3, e = 4, f = 5, g = 6, h = 7, i = 8, j = 9, k = 10, l = 11, m = 12, n = 13, o = 14, p = 15, q = 16, r = 17, s = 18, t = 19, u = 20, v = 21, w = 22, x = 23, y = 24, z = 25$.

Therefore, the ciphertext is: A = 0, S = 18, S = 18, X = 23, G = 6, S = 18, L = 11, H = 7, C = 2, H = 7, A = 0.

The plaintext $P = (C - K) \bmod 26$

The first plaintext letter is G = 6 is given and the first ciphertext letter is A = 0. Using formula given, we get:

$$6 = (0 - K) \bmod 26$$

Therefore, $K = -6$

Therefore, plaintext is:

$$0 - (-6) \bmod 26 = 6 = G$$

$$8 - (-6) \bmod 26 = 14 = O$$

$$8 - (-6) \bmod 26 = 14 = O$$

$$23 - (-6) \bmod 26 = 3 = D$$

$$6 - (-6) \bmod 26 = 12 = M$$

$$8 - (-6) \bmod 26 = 14 = O$$

$$11 - (-6) \bmod 26 = 17 = R$$

$$7 - (-6) \bmod 26 = 13 = N$$

$$2 - (-6) \bmod 26 = 8 = I$$

$$7 - (-6) \bmod 26 = 13 = N$$

$$0 - (-6) \bmod 26 = 6 = G$$

The plaintext is GOODMORNING.

CHAPTER 3

Data Encryption Standards

3.1 INTRODUCTION

Encryption techniques are useful to provide the confidentiality to the data. Encryption techniques are classified into two types: block encryption techniques and stream encryption techniques. This classification is based on the number of bits processed at a time. In block cipher, a block of fixed number of bits is processed at a time whereas in stream cipher, one bit is processed at a time. Block ciphers are faster than stream cipher. In this chapter, we will discuss a block cipher called data encryption standard (DES).

3.2 BLOCK CIPHERS

In the last chapter we have learnt about different classical encryption techniques. Encryption can be performed on a single bit/letter or a group of bits/letters. Therefore, there are two types of encryptions ciphers: a stream cipher and a block cipher.

When encryption algorithms process a block of data at a time and generate a block of data as a ciphertext, then it is called *block cipher*. Symmetric encryption ciphers/ algorithms use a block of bits for processing. The size of block is fixed for a particular algorithm. For example, data encryption standards (DES) uses a plaintext block of size of 64 bits, whereas advanced encryption standard (AES) uses a plaintext block of size of 128, 192 or 256 bits. For symmetric encryption, only one key is required for encryption. The same key is used for decryption.

When an encryption algorithm processes a single bit of data at a time and generates a single bit of data as a ciphertext, then it is called *stream cipher*. RC4 is a stream cipher.

In block cipher, the size of plaintext block and ciphertext block is same. When the number of bits in the plaintext/message are not multiple of the block size, then the technique is called modes of operation. There are various modes of operations such as electronic code book (ECB) mode, cipher block chaining (CBC) mode, feedback modes,

and counter mode. Symmetric encryption algorithms use any one of these modes. These modes of operation provide some advantages to the basic encryption algorithms.

Advantages of Modes of Operation

- For any symmetric encryption algorithm, the block size is fixed. But it is not necessary that the number of bits in the plaintext or message should always be multiples of the block size. In such case these symmetric encryption algorithms with modes of operation can encrypt and decrypt the message.
- Modes of operation help to provide an additional level of security to the encryption algorithm.
- Replay attack of packets can be avoided using modes of operation.

Section 3.3 discusses these modes of operation in detail.

3.3 BLOCK CIPHER MODES OF OPERATION

There are different modes of operation which help the encryption algorithms to process the encryption more easily. Some modes of operations are as below:

1. Electronic code book (ECB) mode
2. Cipher block chaining (CBC) mode
3. Feedback modes
4. Counter mode

3.3.1 Electronic Code Book (ECB) Mode

One of the simple modes of operation is the electronic code book mode. The input for this mode of operation is a plaintext block of 64 bits. The ciphertext block generated using ECB is also of 64 bits. In electronic code book mode, each plaintext block is processed independently of other plaintext blocks. The working of electronic code book mode for encryption is shown in Figure 3.1.

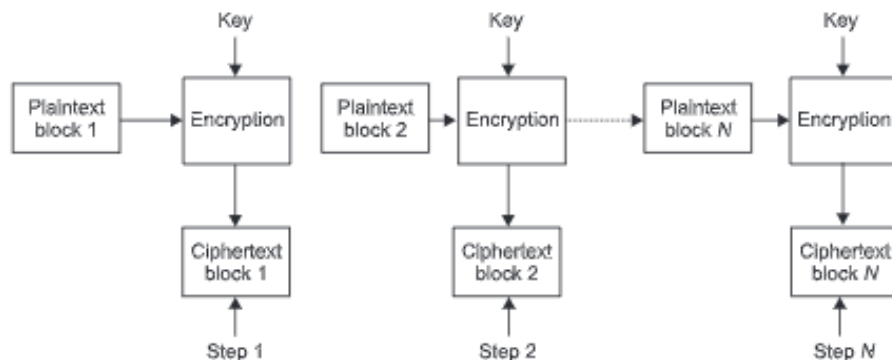


Figure 3.1 Electronic code book mode: Encryption.

The plaintext/message is first divided into blocks of size 64 bits. If the last block of plaintext/message is not having sufficient number of bits, i.e., 64, then the necessary number of bits is appended to the plaintext to complete the block size. Then process each block independently using the key. Same key is used for encryption of all the blocks. The ciphertext is generated which is unique for each plaintext block. But if two plaintext blocks are identical, then the ciphertext block generated are also same. This helps the cryptanalysis of the encryption easy. Here encryption algorithm may be DES, AES or any block cipher. Here we assume data encryption standard (DES). The security of this mode of operation is same as the encryption algorithm used.

Decryption using ECB is the reverse process of encryption using the same key. During decryption, two identical ciphertext blocks also produce the same plaintext blocks. The working of electronic code book mode for decryption is shown in Figure 3.2.

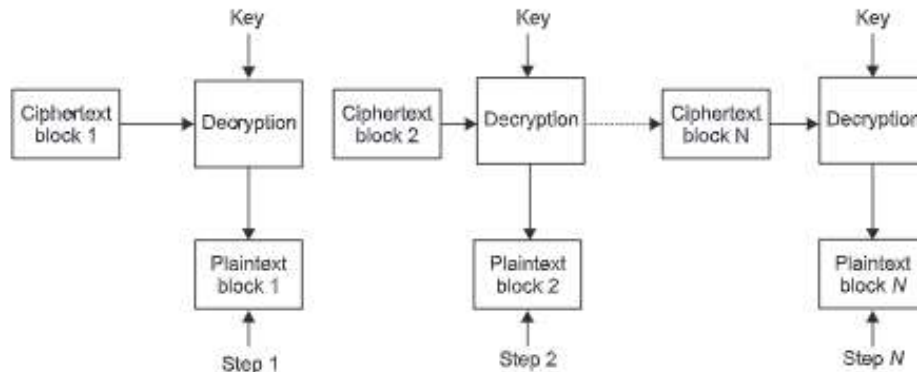


Figure 3.2 Electronic code book mode: Decryption.

Advantages

1. As we can process each block of plaintext independently of each other, we can process multiple blocks simultaneously.
2. If any plaintext or ciphertext blocks lost, it does not affect on the output of other blocks.
3. Parallel processing during encryption as well as decryption helps to increase the speed and the performance of the algorithm.

Disadvantages

If two plaintext blocks are identical, then the ciphertext block generated are also same. Therefore, known plaintext attack is possible.

3.3.2 Cipher Block Chaining (CBC) Mode

Electronic code book mode suffers by known plaintext attack. Cipher block chaining (CBC) overcomes this drawback of electronic code book mode. In CBC, like ECB, plaintext/message is first divided into blocks of size of 64 bits. If the last block of plaintext/message is not having sufficient number of bits, i.e., 64, then the necessary number of bits is appended to the plaintext to complete the block size. An initialisation

vector is selected. It is nothing but the random number which helps to increase the security. A key is used for encryption of all the blocks. Then perform XOR operation between the first plaintext block and the initialisation vector. The output is 64-bit block, which is encrypted using the secret key. Now, the output is 64-bit ciphertext.

The next block of plaintext is now XOR with the 64-bit ciphertext of the previous step. That is, for N^{th} plaintext, $(N-1)^{\text{th}}$ ciphertext block is used. Therefore, for the identical blocks of plaintext, different ciphertext blocks are generated. The encryption process is shown in Figure 3.3.

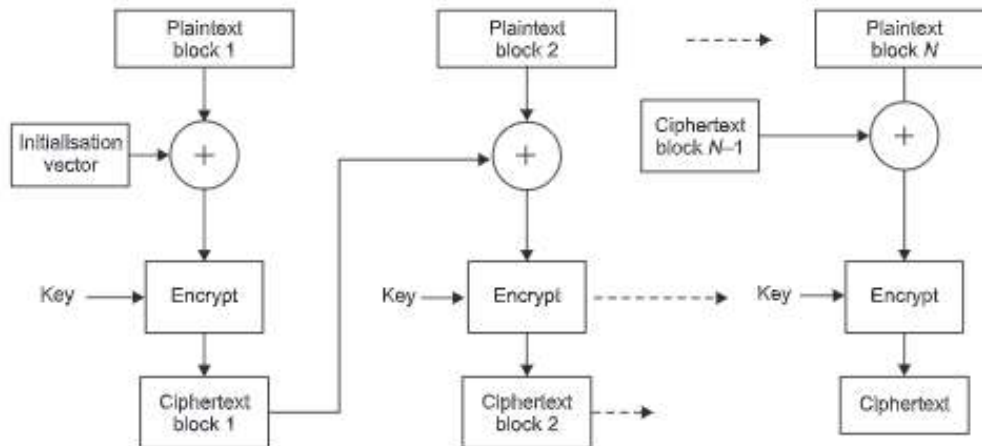


Figure 3.3 Cipher block chaining mode: Encryption.

Decryption is the reverse process of encryption. For decryption, the same key is used as encryption. First block of ciphertext is decrypted using the key. Then XOR operation is performed with the initialisation vector. The output is a plaintext. Same operation is repeated for each ciphertext blocks. Only difference is that instead of initialisation vector, plaintext block of previous step is used. The complete decryption process is as shown in Figure 3.4.

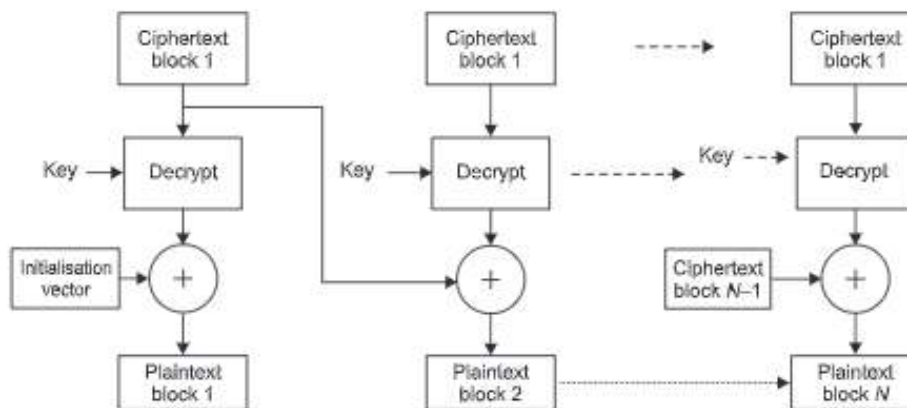


Figure 3.4 Cipher block chaining mode: Decryption.

Cipher block chaining can be used to generate the hash value. The last ciphertext block is used as hash value for the given message. Because the last ciphertext block is dependent on all the plaintext blocks. This hash value helps to check if any one of the ciphertext blocks lost or modified. CBC cannot use parallelisation, as the ciphertext of the first plaintext block is used for the processing of the next plaintext block.

Advantages

1. For identical blocks of plaintext, different ciphertext blocks are generated. So, CBC is more secure as compared to ECB mode.
2. Hash value, i.e., last ciphertext block, helps to identify if the message is original or modified.

Disadvantages

1. Parallel operation cannot be performed. So, it is slower as compared to ECB.
2. Lost/missing of any block of ciphertext stops the decryption process of the remaining blocks.

3.3.3 Feedback Mode

In the above two modes of operation, if the last block does not have sufficient number of bits, padding some bits is required. For feedback mode, this is not required. In this mode, the ciphertext of the previous step is given as input to the next step just like feedback. There are two types of feedback mode: cipher feedback mode and output feedback mode.

Cipher Feedback Mode

If the block size is smaller than the required block size, then the cipher feedback mode can be used to encrypt plaintext. The block size may be a bit or bytes, so there is no need of padding. The encryption process using cipher feedback mode is shown in Figure 3.5.

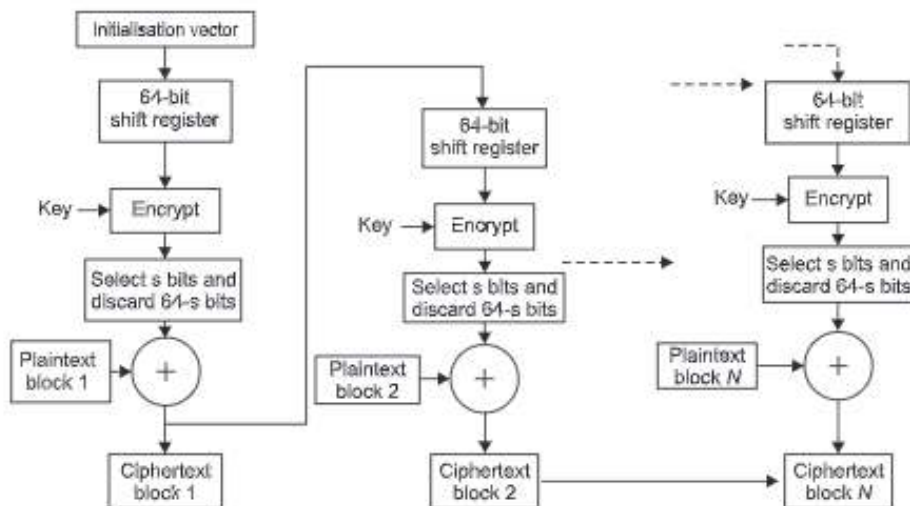


Figure 3.5 Cipher feedback mode: Encryption.

In this mode, 64-bit shift register is used. In the first step, shift register is filled by initialisation vector, i.e., random number. The output of the shift register is encrypted using a key of size 64 bits. Then select leftmost “s” bits from the 64 bits and discard remaining bits. “s” is equal to the number of bits in the plaintext block. Then XOR operation is performed between the plaintext and the “s” bits. The output is ciphertext of block size “s”. The ciphertext is also sent as input to the next step as input to the shift register. This process continues on all the blocks of plaintext to generate the ciphertext.

In decryption, initially the shift register is filled with initialisation vector. The initialisation vector is same as used for encryption. Then encrypt the bits in the shift register using the key and then select “j” bits from the 64 bits and XOR with the ciphertext. The output is plaintext. Figure 3.6 shows the decryption using cipher feedback mode.

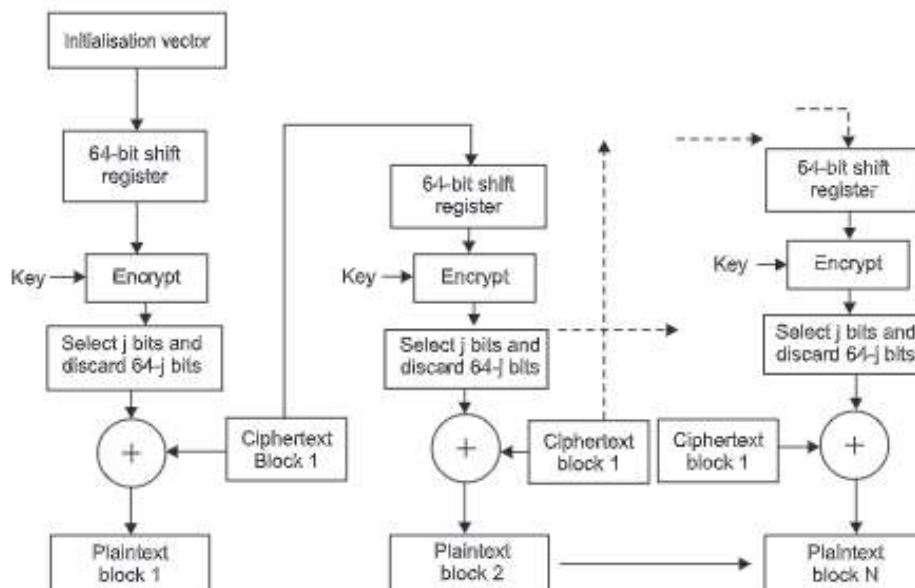


Figure 3.6 Cipher feedback mode: Decryption.

CFB is suffered from bit errors. If in the incoming cipher block, any one bit error is there, then it causes the bit error at the same bit position in the plaintext block. The same ciphertext block is used as input to the shift register of the next step and causes bit errors in the next plaintext block. In this way, the bit error in the shift error remains till the bit remains in the shift register. Suppose there is a bit error at 4th bit position of the 8-bit CFB, then subsequent 8 bytes will be garbled. After that the correct plaintext will be generated.

Output Feedback Mode

Another feedback mode is an output feedback mode. This mode can also process the plaintext block of any size less than or equal to 64 bits. In this mode instead of the

ciphertext, the encrypted output of the shift register is given as input to the shift register of the next step. The complete process of output feedback mode is shown in Figure 3.7.

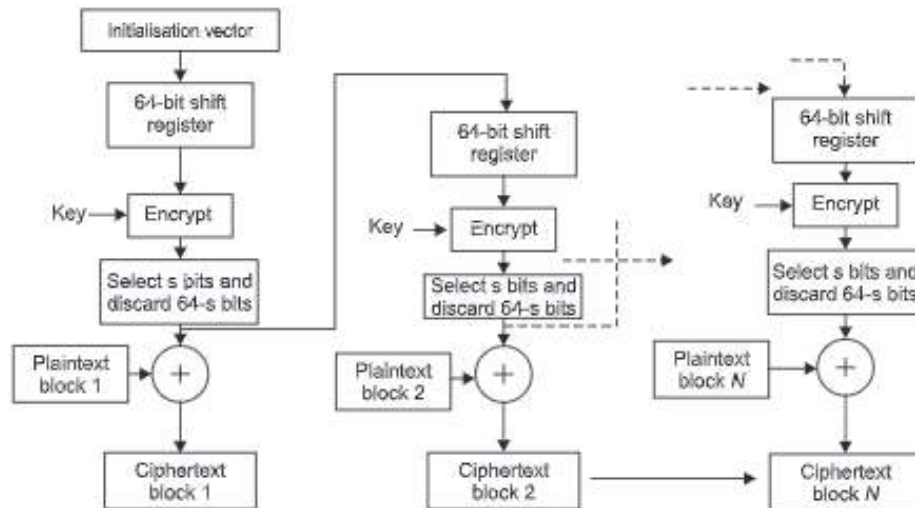


Figure 3.7 Output feedback mode: Encryption.

First step is to select 64-bit random value, called initialisation vector. This value is given as input to the 64-bit shift register. The encryption algorithm is used to encrypt the output of shift register with the key. Then select "s" (value of "s" is equal to the size of plaintext block) bits from the encrypted output and discard 64-s bits. Next step is performed XOR operation between the selected "s" bit and the plaintext block. The output is ciphertext. The selected "s" bits are used as input for the shift register on the next step. In this way, the processing of each block of plaintext is done.

Bit error is occurring in CBC and CFB due to an error in any one bit of the ciphertext. In output feedback, the encrypted output of the shift register instead of ciphertext block is used as input to the shift register of the next step. Here, ciphertext is not used for the processing of the next plaintext block. Therefore, output feedback mode is not suffered by bit error. But cryptanalysis of output feedback mode is easy as only a ciphertext block and encrypted "s" bits are sufficient to get the plaintext block. Here information about the key is not required, which help the cryptanalyst to break the cipher easily. Therefore, this mode is less secure than cipher feedback mode.

The decryption process of output feedback mode is similar to encryption. The only difference is that instead of plaintext block, corresponding ciphertext block is used for XOR operation. The detail decryption using output feedback mode is shown in Figure 3.8.

Modern stream ciphers are faster than block ciphers, this trim down the importance of the feedback modes.

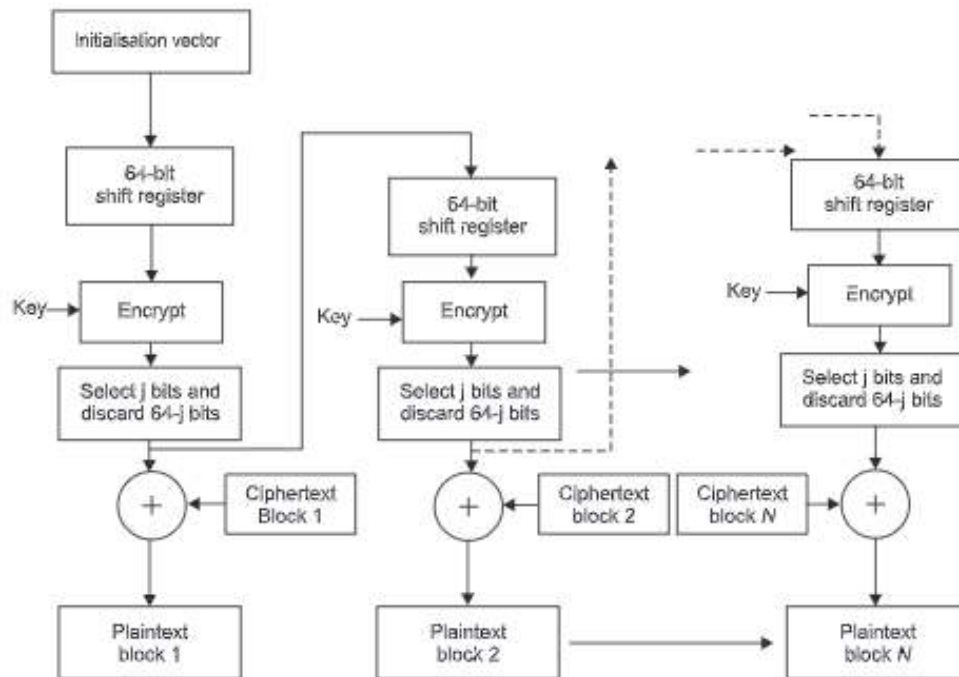


Figure 3.8 Output feedback mode: Decryption.

Advantage

- Free from bit error rate.

Disadvantage

- Vulnerable to a stream modification attack.

3.3.4 Counter Mode

The counter mode (CTR) is the most important mode of operation. In the counter mode, a block cipher is worked like a stream cipher. The counter is used whose value is changed in each round. Initially, the user has to set some value to the counter. The encryption algorithm (DES algorithm) processes the counter value and the key. This encrypted value is XOR with the block of plaintext. The result is a block of ciphertext. For the two identical blocks of plaintext, two different blocks of ciphertext are generated. Encryption process using the counter mode is shown in Figure 3.9.

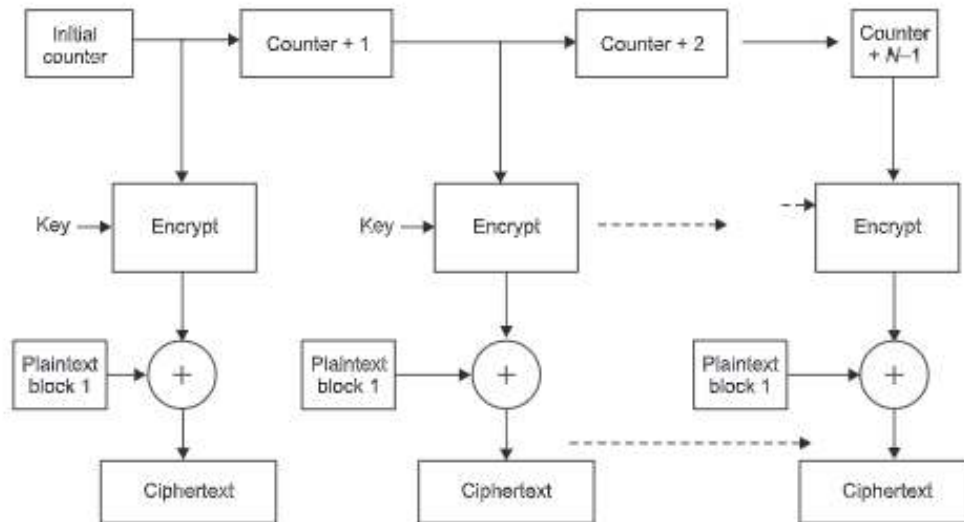


Figure 3.9 Counter mode: Encryption.

Decryption process is similar to encryption except ciphertext blocks are used instead of plaintext blocks. Key and the value of counter are same as encryption. Decryption process is shown in Figure 3.10.

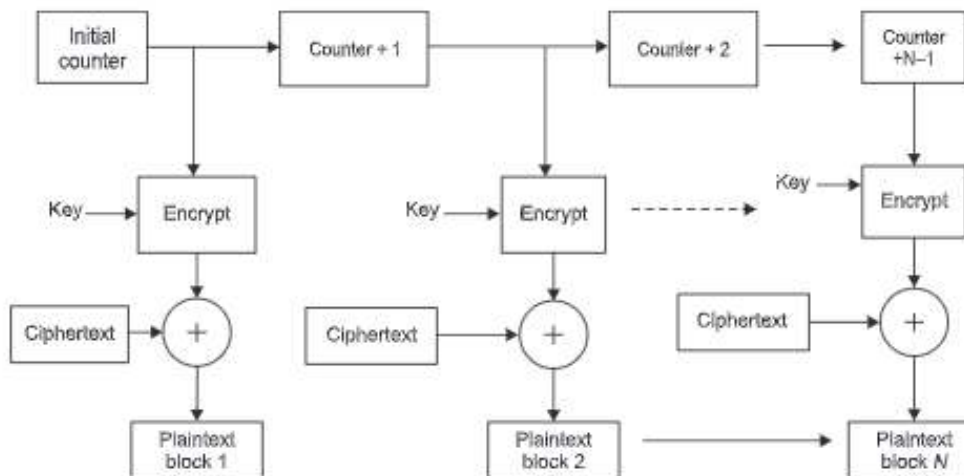


Figure 3.10 Counter mode: Decryption.

Advantages

1. The counter mode may be faster than of cipher block chaining mode.
2. Encryption can be done in parallel.

3. Padding is not required.
4. Processing of plaintext blocks can be done randomly.
5. Only encryption algorithm is required.
6. It is as secure as the other modes.

Disadvantages

1. Integrity of the message is not maintained.
2. Reuse of counter value, compromise the security.

3.4 FEISTEL CIPHERS

Feistel ciphers are a block cipher. In this, cipher iterations are carried on the blocks. In this cipher, the single plaintext block undergoes through different rounds repeatedly. Each round has a number of transformations. After going through a number of rounds, the ciphertext is generated. Feistel structure is the building block for many block ciphers. The design of data encryption standard (DES) algorithm is based on Feistel structures. Initially, the plaintext is split into the blocks of equal size. Then each block is split into two equal parts: left part and right part. The Feistel structure has many rounds and each round has different subkeys. These subkeys are generated from the key entered by the user. Different mathematical operations are performed on the right part of the plaintext with one of the subkeys. These mathematical operations include the hash or round function. The XOR operation is performed between the output of function and the left part of the plaintext.

Then the right and left parts are interchanged. This completes the round one of the Feistel cipher. For the next round, output of the previous round is used as input for the next round, i.e., left part of previous round is used as right part for the next round and right part is used as left part. This process is repeated for all the rounds. In the last round, there is no interchange of left and right part. Combine both the parts to get the ciphertext.

By reversing the order of subkeys(i.e., the last subkey of encryption is used for the first round of decryption), the same Feistel structure can be used for encryption and decryption. The complete operation of Feistel is shown in Figure 3.11.

The security of Feistel cipher depends on the key size and hash function. The design of Feistel cipher depends on following parameters:

1. *Block size:* Block size indicates the total number of bits in a block as DES algorithm uses the design of Feistel cipher. DES uses a block of 64 bits.
2. *Key length:* It is the length of key. DES uses 64-bit key.
3. *Number of rounds:* The security of any block ciphers depend on the number of rounds in the cipher. DES has 16 rounds, so every plaintext block undergoes 16 iterations.
4. *Subkeys:* Each round uses different keys called subkeys. These subkeys are derived from an original key.
5. *Round function:* The mathematical operation performs on each plaintext block in each round called round function.

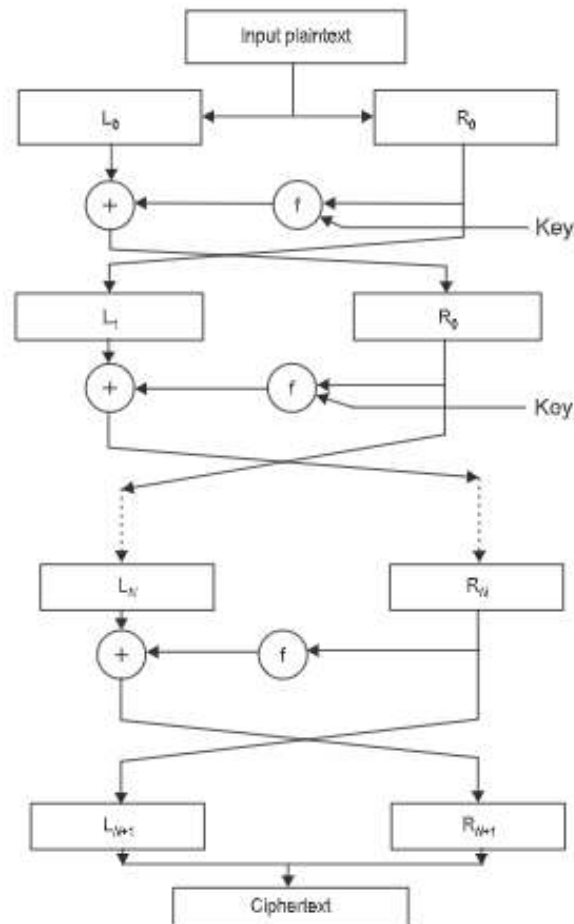


Figure 3.11 Feistel cipher.

3.5 DATA ENCRYPTION STANDARD

Data encryption standard (DES) is a block cipher. The design of DES depends on Feistel-structure. DES was published in 1977. DES uses 64 bits plaintext block and 64 bits key (actually DES uses 56 bits out of 64 bits). To add more security, it was modified time to time. Hardware version of DES is faster than software version. Using brute force attack, DES is broken in 1998. In 1998 a modified DES version was published known as Triple DES or 3DES. It is more secure than simple DES but it consumes more time for encryption. DES is broken and there was a need for new cipher. In 1999, a new block cipher known as advanced encryption standard (AES) was superseded the DES cipher. In this chapter we will discuss the detail working of DES, 3DES and the cryptanalysis of DES algorithm. Chapter 4 covered the detail working of AES.

Design of DES is based on Feistel structure. Various permutations and S-boxes (substitution boxes) are used to provide confusion and diffusion. Diffusion means the wide range of ciphertext is generated from plaintext. The purpose of diffusion is to make the relationship between the plaintext and ciphertext as complex as possible. That means there is no linear mathematical relation between the plaintext and the ciphertext. Each bit of plaintext affects the value of many bits of ciphertext. Confusion means the relationship between the key and the ciphertext as complex as possible. Confusion and diffusion are the attributes for defining the strong encryption algorithm.

The message is divided into a plaintext block of 64 bits. Each plaintext block undergoes various permutations and substitution operations. There are total 16 rounds in DES. The plaintext block used is of size 64 bits. The key size for DES is 64 bits of which 56 bits are actually used as a key. From this key 16 subkeys are generated, one subkey for each round. Same encryption algorithm is used as decryption except the subkeys are used in reverse order. The detail working of DES is discussed below.

3.5.1 Working of DES

DES algorithm is designed for encryption and decryption of blocks of plaintext. The size of plaintext block is of 64 bits which is encrypted into a ciphertext block of 64 bits. Each bit may be either 1 or 0. Initially a block of plaintext undergoes permutation called initial permutation. This allows 2^{64} possible arrangements of the 64 bits in a plaintext block. The architecture of DES is shown in Figure 3.12.

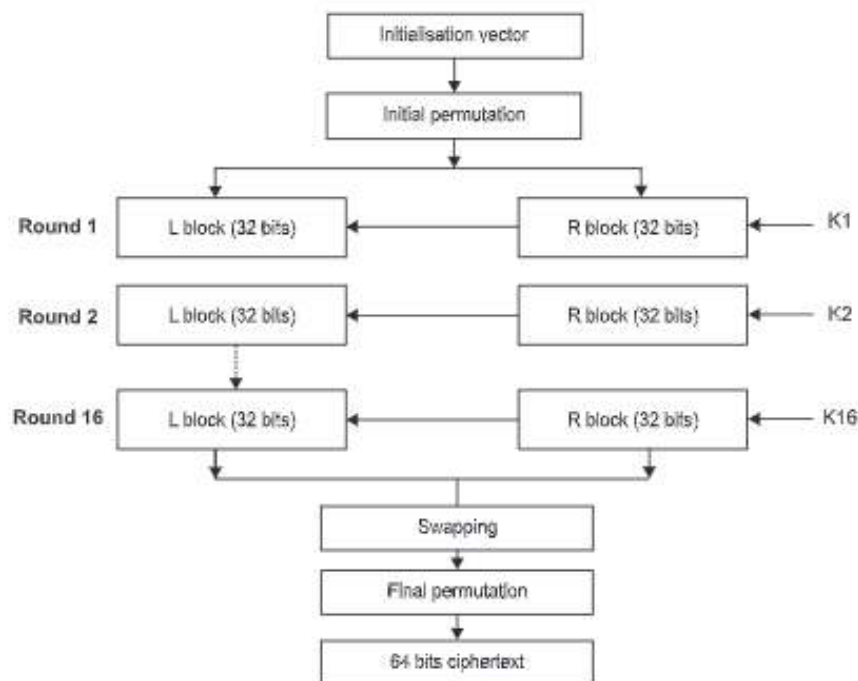


Figure 3.12 Steps in DES.

The working of DES algorithm is divided into following four parts.

1. Subkey generation
2. Initial permutation
3. 1 to 16 rounds
4. Final permutation

Now we will discuss each key generation first and then each step for first round in detail:

Subkey Generation

DES uses the key of size of 64 bits. The first step is to apply initial permutation on 64-bit key using initial permutation (64 bits) table. Then remove the parity bits from the 64-bit key and reduced to 56-bit key. We can do this by not selecting the parity bits (i.e., bits position 8, 16, 24, 32, 40, 48, 56 and 64). Apply expansion permutation on this 56-bit key using expansion permutation (56 bits) table. Next, divide this 56 bit key into two halves, viz. left and right. Each half has 28 bits. Each half undergoes circular left shift. The shift of bits by 1 or 2 positions is round dependent as shown in Table 3.1.

Table 3.1

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits shifted left	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

To generate 16 subkeys, the above procedure is carried out on each subkey. After shifting each half, concatenate two halves to get 56 bits key. Then apply the compression permutation (48 bits) on 56 bit keys. The result is 48-bit subkey. Figure 3.13 shows the subkey generation in details.

The 64-bit key is permuted using following permutation tables (Table 3.2(a-c)).

Table 3.2 Permutation tables

(a) Initial permutation (64 bits)	(b) Expansion permutation (56 bits)	(c) Compression permutation (56 bits)
[58 50 42 34 26 18 10 2]	[57 49 41 33 25 17 9]	[14 17 11 24 1 5]
[60 52 44 36 28 20 12 4]	[1 58 50 42 34 26 18]	[3 28 15 6 21 10]
[62 54 46 38 30 22 14 6]	[10 2 59 51 43 35 27]	[23 19 12 4 26 8]
[64 56 48 40 32 24 16 8]	[19 11 3 60 52 44 36]	[16 7 27 20 13 2]
[57 49 41 33 25 17 9 1]	[63 55 47 39 31 23 15]	[41 52 31 37 47 55]
[59 51 43 35 27 19 11 3]	[7 62 54 46 38 30 22]	[30 40 51 45 33 48]
[61 53 45 37 29 21 13 5]	[14 6 61 53 45 37 29]	[44 49 39 56 34 53]
[63 55 47 39 31 23 15 7]	[21 13 5 28 20 12 4]	[46 42 50 36 29 32]

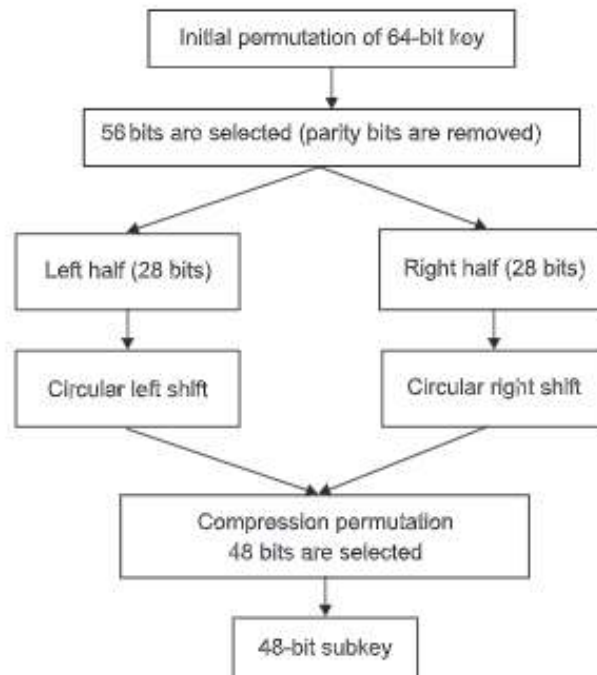


Figure 3.13 Subkey generation for DES.

Subkey generation steps are summarised as below:

1. Apply initial permutation on 64-bit key using expansion table.
2. Remove the parity bits from 64 bits to reduce the key to 56 bits.
3. Apply permutation on the 56-bit key.
4. Split the 56-bit key into two halves 28 bits each.
5. Apply circular left shift on each half.
6. Concatenate the two halves.
7. Apply compression permutation to reduce the key to 48 bits.

Initial Permutation

The message is converted into binary form. Then each bit allocates its position starting from right (0^{th} position) to left (64^{th} position). Initial permutation is applied on 64-bit block of plaintext using Table 3.3. The first bit is the 58^{th} bit from the plaintext block; second bit is the 50^{th} bit and so on. This table performs the permutations of the plaintext bits.

Table 3.3 Initial permutation

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Then this 64-bit plaintext block undergoes different operation from round 1 to 10.

Rounds 1 to 16

There are 16 rounds in DES. All the 16 rounds have similar operations of the plaintext blocks. Single round of DES is explained diagrammatically in Figure 3.14. The 64 bits plaintext block is split into two parts of 32 bits each called right and left parts. Each round of DES has following steps:

- (a) Expansion permutation
- (b) XOR operation
- (c) S-box substitution
- (d) P-box permutation
- (e) XOR operation
- (f) Swapping

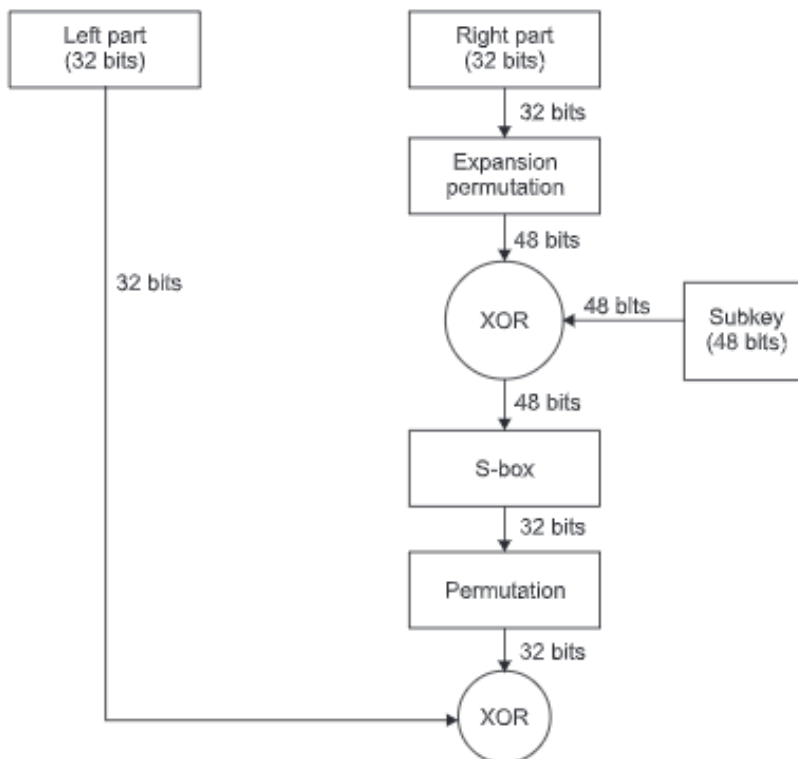


Figure 3.14 Details of single round of DES.

Expansion permutation: The right part having 32 bits undergoes expansion permutation using Table 3.4. This step expands the right half part to 48 bits. Therefore, some of the bits are repeated. The new bit position is as per the table, i.e., first bit is the 32nd bit, and second bit is the 1st bit and so on.

Table 3.4 Expansion permutation

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

XOR operation: In this step XOR operation is performed between the 48 bits of right half and the subkey. This creates key dependency.

S-box substitution: The 48 bits of right half are divided into 8 groups. Each group has 6 bits. There are eight S-boxes, one for each group. The selection of S-boxes is very important as proper selection of S-boxes make the cryptanalysis of DES difficult. The S-box structure is shown in Table 3.5.

Table 3.5 S-box structure

R\C	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	9	15	8	3	6	10	14	0	4	13	7	1	11	2	5
1	3	7	1	6	11	4	0	15	2	8	5	12	10	14	13	9
2	15	4	5	14	12	0	3	11	7	1	10	6	13	9	2	8
3	8	11	9	13	0	5	10	7	12	6	3	2	15	14	4	1

The input to the S-box is a group of 6 bits and the output of S-box is 4 bits. There are total eight S-boxes in each round which convert 48 bits (6 bits per group \times 8 groups) input to 32 bits (4 bits per group \times 8 groups) output. S-box can be used as follow:

1. 48 bit plaintext is split into eight groups of 6 bits each.
2. Then select MSB and LSB, i.e., the first and the last bit of a group. These two bits form a 2-bit binary number. The decimal equivalent of this number is a row number from the S-box. For example, suppose a group is 100000, then MSB is 1 and LSB is 0. The pair is 10 (binary). The decimal equivalent of $(10)_2$ is $(02)_{10}$. So row 2 is selected.
3. Then select the middle four bits of the same group. These four bits form a 4-bit binary number. The equivalent decimal number is a column number from the S-box. For example, suppose a group is 100000, then middle four bits are

0000. Then the binary number is 0000 (binary). The decimal equivalent of $(0000)_2$ is $(00)_{10}$. So column 0 is selected.
4. The intersection of the selected row and column is the output in decimal. For the above example, row 2 and column 0 is selected. The intersection from S-box is $(15)_{10}$.
 5. The binary equivalent of this decimal number is the 4-bit output from the S-box. In this example, the output is $(1111)_2$.

Using the above procedure, for all 8 groups and 8 different S-boxes, 8 groups of 4 bits each is generated. S-boxes are used to provide the confusion to the algorithm. Figure 3.15 shows the use of S-boxes.

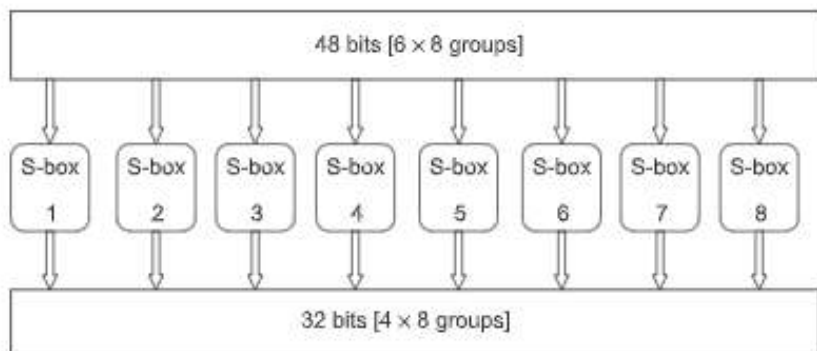


Figure 3.15 S-boxes working.

P-box permutation: The output of S-boxes (32 bits) is sent as input for permutation to P-box. Here every bit is used once. The output of P-box is of 32 bits. Table 3.6 is used for permutation. Permutation is used to provide the diffusion to the algorithm.

Table 3.6 Permutation table

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

XOR operation: The XOR operation is carried out between the output of the P-box and the left half part. The result is 32-bit output.

Swapping: The result of XOR is used as a new right half for the next round and the old right half part is used as a new left half for the next round.

Final Permutation

This procedure continues for total 16 rounds with each round using a different subkey. Then the invert permutation is applied on the output of last round. This gives 64-bit ciphertext block of 64 bits for a plaintext block of 64 bits.

Same process is repeated for all plaintext which gives the ciphertext for a given message. DES uses same encryption algorithm for decryption. But the order of subkeys is reversed, i.e., first round uses 16th subkey, second round uses 15th subkey, in the same way the last round uses 1st subkey.

3.5.2 Cracking DES

Diffie and Hellma proved that DES can break by brute force attack. As the key size for DES is 56 bits, maximum number of possible keys using 56 bits is 2^{56} . To break DES by brute force attack, maximum number of keys required is 2^{56} . So, they raise some objections about the use of data encryption standard. Next attack on DES was made by John Gilmore and his team. They proved that DES can break within 4.5 days. Using high performance, they announced that DES had broken in 22 hours.

3.6 TRIPLE DES

From above attempts, it was known that DES is not secure. So, the alternative algorithm or improvement in DES was expected. From time to time various versions of DES were proposed. One of the versions of DES is double DES. The working of double DES is same as that of simple DES. Only difference is the data encryption is done two times. Two keys are required for double DES. It is more secure than simple DES because key size is 112 bits (two keys of 56 bits each). To break double DES by brute force attack, maximum number of keys required is 2^{112} . Therefore, it makes brute force attack more difficult as compared to simple DES. But double DES suffers by meeting in the middle attack. The next version of DES is triple DES also known as 3 DES. Triple DES uses three keys and encrypted the data three times. So triple DES takes more time as compared to DES algorithm. As three keys are used, brute force attack is difficult against triple DES. It is more secure algorithm than DES algorithm.

3.6.1 Working of Triple DES

The working of triple DES is same as DES. But the complete encryption process is repeated three times. Three keys of 56 bits each are used for triple DES. Therefore, the overall key length is 168 bits. In triple DES, the data is encrypted thrice using these three keys. The first step is encrypting the data using first key. The second step is to decrypt the ciphertext (output of the first step) using second key. The output of the second step is encrypted using third key (Figure 3.16). So, DES algorithm has to run three times. Mathematically, triple DES encryption can be represented as:

$$C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$

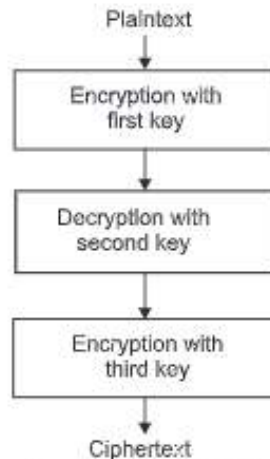


Figure 3.16 Triple DES.

The first step is simple execution of DES algorithm. The second step is decryption in which DES is executed in reverse order of encryption. If any two keys such as first and second or second and third keys are same then the encryption in triple DES is same as simple DES. So for more security three different keys should be selected. There are no current known practical attacks of triple DES.

3.6.2 Modes of Operation

Triple Electronic Code Book (3ECB)

The triple electronic code book works like the ECB mode of simple DES.

Triple CBC (Cipher Block Chaining)

Cipher block chaining (CBC) mode of operation for triple DES is similar to that of simple DES. The first 64-bit key used as the initialisation vector. Triple CBC is then executed for a single 64-bit block of plaintext. Then XOR operation is performed between the ciphertext and the next block of plaintext. CBC provides more security to triple DES.

3.7 DES DESIGN CRITERIA

3.7.1 Design of S-box

For more secure DES, S-box plays the major role. We know that S-box provides confusion property to the DES. Large S-box provides more resistance to cryptanalysis. In DES, other than S-box, other operations are linear which are helpful for cryptanalysis of the

algorithm. So, one should take more care for the design of S-box. Large S-box provides more confusion but it is difficult to design larger S-box. There for more time and effort have to spend for the design of S-box. Strong S-box is designed using the non-linear Boolean function. The NIST set the design criteria for S-1. If we select the values of S-box randomly, then the S-box may be weak and cryptanalysis of such S-box is easy. So, the values in the S-box are selected using the non-linear Boolean function, such as Bent functions, which provide strong S-boxes. Due to a change in one bit of input, each bit in the output will change. This is called *Avalanche effect*. The S-box should achieve Avalanche effect.

Design criteria for a good S-box are:

- For the design of $n \times n$ S-box, mapping from input to output should be one-to-one and onto.
- Use a function such that for a change in one bit in the input, at least 50% of the bits in the output should change. This helps to achieve Avalanche effect.
- Output bits act independently from each other.
- There should not be a linear relationship between the input and the output of the S-box.
- The number of 0's and 1's in the S-box should be equal.

3.8 OTHER BLOCK CIPHERS

Apart from DES and triple DES, there are other block ciphers which have been developed from time to time. Some of the known block ciphers are advanced encryption standard (AES), CAST-128, IDEA, and Blowfish. AES is a block cipher having a variable number of rounds (10, 12 or 14 rounds), 128-bit block of plaintext and 128-bit/192-bit/256-bit key for encryption. CAST is a block cipher having 16 rounds, 64-bit plaintext block, and 128-bit key. Another block cipher is the international data encryption algorithm (IDEA). Another block cipher is Blow fish. It is unpatented algorithm. Its architecture is based on Feistel structure. Like AES, it also uses a key of variable length. We learn in detail about these algorithms in the next chapters.

3.9 DIFFERENTIAL CRYPTANALYSIS

Biham and Shamir introduced a cryptanalysis technique known as *differential cryptanalysis*. This technique helps for the study and attack of the cryptographic algorithms. This cryptanalysis technique is a chosen-plaintext or chosen-ciphertext attack. In this technique, the difference in plaintext blocks is compared with the difference in ciphertext blocks to know the key. To get the correct key, large number of plaintext blocks and ciphertext blocks comparison should be done. Cryptanalyst first chooses pairs of plaintext with some particular difference among them. The corresponding ciphertext with these pairs is taken. Then study the difference between the ciphertext pairs and plaintext pairs. Using this information, cryptanalyst tries to find out the key. This technique is generally used for the cryptanalysis of the block

ciphers. It is also used for the cryptanalysis of stream ciphers. The basic architecture of DES is based on substitution and diffusion.

3.10 LINEAR CRYPTANALYSIS

Another form of cryptanalysis technique is linear cryptanalysis which is based on linear approximations. This cryptanalysis technique can be used against both the stream and block ciphers. The loopholes in the cipher can be found out using linear cryptanalysis. This helps to improve the performance of the cipher. In this technique, both plaintext and ciphertext are used for cryptanalysis. The key is found out by using the plaintext through a simplified cipher in the complete ciphertext. XOR operation is used to get the key. Some bits of the plaintext are XOR, also some bits of the ciphertext are XOR. The result is XOR together. This result is same as the XOR of some bits of the key. This helps to get the complete key.

3.11 WEAK KEYS IN DES ALGORITHMS

The performance of any encryption algorithm is based on the keys used. But all the keys may not be strong. Some ciphertexts are relatively easy for cryptanalysis. The keys used for generation of such ciphertexts are called *weak keys*. The keys having high degree of similarity are called *simple weak key*.

For example, if any key is composed of:

- all bits are zeros,
- all bits are ones,
- alternating bits are ones and zeroes, and
- alternating bits are zeroes and ones, weak keys have one of the above combinations. For DES, there are total 2^{66} keys of which sixteen keys are considered as weak keys. With the above combinations, following 16 weak keys are produced:

Table 3.7 Weak keys

	L_0	L_0	L_0	L_0
R_0	0,0..	1,0..	0,1..	1,1..
R_0	0,0..	0,0..	0,0..	0,0..
R_0	1,0..	1,0..	1,0..	1,0..
R_0	0,1..	0,1..	0,1..	0,1..
R_0	1,1..	1,1..	1,1..	1,1..

From Table 3.7, two keys which have all the bits of L_0 and R_0 as zeroes or ones. These keys are weak because they have their own inverses. Permutation and shifting does not change the key. Therefore, subkeys of these two keys are the same keys. Therefore, all the rounds have the same key. Other than these two subkeys, there are two other keys having each half all ones or zeroes. That means left 28 bits are zeroes and right 28 bits are ones and vice-versa. These four keys are very weak keys and recommended for not to use. Other twelve keys are the combinations of zeroes and

ones, such as alternate bits in the key are ones and zeroes as per the table. These twelve keys are called *semi-weak keys*. For good encryption it is recommended not to select such keys.

EXAMPLE 3.1 Let the message be $M = \text{COMPITDT}$ and the key be $K = \text{COEPPUNE}$. USE DES algorithm to encrypt and decrypt the message.

Convert M to ASCII and rewrite it in binary format, we get the 64-bit block of plaintext:

```
M = 01100011 01101111 01101101 01110000 01101001 01110100 01100100 01110100
L = 01100011 01101111 01101101 01110000
R = 01101001 01110100 01100100 01110100
```

We first write the message in 8×8 matrix form as below:

```
01100011
01101111
01101101
01110000
01101001
01110100
01100100
01110100
```

The first bit of M is "0". The last bit is "0". We read from left to right.

Convert K to ASCII and rewrite it in binary format, we get the 64-bit key as:

```
K = 01100011 01101111 01100101 01110000 01110000 01110101 01101110 01100101
Write the key in  $8 \times 8$  matrix form as below:
```

```
01100011
01101111
01100101
01110000
01110000
01110101
01101110
01100101
```

Solution The DES algorithm uses the following steps.

Step 1 Generate 16 subkeys (48-bit length)

```
Round=1 key=111000001011111011101110110100000100000010011110
Round=2 key=111000001011011011110110100100011010110110000100
Round=3 key=111101001101111001110110001010000010011010010001
Round=4 key=111001101111001101110010011110110110000000000111
Round=5 key=101011101101011101110111001001100100000110001010
Round=6 key=111011110101001101011011100001000011000101000111
Round=7 key=001011111101001111111001111001101000001011100000
Round=8 key=100111110101100111011011010100001000111101001011
```

```

Round=9  key=000111110100101111011011011001001001010100001100
Round=10 key=001111110111100110011101100010000001010011101110
Round=11 key=000111110010110111001101010011001101101010100001
Round=12 key=010110110110110010111101000100100100110001111001
Round=13 key=110111011010110110101100100010111001100100010000
Round=14 key=110100101010111010101111100000010110011100110000
Round=15 key=111110011011111000100110011110010000101000000100
Round=16 key=111100011011111000101110000100101000001001110100

```

Plaintext after rounds

```

10100010
00001111
11100011
01010000
01011100
11101111
01010011
00001110

```

Printing Ciphertext in int form

```
60 126 178 178 137 100 173 100
```

Ciphertext generated:

```
< ~ ^ ^ %o d d
```

-----DECRPTION-----

After initial permutation

```

1 0 1 0 0 0 1 0
0 0 0 0 1 1 1 1
1 1 1 0 0 0 1 1
0 1 0 1 0 0 0 0
0 1 0 1 1 1 0 0
1 1 1 0 1 1 1 1
0 1 0 1 0 0 1 1
0 0 0 0 1 1 1 0

```

Plaintext matrix after Decryption

```

0 1 1 0 0 0 1 1
0 1 1 0 1 1 1 1
0 1 1 0 1 1 0 1
0 1 1 1 0 0 0 0
0 1 1 0 1 0 0 1
0 1 1 1 0 1 0 0
0 1 1 0 0 1 0 0
0 1 1 1 0 1 0 0

```

After Decryption
compitdt

CHAPTER 5

Symmetric Ciphers

5.1 INTRODUCTION

In the last two chapters, we learn two symmetric encryption ciphers: DES and AES. There are many other symmetric encryption ciphers. In this chapter, we are going to discuss symmetric encryption ciphers such as Blowfish, RC4, RC5, RC6 and IDEA. RC4 is a stream cipher whereas Blowfish, RC5, RC6 and IDEA are block ciphers.

5.2 BLOWFISH ENCRYPTION ALGORITHM

Another encryption algorithm is Blowfish algorithm. It is a symmetric encryption block cipher. It is based on Feistel structure. Bruce Schneier designed it in 1993 as an alternative to DES and IDEA algorithm. Blowfish uses the plaintext block of 64 bits length and variable key length from 32 bits to 448 bits. It is unpatented algorithm.

Characteristics of Blowfish algorithm:

- A symmetric block cipher with 64-bit plaintext block.
- Key length is variable and it is from 32 bits to 448 bits.
- Stronger and faster than contemporary ciphers such as DES and IDEA.
- Less memory is required and easy for implementation.
- It is unpatented and so freely available.

The working of Blowfish algorithm is divided into following two parts:

1. Key expansion
2. Encryption

The important feature of Blowfish is its key scheduling. The eighteen subkeys are generated from the original key. Even shortkeys can be used for Blowfish.

The cryptanalysis of the keys for this algorithm is difficult. The beauty of Blowfish algorithm architecture is the use of round keys, the S-boxes, and number of iterations. This makes Blowfish stronger and faster than contemporary ciphers such as DES and IDEA.

5.2.1 Key Expansion

Blowfish needs eighteen subkeys for encryption. Same subkeys are used for decryption. These keys are generated by processing the original key having maximum length 448 bits. During encryption, the original key is converted into eighteen subkeys having total 4168 bytes. There are two arrays: key-array and S-boxes as shown in Figure 5.1. The key-array consists of eighteen subkeys of length 32 bits each. The subkeys are K_1, K_2, \dots, K_{18} . All the boxes are initialised with a fixed string.

1. The key-array consists of 18 subkeys of 32 bits each such as: K_1, K_2, \dots, K_{18} .
2. There are four S-boxes such as S_1, S_2, S_3 and S_4 . The size of each S-box is 32 bits. In each S-box there are total 256 entries as from 0 to 255.

$$\begin{aligned} S_{1,0}, S_{1,1}, \dots, S_{1,255} \\ S_{2,0}, S_{2,1}, \dots, S_{2,255} \\ S_{3,0}, S_{3,1}, \dots, S_{3,255} \\ S_{4,0}, S_{4,1}, \dots, S_{4,255} \end{aligned}$$

The key generation process for the Blowfish algorithm is as follows:

- Step 1** Initialise the key-array first and then the four S-boxes, with the fixed hexadecimal digits.

$$\begin{aligned} K_1 &= 0 \times 243f6a88 \\ K_2 &= 0 \times 85a308d3 \\ K_3 &= 0 \times 13198a2e \\ K_4 &= 0 \times 03707344 \end{aligned}$$

- Step 2** The first 32 bits of the original key are XOR with the first 32 bits of the key-array, i.e., K_1 . The second 32 bits of the original key are XOR with the second 32 bits of the key-array, i.e., K_2 . Repeat this for all the entries of the key-array is XOR with the original key. There is at least one equivalent longkey for every shortkey. For example, suppose the shortkey is X, there may be equivalent longkeys as XX, XXX, XXXX, etc.

- Step 3** Use the above subkeys to encrypt (using Blowfish algorithm) all-zero string.
- Step 4** Replace K_1 and K_2 with the output of Step 3.
- Step 5** Use these modified subkeys to encrypt (using Blowfish algorithm) the output of Step 3.
- Step 6** Replace K_3 and K_4 with the output of Step 5.
- Step 7** Continue the above process. Replace all entries of the K-array, and then all S-boxes with the changing output of the algorithm.

Total 521 iterations are required for the generation of eighteen subkeys. Generation of subkeys needs large memory. The same subkey can be used for next time so it is important to store these subkeys. This required memory for storage. Due to this requirement, Blowfish is not suitable for applications where frequently changes in keys are required.

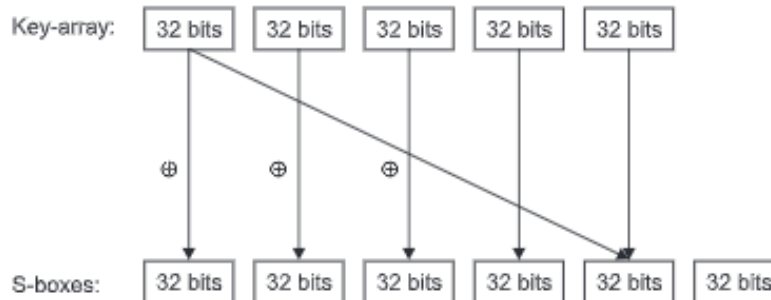


Figure 5.1 Subkeys generation

5.2.2 Encryption

The Blowfish algorithm has sixteen rounds. First divide the plaintext into a block of 64 bits. Then each block of plaintext is divided into two halves: left part and right part. The left part undergoes the “f” function. This is opposite to DES, where the “f” function is applied to the right half of the block. The result is XOR with the right half of the block. The working of each round has following steps:

- Step 1** Left half of the plaintext block is XOR with the subkey for that particular round.
- Step 2** Apply the f-function to the output of first step.
- Step 3** The result of second step XOR with the right half of the plaintext block.
- Step 4** The result of Step 3 is shifted as left half for the next step and the output of the first step is shifted as right half for the next step.

Repeat above steps for all sixteen rounds. At the end of sixteenth round the 17th subkey is XOR with the right half and the 18th subkey is XOR with the left half. Then interchange the right and left halves. Concatenate the left and right half to get the ciphertext. Each round uses one subkey for XOR operation with the left half. The “f” function uses S-boxes which are dependent on the key. The use of addition and bitwise XOR operations make the cryptanalysis of Blowfish more difficult.

The architecture of Blowfish for encryption is shown in Figure 5.2.

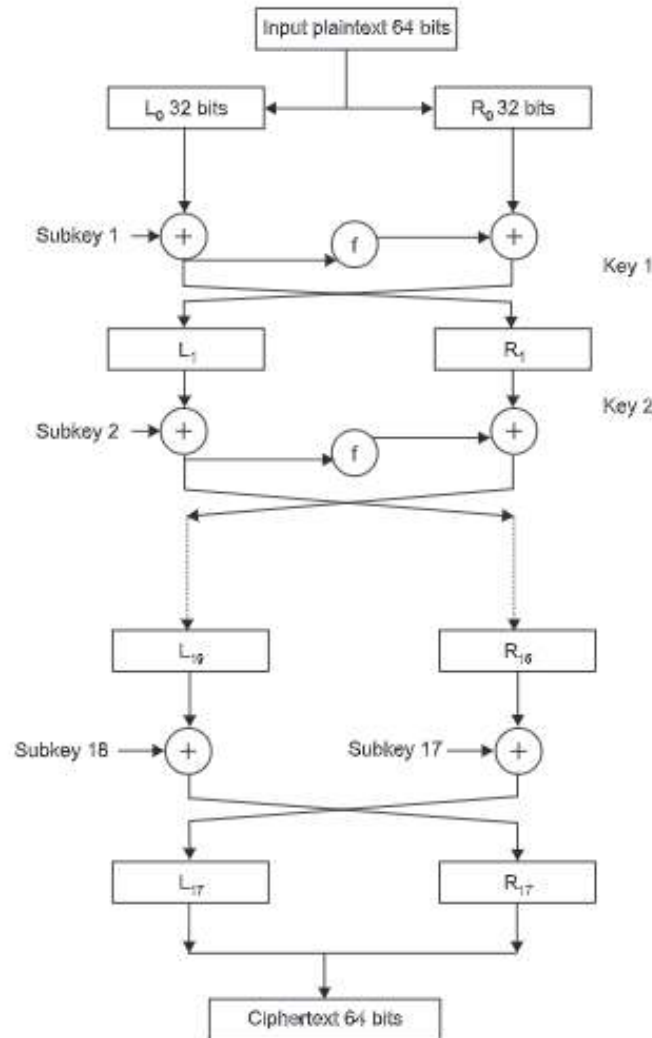


Figure 5.2 Architecture of Blowfish algorithm for encryption.

Decryption follows the same procedure as encryption. The subkeys are used in the reverse order of encryption. First round uses 18th subkey and the first and second subkeys are used after the last round. The function "f" and S-boxes are same as encryption. The architecture of Blowfish for decryption process is shown in Figure 5.3.

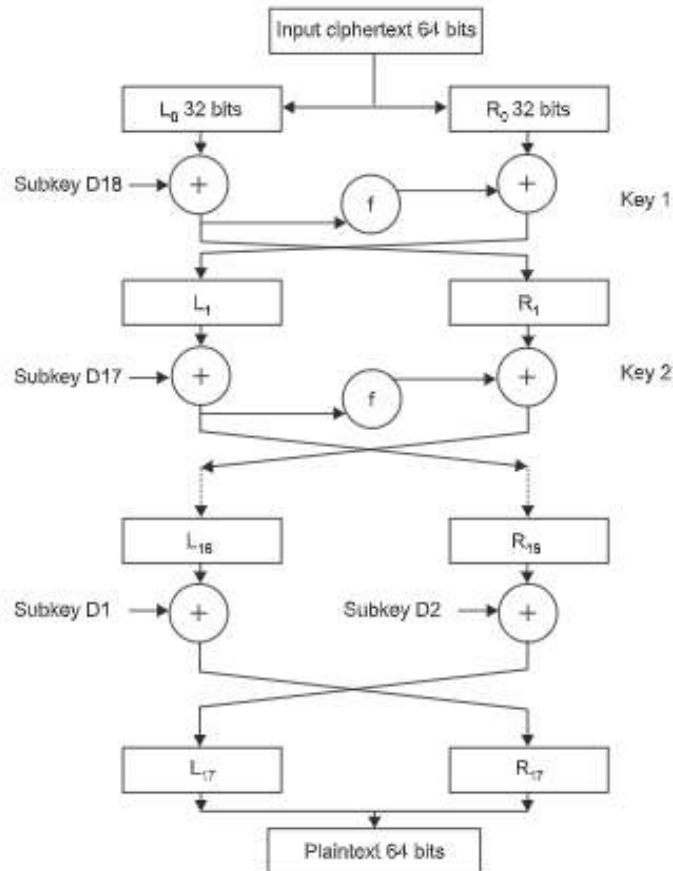


Figure 5.3 Architecture of Blowfish for decryption.

5.2.3 Blowfish Architecture

The architecture of Blowfish algorithm depends on the Feistel structure. Like DES, Blowfish also have 16 rounds with 18 subkeys. There are one subkey for each round and two subkeys are used after the last round. Here the subkeys are used to perform XOR with the left part of the plaintext in each round whereas the output of the last round, each part is XOR separately with the last two subkeys respectively. Blowfish uses four S-boxes. The S-boxes are used to convert the 8-bit input into 32-bit output. The output of the last round is XOR in such a way that left part is XOR with the 17th subkey and right part is XOR with the 18th subkey. The output of these XOR is concatenating to get the ciphertext.

Figure 5.4 shows the architecture of Blowfish's f -function. The left 32-bit plaintext block after XOR with key is the input for S-boxes. This 32-bit input is split into four

groups. Each group have 8 bits each. There are four S-boxes, one for each group. For each S-box, the output is 32 bit. Then addition modulo 2^{32} is carried out for all the four outputs of the S-boxes and then XOR operation is performed which produce the 32-bit output.

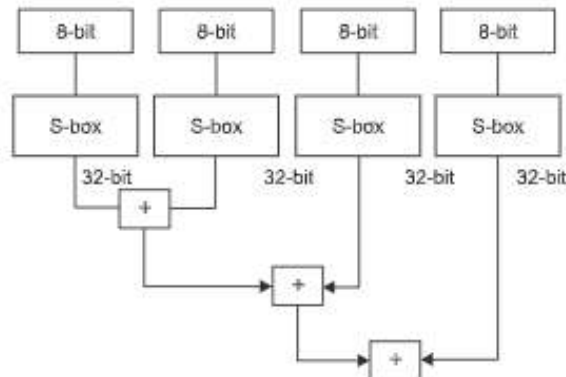


Figure 5.4 Blowfish's f -function.

5.2.4 Cryptanalysis of Blowfish

Blowfish algorithm is more secure against the cryptanalysis. There is no known cryptanalysis attack against Blowfish algorithm. Vaudenay found a known plaintext attack in 1996 against Blowfish. This requires 2^{8n+1} known plaintexts to break this algorithm, where n is the number of rounds. But for $n = 16$ rounds, this attack is not successful. He also claims about some weakkeys for Blowfish. Blowfish is suitable for applications having small plaintext but for large plaintext it may not be suitable algorithm.

5.3 RC5

One more block cipher algorithm is RC5. It is a symmetric encryption algorithm. It is designed by Prof. Ronald Rivest of MIT in 1994. RC5 architecture is based on the Feistel structure. It is a parameterised algorithm such as $w:r:b$ parameters. Where w is word size in bits, r is number of rounds and b is number of bytes in the secret key K . It is a very fast algorithm. RC5 uses rotation of bits which is data dependent and mixture of different operations. This makes linear and differential cryptanalysis of RC5 more difficult.

RC5 has the variable number of rounds. Also it uses the plaintext block and the key of variable length. The number of rounds used by RC5 may be from 1 to 255. The plaintext block may be of size: 32 bits, 64 bits or 128 bits. RC5 uses the key of size from 0 to 2048 bits. If the plaintext block is of size of 64 bits then the key of size of 28 bits is recommended. The variability in number of rounds, block size and the key length provide flexibility in the working of RC5 algorithm. This helps to provide suitable security to RC5. User can select the number of rounds, size of plaintext block and the key length as per his/her requirements depending on the applications.

RC5 encryption is shown in Figure 5.5.

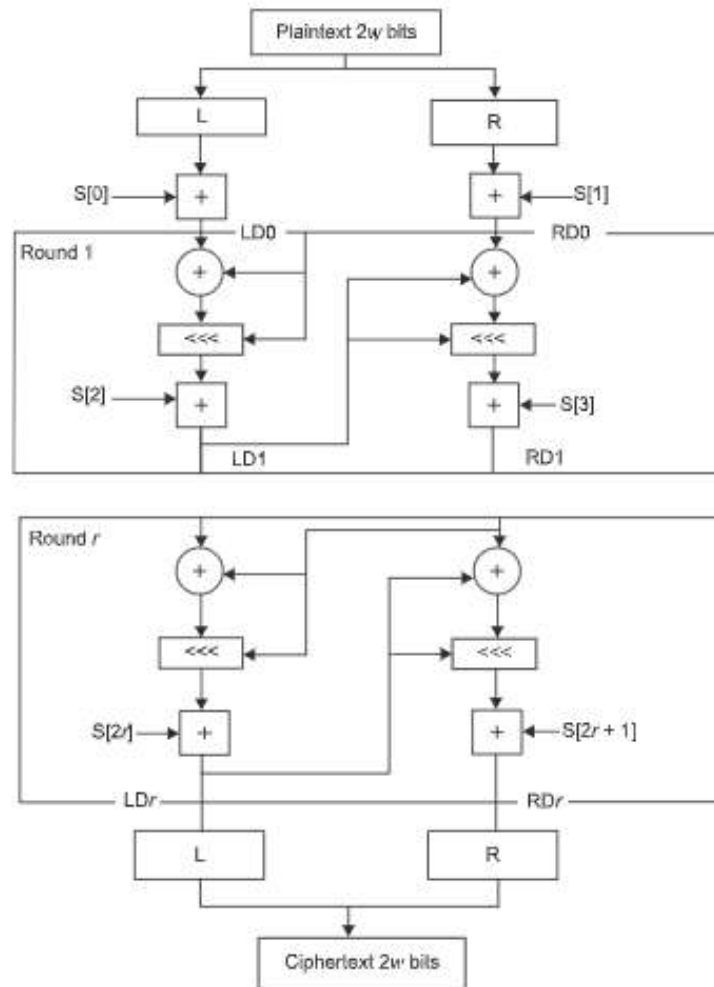


Figure 5.5 RC5 encryption.

5.3.1 Characteristics of RC5

- Uses number of parameters and variables
- Variable number of rounds (from 1 to 255)
- Variable plaintext block size (32 bits, 64 bits or 128 bits)
- Variable key length (0 to 2048 bits)
- Rotation of bits is data dependent
- Uses mixture of different operations like, 2's complement, addition words modulo 2^w , bitwise XOR of words
- Fast and simple software and hardware implementation.

5.3.2 Parameters

RC5 has number of parameters and variables as below:

RC5 uses three parameters such as w , r and b .

- ' w ': word size in bits, RC5 encrypts plaintext blocks of 2 words at a time. Allowable values are: 16, 32, 64 bits.
- ' r ': number of rounds. Allowable values are: 0, 1, 2, - - - , 255.
- ' b ': number of bytes in the secret K (1 byte = 8 bits). Allowable values are: 0, 1, 2, - - - , 255.
- The working of RC5 algorithm is divided into following three parts.

1. *Key expansion:* The key expansion algorithm consists of 4 steps. In this, original key is used to fill the expanded key table. The size of the key table depends on the number of rounds.

- Definition of the constant
- Converting the key from bytes to words
- Initialising the S-array
- Mixing the key

Two constants are used for key expansion step. These constants are of size 2-word binary number. For $w = 32$, the constants are: B7E1 (1011 0111 1110 0001) and 9E37 (1001 1110 0011 0111). The key is stored in an array $A[0,1, \dots, n-1]$. Where A-array consists of $n = (r-1)w$ -bit words. Copy this key from array A to another array $B[0, 1, 2, \dots, m-1]$ where $m = n/p$, where $p = w/8$ is the number of bytes per word. Unfilled positions of B are filled with zeroes. Then initialise the S-array such as

$S[0] = \text{first-constant}$.

$S[i] = S[i-1] + \text{second constant}$.

Then mix the original key in three passes over B-array and S-array. The size of B and S-arrays are different. So, the array having large size should process three times and other array process more than three times.

The above key expansion process is one way and it is difficult to find out the key from the S-array.

2. *Encryption:* The encryption part consists of three operations: add words modulo 2^w , bitwise XOR of words and left hand rotation. The encryption using RC5 is shown in Figure 5.4. The plaintext is divided into a block of two words of size w bits each. Suppose the two plaintext words are P and Q. The encryption algorithm can work as:

$P = P + S[0];$

$Q = Q + S[1];$

for $i = 1; i < r; i++$

{

$P = ((P \text{ XOR } Q) \lll Q) + S[2 * i]$

$Q = ((Q \text{ XOR } P) \lll P) + S[2 * i + 1]$

}

The result is ciphertext of length $2w$.

3. *Decryption*: It is the reverse of the encryption process. Last round of encryption is processed first. The last but one round is processed second and the first round processed last. It uses subtraction words modulo 2^w , bitwise XOR of words and right hand rotation. The encryption algorithm can work as:

```

for i = r; i > 1; i--
{
    Q = ((Q - S[ 2 * i + 1] >>> P) XOR P);
    P = ((P - S[ 2 * i ] >>> Q) XOR Q);
}
P = P - S[ 0];
Q = Q - S[ 1];

```

5.3.3 Cipher Modes in RC5

RC5 have four different cipher modes of operation:

1. *The raw block cipher mode*: Fixed size block of plaintext is converted into a fixed size block of ciphertext.
2. *Cipher block chaining (CBC)*: It helps RC5 to process the messages of variable length. Like DES, for RC5 it generates different ciphertext for the similar plaintext blocks.
3. *CBC pad mode*: Generally, the plaintext is not always multiples of block size. So, the last block has less than necessary number of bytes. So, extra bytes are appended to get the required number of bytes in the last block. This makes the processing of the last block possible. All the padding bytes should have same pattern. For example, if there are 2 bytes of padding, each byte has the pattern 1110 0010.
4. *CTS cipher mode*: It is the ciphertext stealing mode. This mode handles plaintext block of any size. The ciphertext blocks generated are of the same size as that of plaintext blocks.

The cryptanalysis of RC5 is difficult due to its flexible number of rounds, key length and plaintext block selection. RC5 can use rounds from 1 to 255 but RC5 with 12 rounds provide sufficient security against differential and linear cryptanalysis. RC5 with more number of rounds provides more security but this also increases the processing time. For maximum number of rounds, the performance of RC5 decreases. Brute force attack is difficult for 128 bits key length. This provides sufficient security to RC5. Increasing the key size provide more security but the key expansion takes more time as compared to small key. However, the total time required for encryption depends only on the number of rounds and not on the key size. The strength of RC5 is its data dependent rotation. Shift operation depends on mod w . If w is a power of 2, lower bits of $\log_2 w$ determine the number of shift positions. XOR operation in RC5 provides an Avalanche effect.

5.4 RC4

In 1987, a cipher designed by Prof. Ron Rivest is RC4 cipher. It is also known as Ron's code 4. It is publicly available in 1994. It is a stream cipher. RC4 uses keys of variable length. RC4 processing is based on byte-oriented operations. The cryptanalysis of RC4 is difficult. RC4 used in different applications such as secure socket layer (SSL), IEEE 802.11 wireless networking security standard, wireless WEP and email encryption products.

5.4.1 Design

RC4 is a binary additive stream cipher. It uses variable length key ranges from 8 to 2048 bits in multiples of 8. The key generation function in RC4 generates the keys which are XOR with the plaintext bits. The reuse of these keys makes the cryptanalysis of RC4 easier. This algorithm works in two parts: key generation and encryption.

5.4.2 Characteristics

- Variable key length (8 to 2048 bits)
- Use random permutation
- Encryption is faster than decryption
- Design is simple and effective

5.4.3 Algorithms

A random key is generated first. This key is XOR with the plaintext. Decryption uses the same process as encryption. To generate the key, the RC4 uses following steps:

Step 1 Uses 256 bytes array A, contains the permutation of the values from 0 to 255 byte.

Step 2 The RC4 cipher uses two algorithms: the key-scheduling algorithm, and the pseudo-random generating algorithm.

The key-scheduling algorithm is given as:

Initialisation

```
Input: p,  
       key length (in bytes) = n;  
for i = 0 to  $2^p - 1$   
  {  
    A[i] := i;  
  }  
j=0;
```

Scrambling of bits

```

For i=0 to  $2^p - 1$ 
  j=j+A[i]+B[i mod n]
  swap(A[i], A[j])

```

The pseudo-random generating algorithm is given as:

```

Initialisation
i = 0;
j = 0;

Generation loop
Loop
  i:=i+1
  j:=j+A[i]
  swap(A[i], A[j])
  output A[A[i]+A[j]]
end Loop

```

RC4 has a weak key problem. Out of every 256 keys, one key is a weak key. Cryptanalyst can find out this key and able to find one or more generator bytes.

5.5 RC6

RC6 is a symmetric block cipher designed by Ronald Rivest, Matt Robshaw, Ray Sidney, and Yiqun Lisa Yin at RSA Laboratories. It is one of the algorithms selected by others for the final round to become the new federal advanced encryption standard (AES). It is an improvement over RC5. Like RC5, it is a parameterised algorithm. RC6 consists of three parts: a key expansion, an encryption, and a decryption. RC6 uses the variable size plaintext block, variable length key and variable number of rounds. RC6 uses four w -bit registers, integer multiplications, quadratic equations and shifting by fixed bits.

RC6 uses 44 subkeys, S_0 to S_{43} . Each subkey is of 32 bits sizes RC6 uses 128-bit plaintext block. For each round of RC6, two subkeys are required. It uses four registers: A, B, C and D each of 32 bits to store the plaintext. Least significant byte of register A is used to store the first byte of plaintext or ciphertext. Most significant byte of register D is used to store the last byte of plaintext or ciphertext. Two subkeys are used at the initial step before the start of the first round. RC6 begins with an initial step: register B is XOR with subkey S_0 , and register D is XOR with subkey S_1 . Each round of RC6 uses two subkeys. The first round uses subkeys S_2 and S_3 , and next rounds use consecutive subkeys. The block diagram of RC6 is shown in Figure 5.6.

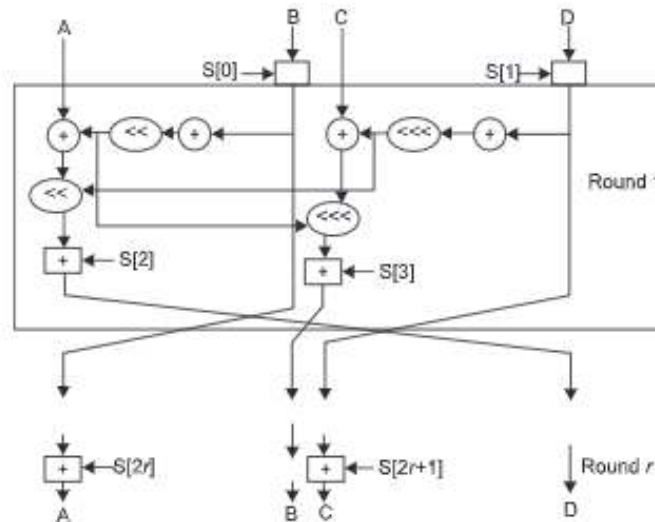


Figure 5.6 RC6 encryption.

5.5.1 Parameters of RC6

RC6- $w/r/b$ parameters:

- Word size: w (32 bits)
- Number of rounds: r (20)
- Size of key: b (16, 24, or 32 bytes)

Key expansion:

- Key-array $S[0 \dots 2r + 3]$ of w -bit round keys.
- Encryption and decryption: 4 input/output registers

5.5.2 Basic Operations

RC6 operates on units of four w -bit words using the following six basic operations.

- Integer addition modulo 2^w : $(a + b)$
- Integer subtraction modulo 2^w : $(a - b)$
- Bitwise XOR of w -bit words $(a \oplus b)$
- Integer multiplication modulo 2^w $(a \times b)$
- Shift left the w -bit word a by the amount given by the LSB (least significant bit) $\log_2(w)$ bits of b ($a \lll b$)
- Shift right the w -bit word a by the amount given by LSB $\log_2 w$ bits of b ($a \ggg b$)

5.5.3 Working of RC6

The encryption of RC6 consists of following three stages:

- pre-whitening stage
- an inner loop of rounds, and
- post-whitening stage

First and last stage help to remove the possibility of the plaintext informative part of the input to the first round of encryption and the ciphertext informative part of the input to the last round of encryption. The encryption process in RC6 can work as:

- The registers B and D undergo pre-whitening.
- The registers B and D are put through the quadratic equation which is defined as $f(x) = x(2x + 1)$ and applied circular left shift by $\log_2 w$ bits.
- The XOR is performed between the value of register B and the value of register A. Also XOR is performed between the value of register D and the value of register C.
- Apply circular left shift on the value of t by u bits. The result is added to round key $S[2i]$.
- Apply circular left shift on the resulting value of register D and C by t bits and added to round key $S[2i + 1]$.
- Then permutation is applied on A, B, C, D registers. This is done by using parallel assignment to mix the AB computation with the CD computation. This makes difficulty for cryptanalysis (Rivest et al., 1998a).
- Registers A and C undergo post-whitening.

In decryption, the first step, i.e., pre-whitening step is started from registers C and A instead of B and D. The loop runs in reverse for the number of r rounds.

The decryption process in RC6 can work as:

- Registers C and A undergo pre-whitening.
- The registers D and B are put through the quadratic equation which is defined as $f(x) = x(2x + 1)$ and apply circular right shift by $\log_2 w$ bits.
- Apply circular left shift on the resulting value for u and t respectively by $\log_2 w$ bits.
- The subkey $S[2i + 1]$ for that particular round is subtracted from register C. The result is circular right shifted by t bits.
- Subkey $S[2i]$ is subtracted from register A, the result of which is right-shifted by u bits.
- The values of register C and u undergo XOR operation. The values of register A is XOR with the value of t .
- Registers D and B undergo a post-whitening.

5.6 COMPARISON BETWEEN RC6 AND RC5

RC6 added two new features which are not present in RC5. These features are: (i) Integer multiplication modulo 2^w , and (ii) four w -bit word registers A, B, C and

D. RC5 uses two registers of 2-bit. Use of integer multiplication in RC6, increases the diffusion achieved per round. Due to this, the cipher is faster and less number of rounds are required to provide the necessary security.

5.7 IDEA

The international data encryption algorithm (IDEA) is a symmetric encryption block cipher. It is designed by Xuejia Lai and James Massey of ETH Zurich. It was first published in 1991. The main intention of this algorithm is to find out the alternate solution to the data encryption standard (DES).

IDEA algorithm uses the plaintext block of size 64 bits and the key of size 128 bits. There are total nine rounds of which working of the first eight rounds are identical. The last round is a half round which uses only first four steps (operations) of the other rounds. Decryption uses the same steps as encryption. But the subkeys for decryption are different from encryption. Different groups of operations are performed in the round of IDEA. This provides the security and makes cryptanalysis difficult. The different operations include modular addition, modular multiplication and bitwise XOR. These operators are:

- Bitwise XOR
- Addition modulo 2^{16}
- Multiplication modulo $2^{16} + 1$, where the all-zero word (0×0000) is interpreted as 2^{16} .

The architecture of the IDEA is as shown in Figure 5.6 which explains the working of IDEA.

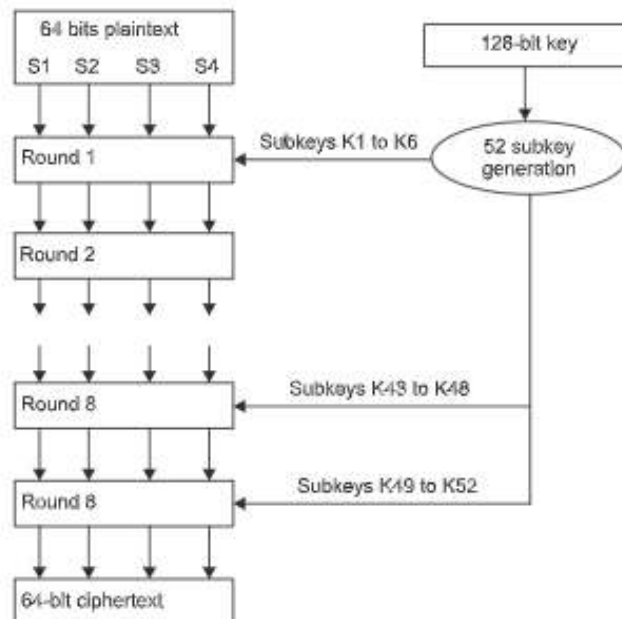


Figure 5.7 IDEA structure.

5.7.1 Working of IDEA

The working of IDEA is divided into two parts:

1. Key generation
2. Encryption

Key Generation

IDEA algorithm has total nine rounds. First eight rounds use six subkeys whereas last round uses only four keys. Therefore, total 52 subkeys of 16 bits are required for encryption. Same number of keys are required for decryption. For decryption, the key generation process is different from key generation process of encryption. The first step of the algorithm is to generate 52 subkeys, K_1 to K_{52} . The original key for IDEA is 128-bit key. This key is used to generate the subkeys. The subkey generation process is as follows.

- (a) Initially, split the 128-bit key into 8 parts of 16 bits each. These are the first eight subkeys K_1 to K_8 . This process is shown in Table 5.1.

Table 5.1 Original key (128-bit)

Bit position	1 to 16	17 to 32	33 to 48	49 to 64	65 to 80	81 to 96	97 to 112	113 to 128
Subkey	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8

- (b) Then apply circular left shift by 25 bits position on the 128-bit key and split the key again into eight parts of 16 bits each. As shown in Table 5.2, this gives next 8 subkeys, i.e., K_9 to K_{16} . Subkey K_9 is started from bit 26 to 41 of the original key. Subkey K_{16} having bits 122 to 128 and bits 1 to 9 of the original key. The process is as shown in the Table 5.2.

Table 5.2 128-bit key

Bit position	26 to 41	42 to 57	58 to 73	74 to 89	90 to 105	106 to 121	122 to 9	10 to 25
Subkey	K_9	K_{10}	K_{11}	K_{12}	K_{13}	K_{14}	K_{15}	K_{16}

- (c) Repeat step (b) until all 52 subkeys are generated. Subkeys K_{17} to K_{24} are generated by starting from bit number 51 of the original key. Subkeys K_{25} to K_{32} are generated by starting from bit number 76 of the original key. Subkeys K_{33} to K_{40} are generated by starting from bit number 101 of the original key. Subkeys K_{41} to K_{49} are generated by starting from bit number 125 of the original key. Subkeys K_{40} to K_{52} are generated by starting from bit number 22 of the original key.

The repetitions in the subkeys are avoided due to circular left shift of 25 bits.

Encryption

In IDEA algorithm there are total nine rounds, eight complete and one half rounds. The encryption is done using 52 subkeys such that the first eight rounds use 6 subkeys

each and the last round uses 4 subkeys. So, the first round uses subkeys K_1 to K_4 . The transformation operations use field $F(2^{16}+1)$, i.e., the output will be calculated using modulo $(2^{16}+1)$. A round looks like this:

The message is split into plaintext blocks of 64 bits each. Then a plaintext block is split into four sub-blocks of 16 bits each. Suppose the sub-blocks are P_1, P_2, P_3 , and P_4 . K_1 to K_4 are the subkeys and S_1 to S_{14} are the result of respective step.

$$\begin{aligned} P_1 \times K_1 &\rightarrow S_1 \\ P_2 + K_2 &\rightarrow S_2 \\ P_3 + K_3 &\rightarrow S_3 \\ P_4 \times K_4 &\rightarrow S_4 \\ S_1 \oplus S_3 &\rightarrow S_5 \\ S_2 \oplus S_4 &\rightarrow S_6 \\ S_5 \times K_5 &\rightarrow S_7 \\ S_6 + S_7 &\rightarrow S_8 \\ S_8 \times K_6 &\rightarrow S_9 \\ S_7 + S_9 &\rightarrow S_{10} \\ S_1 \oplus S_9 &\rightarrow S_{11} \\ S_3 \oplus S_9 &\rightarrow S_{12} \\ S_2 \oplus S_{10} &\rightarrow S_{13} \\ S_4 \oplus S_{10} &\rightarrow S_{14} \end{aligned}$$

Here multiplication modulo $2^{16}+1$ is used which is shown by symbol (\times). Addition modulo 2^{16} is shown by symbol ($+$) and XOR is shown by symbol (\oplus). The output of last four steps is used as sub-blocks of plaintext for the next round. So, we set $P_1 = S_{11}$, $P_2 = S_{12}$, $P_3 = S_{13}$, $P_4 = S_{14}$. The subkeys K_7 to K_{12} are used for the second round. Continue this process for eight rounds. In the last round there are only first four steps. The last four subkeys, i.e., K_{49}, \dots, K_{52} are used for this round. The output is 4 sub-blocks of ciphertext C_1, C_2, C_3 and C_4 which after concatenate gives 64-bit ciphertext. The steps in the last round are shown as:

$$\begin{aligned} P_1 \times K_{49} &\rightarrow C_1 \\ P_2 + K_{50} &\rightarrow C_2 \\ P_3 + K_{51} &\rightarrow C_3 \\ P_4 \times K_{52} &\rightarrow C_4 \end{aligned}$$

IDEA algorithm for single round is as follows:

ALGORITHM

1. Multiplication modulo $2^{16}+1$ between P_1 and the first subkey K_1 . The result is S_1 .
2. Addition modulo 2^{16} between P_2 and the second subkey K_2 . The result is S_2 .
3. Addition modulo 2^{16} between P_3 and the third subkey K_3 . The result is S_3 .
4. Multiplication modulo $2^{16}+1$ between P_4 and the fourth subkey K_4 . The result is S_4 .
5. Apply XOR between S_1 and S_3 . The result is S_5 .

6. Apply XOR between S_2 and S_4 . The result is S_6 .
7. Multiplication modulo $2^{16}+1$ between S_6 and the fifth subkey K_5 . The result is S_7 .
8. Addition modulo 2^{16} between S_5 and S_7 . The result is S_8 .
9. Multiplication modulo $2^{16}+1$ between S_8 and the sixth subkey K_6 . The result is S_9 .
10. Addition modulo 2^{16} between S_7 and S_9 . The result is S_{10} .
11. Apply XOR between S_1 and S_9 . The result is S_{11} .
12. Apply XOR between S_3 and S_9 . The result is S_{12} .
13. Apply XOR between S_2 and S_{10} . The result is S_{13} .
14. Apply XOR between S_4 and S_{10} . The result is S_{14} .

The working of the single round of IDEA is shown in Figure 5.8.

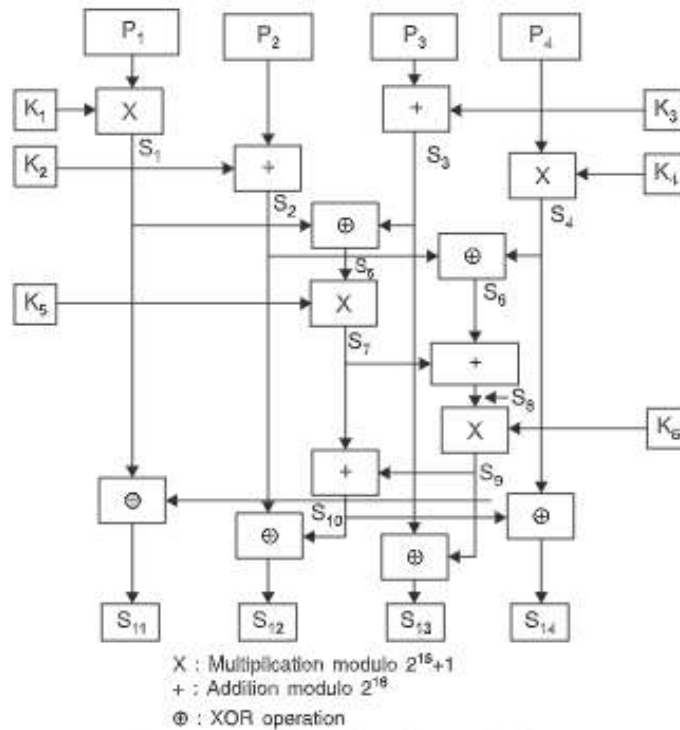


Figure 5.8 Working of single round of IDEA.

The last round (9th round) of IDEA has only first four steps. It uses only four subkeys K_{49} to K_{52} . The output of this round is C_1 , C_2 , C_3 and C_4 . After concatenating C_1 , C_2 , C_3 and C_4 the ciphertext is generated for one plaintext block. Repeat the same procedure for all the plaintext blocks, the complete ciphertext is generated. The working of last round of IDEA is shown in Figure 5.9.

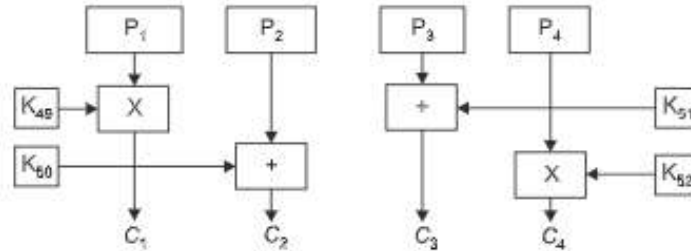


Figure 5.9 Working of last round of IDEA.

5.7.2 Decryption

In IDEA algorithm, the decryption process is exactly the same as the encryption process, except the use of subkeys. The subkeys for decryption are generated from the subkeys used for encryption. This makes it difficult for cryptanalysis. For generation of subkeys, reverse order subkeys are used, i.e., encryption subkeys K_{48} to K_{52} are used to generate decryption subkeys K_1 to K_4 . For generation of these subkeys, multiplicative inverse and additive inverse are used.

Suppose the key for encryption is denoted by K and the key for decryption is denoted by Z . Then, K_j^i denotes the j th subkey for i th round of the encryption and Z_j^i denotes the j th subkey for i th round of the decryption. For the first round of decryption $i = 1$, and for first subkey $j = 1$. The decryption subkey is calculated from the 9th round (last round) and first subkey. Therefore, $Z_1^1 = (K_1^9)^{-1}$ where $(K_1^9)^{-1}$ denotes the multiplicative inverse of the first encryption key of encryption round 9. Second subkey can be calculated using, $Z_2^1 = -(K_2^9)$ where $-(K_2^9)$ denotes the additive inverse of the second encryption key of encryption round 9 modulo 16; $Z_3^1 = -(K_3^9)$, $Z_4^1 = -(K_4^9)^{-1}$. The last two subkeys for the first round of decryption is the last two keys of 8th round such as $Z_5^1 = (K_5^8)$, $Z_6^1 = (K_6^8)$. Similarly, the subkeys for all the eight rounds of decryption are generated. The subkeys for the last 9th round of the decryption are: $Z_1^9 = (K_1^1)^{-1}$, $Z_2^9 = -(K_2^1)$, $Z_3^9 = -(K_3^1)$, $Z_4^9 = (K_4^1)^{-1}$. The subkeys for the first round of decryption are illustrated from the following examples.

The decryption subkey Z_1 is the multiplicative inverse of K_{48} . This is illustrated by following example. Suppose, the subkey K_{48} is 0111 (decimal number is 7_{10}). The multiplicative inverse of $7 \bmod 17$ is 5 (i.e., $7 * 5 \bmod 17 = 1$). So, the decryption subkey equivalent to 0111 (7_{10}) is 0101 (5_{10}) which is the first subkey Z_1 for decryption.

The decryption subkeys Z_2 and Z_3 are the additive inverse of K_{50} and K_{51} . This is illustrated by following example. Suppose, the subkey K_{50} is 1001 (decimal number is 9_{10}). The additive inverse of $9 \bmod 16$ is 7 (i.e., $9 + 7 \bmod 16 = 0$). So, the decryption subkey equivalent to 1001 (9_{10}) is 0111 (7_{10}) which is the second subkey Z_2 for decryption. This can be done by subtracting the decimal value of subkey from 16. Here, $16 - 9 = 7$. So, the subkey Z_2 is 7.

Suppose the subkey K_{51} is 1101 (decimal number is 13_{10}). The additive inverse of $13 \bmod 16$ is 3 (i.e., $13 + 3 \bmod 16 = 0$). So, the decryption subkey equivalent to 1101 (13_{10}) is 0011 (3_{10}) which is the third subkey Z_3 for decryption. This can be done by subtracting the decimal value of subkey from 16. Here, $16 - 13 = 3$. So, the decryption subkey Z_3 is 3.

The decryption subkey Z_4 is the multiplicative inverse of K_{52} . This is illustrated by following example. Suppose the subkey K_{52} is 1010 (decimal number is 10_{10}). The multiplicative inverse of $10 \bmod 17$ is 12 (i.e., $10 * 12 \bmod 17 = 1$). So, the decryption subkey equivalent to 1010 (10_{10}) is 1100 (12_{10}) which is the fourth subkey Z_4 for decryption.

The decryption subkey Z_5 and Z_6 are the last two subkeys of 8th round. $Z_5 = K_{47}$ and $Z_6 = K_{48}$. Suppose, the subkeys K_{47} is 1100 and K_{48} is 0110, then Z_5 is 1100 and Z_6 is 0110. In this way the subkeys for all the rounds of decryption are generated.

5.7.3 Security

Cryptanalysis is possible against IDEA algorithm but this is for 5 rounds. Brute force attack is possible but it takes large time due to 128-bit key. There are weak keys in IDEA. So, key selection for IDEA should be done carefully. Against IDEA algorithm, linear cryptanalysis attack is not reported where differential cryptanalysis attack is possible.

SOLVED PROBLEMS

5.1 Encrypt the following message using IDEA algorithm

Message: 1111 1011 1101 1010
Key: 10101001110111110110010111000011

Solution:

Plaintext: 1111 1011 1101 1010
Key: 1010 1001 1101 1111 0110 0101 1100 0011
0111 0111 1101 1001 0111 0000 1110 1010
1111 0110 0101 1100 0011 1010 1001 1101
1001 0111 0000 1110 1010 0111 0111 1101

Generation of encryption keys:

	Round-1	Round-2	Round-3	Round-4	Round-5
Key-1	1010	1100	0111	0101	1001
Key-2	1001	0011 *	0000	1100	0111
Key-3	1101	0111	1110	0011	0000
Key-4	1111	0111	1010 *	1010	1110
Key-5	0110	1101	1111	1001	
Key-6	0101	1001	0110	1101*	

Inverses of nibbles for addition modulo 16:

Number in binary	Number in decimal	Inverse in binary	Inverse in decimal
0000	0	0000	0
0001	1	1111	15

Number in binary	Number in decimal	Inverse in binary	Inverse in decimal
0010	2	1110	14
0011	3	1101	13
0100	4	1100	12
0101	5	1011	11
0110	6	1010	10
0111	7	1001	9
1000	8	1000	8
1001	9	0111	7
1010	10	0110	6
1011	11	0101	5
1100	12	0100	4
1101	13	0011	3
1110	14	0010	2
1111	15	0001	1

Inverses of nibbles for multiplication modulo 17:

Number in binary	Number in decimal	Inverse in binary	Inverse in decimal
0001	1	0001	1
0010	2	1001	9
0011	3	0110	6
0100	4	1101	13
0101	5	0111	7
0110	6	0011	3
0111	7	0101	5
1000	8	1111	15
1001	9	0010	2
1010	10	1100	12
1011	11	1110	14
1100	12	1010	10
1101	13	0100	4
1110	14	1011	11
1111	15	1000	8
0000	16 = -1	0000	16 = -1

Round-1

$S_1 = P_1 \times K_1$	$S_2 = P_2 + K_2$	$S_3 = P_3 + K_3$	$S_4 = P_4 \times K_4$
1111	1011	1101	1010
1010	1001	1101	1111
1110	0100	1010	1110
$S_5 = S_1 \oplus S_3$			$S_6 = S_2 \oplus S_4$
1110			0100
1010			1110
0100			1010
$S_7 = S_3 \times K_5$			$S_8 = S_6 + S_7$
0100			1010
0110			0111
0111			0001
$S_{10} = S_7 + S_9$			$S_9 = S_8 \times K_6$
0111			0001
0101			0101
1100			0101
$S_{11} = S_9 \oplus S_1$	$S_{12} = S_9 \oplus S_3$	$S_{13} = S_{10} \oplus S_2$	$S_{14} = S_{10} \oplus S_4$
0101	0101	1100	1100
1110	1010	0100	1110
1011	1111	1000	0010
$P_1 = S_{11}$	$P_2 = S_{13}$	$P_3 = S_{12}$	$P_4 = S_{14}$
1011	1000	1111	0010

Round-2

$S_1 = P_1 \times K_1$	$S_2 = P_2 + K_2$	$S_3 = P_3 + K_3$	$S_4 = P_4 \times K_4$
1011	1000	1111	0010
1100	0011	0111	0111
1101	1011	0110	1110
$S_5 = S_1 \oplus S_3$			$S_6 = S_2 \oplus S_4$
1101			1011
0110			1110
1011			0101

$$\begin{array}{r}
 S_7 = S_5 \times K_5 \\
 1011 \\
 1101 \\
 0111 \\
 \\
 S_{10} = S_7 + S_9 \\
 0111 \\
 0110 \\
 1101 \\
 \\
 S_{11} = S_9 \oplus S_1 \\
 0110 \\
 1101 \\
 1011 \\
 \\
 P_1 = S_{11} \\
 1011
 \end{array}
 \qquad
 \begin{array}{r}
 S_8 = S_6 + S_7 \\
 0101 \\
 0111 \\
 1100 \\
 \\
 S_9 = S_8 \times K_6 \\
 1100 \\
 1001 \\
 0110 \\
 \\
 S_{12} = S_9 \oplus S_3 \\
 0110 \\
 0110 \\
 0000 \\
 \\
 P_2 = S_{12} \\
 0110
 \end{array}
 \qquad
 \begin{array}{r}
 S_{13} = S_{10} \oplus S_2 \\
 1101 \\
 1011 \\
 0110 \\
 \\
 P_3 = S_{12} \\
 0000
 \end{array}
 \qquad
 \begin{array}{r}
 S_{14} = S_{10} \oplus S_4 \\
 1101 \\
 1110 \\
 0011 \\
 \\
 P_4 = S_{14} \\
 0011
 \end{array}$$

Round-3

$$\begin{array}{r}
 S_1 = P_1 \times K_1 \\
 1011 \\
 0111 \\
 1001 \\
 \\
 S_2 = P_2 + K_2 \\
 0110 \\
 0000 \\
 0110 \\
 \\
 S_3 = P_3 + K_3 \\
 0000 \\
 1110 \\
 1110 \\
 \\
 S_4 = P_4 \times K_4 \\
 0011 \\
 1010 \\
 1101 \\
 \\
 S_5 = S_1 \oplus S_3 \\
 1001 \\
 1110 \\
 0111 \\
 \\
 S_6 = S_2 \oplus S_4 \\
 0110 \\
 1101 \\
 1011 \\
 \\
 S_7 = S_5 \times K_5 \\
 0111 \\
 1111 \\
 0011 \\
 \\
 S_8 = S_6 + S_7 \\
 1011 \\
 0011 \\
 1110 \\
 \\
 S_{10} = S_7 + S_9 \\
 0011 \\
 0000 \\
 0011 \\
 \\
 S_9 = S_8 \times K_6 \\
 1110 \\
 0110 \\
 0000
 \end{array}$$

$$\begin{array}{cccc}
 S_{11} = S_9 \oplus S_1 & S_{12} = S_9 \oplus S_3 & S_{13} = S_{10} \oplus S_2 & S_{14} = S_{10} \oplus S_4 \\
 0000 & 0000 & 0011 & 0011 \\
 1001 & 1110 & 0110 & 1101 \\
 1001 & 1110 & 0101 & 1110
 \end{array}$$

$$\begin{array}{cccc}
 P_1 = S_{11} & P_2 = S_{13} & P_3 = S_{12} & P_4 = S_{14} \\
 1001 & 0101 & 1110 & 1110
 \end{array}$$

Round-4

$$\begin{array}{cccc}
 S_1 = P_1 \times K_1 & S_2 = P_2 + K_2 & S_3 = P_2 + K_2 & S_4 = P_4 \times K_4 \\
 1001 & 0101 & 1110 & 1110 \\
 0101 & 1100 & 0011 & 1010 \\
 1011 & 0001 & 0001 & 0100
 \end{array}$$

$$\begin{array}{ccc}
 S_5 = S_1 \oplus S_3 & & S_6 = S_2 \oplus S_4 \\
 1011 & & 0001 \\
 0001 & & 0100 \\
 1010 & & 0101
 \end{array}$$

$$\begin{array}{ccc}
 S_7 = S_5 \times K_5 & & S_8 = S_6 + S_7 \\
 1010 & & 0101 \\
 1001 & & 0101 \\
 0101 & & 1010
 \end{array}$$

$$\begin{array}{ccc}
 S_{10} = S_7 + S_9 & & S_9 = S_8 \times K_6 \\
 0101 & & 1010 \\
 1011 & & 1101 \\
 0000 & & 1011
 \end{array}$$

$$\begin{array}{cccc}
 S_{11} = S_9 \oplus S_1 & S_{12} = S_9 \oplus S_3 & S_{13} = S_{10} \oplus S_2 & S_{14} = S_{10} \oplus S_4 \\
 1011 & 1011 & 0000 & 0000 \\
 1011 & 0001 & 0001 & 0100 \\
 0000 & 1010 & 0001 & 0100
 \end{array}$$

$$\begin{array}{cccc}
 P_1 = S_{11} & P_2 = S_{13} & P_3 = S_{12} & P_4 = S_{14} \\
 0000 & 0001 & 1010 & 0100
 \end{array}$$

Round-5

$$\begin{array}{cccc}
 S_1 = P_1 \times K_1 & S_2 = P_2 + K_2 & S_3 = P_3 + K_3 & S_4 = P_4 \times K_4 \\
 0000 & 0001 & 1010 & 0100 \\
 1001 & 0111 & 0000 & 1110 \\
 0000 & 1000 & 1010 & 0101
 \end{array}$$

The ciphertext is: 0000 1000 1010 0101

5.2 Generate the key for decryption from the following encryption key.

Key: 10101001110111110110010111000011

Solution:

	(K_j^i)	Integer	Inverse in integer	Z_j^i	Key for 1st round
(K_1^5)	1001	9	2 (multiplicative modulo 17)	0010	Z_1^1
(K_2^5)	0111	7	6 (addition modulo 16)	1001	Z_2^1
(K_3^5)	0000	0	(Addition modulo 16)	0000	Z_3^1
(K_4^5)	1110	14	11 (Multiplicative modulo 17)	1011	Z_4^1
(K_5^4)	1001	9	9	1001	Z_5^1
(K_6^4)	1101	13	13	1101	Z_6^1

We can generate the keys as above for all 5 rounds as shown below:

Generation of decryption keys:

	Round-1	Round-2	Round-3	Round-4	Round-5
Key-1	0010	0111	0101	1010	1100
Key-2	1001	0100	0000	1101	0111
Key-3	0000	1101	0010	1001	0011
Key-4	1011	1100	1100	0101	1000
Key-5	1001	1111	1101	0110	
Key-6	1101	0110	1001	0101	

5.3 Use IDEA algorithm for encryption and decryption of the following message:

Message: 1110 1111 1001 0101

Key: 1100 0110 0011 0101 1111 0111 0101 1010

Solution:

Plaintext: 1110 1111 1001 0101

Key: 1100 0110 0011 0101 1111 0111 0101 1010

Generation of encryption keys:

	Round-1	Round-2	Round-3	Round-4	Round-5
Key-1	1100	0101	1101	0111	1101
Key-2	0110	1010	0110	0101	1101
Key-3	0011	1000	1011	1010	0110
Key-4	0101	1101	0001	1100	1011
Key-5	1111	0111	0101	0110	
Key-6	0111	1101	1111	0011	

Round-1

$$\begin{array}{llll}
 S_1 = P_1 \times K_1 & S_2 = P_2 + K_2 & S_3 = P_3 + K_3 & S_4 = P_4 \times K_4 \\
 1110 & 1111 & 1001 & 0101 \\
 1100 & 0110 & 0011 & 0101 \\
 1111 & 0101 & 1100 & 1000
 \end{array}$$

$$\begin{array}{ll}
 S_5 = S_1 \oplus S_3 & S_6 = S_2 \oplus S_4 \\
 1111 & 0101 \\
 1100 & 1000 \\
 0011 & 1101
 \end{array}$$

$$\begin{array}{ll}
 S_7 = S_5 \times K_5 & S_8 = S_6 + S_7 \\
 0011 & 1101 \\
 1111 & 1011 \\
 1011 & 1000
 \end{array}$$

$$\begin{array}{ll}
 S_{10} = S_7 + S_9 & S_9 = S_8 \times K_6 \\
 1011 & 1000 \\
 0101 & 0111 \\
 0000 & 0101
 \end{array}$$

$$\begin{array}{llll}
 S_{11} = S_9 \oplus S_1 & S_{12} = S_9 \oplus S_3 & S_{13} = S_{10} \oplus S_2 & S_{14} = S_{10} \oplus S_4 \\
 0101 & 0101 & 0000 & 0000 \\
 1111 & 1100 & 0101 & 1000 \\
 1010 & 1001 & 0101 & 1000
 \end{array}$$

$$\begin{array}{llll}
 P_1 = S_{11} & P_2 = S_{12} & P_3 = S_{13} & P_4 = S_{14} \\
 1010 & 0101 & 1001 & 1000
 \end{array}$$

Input to Round-2: 1010 0101 1001 1000

Round-2

$S_1 = P_1 \times K_1$	$S_2 = P_2 + K_2$	$S_3 = P_3 + K_3$	$S_4 = P_4 \times K_4$
1010	0101	1001	1000
0101	1010	1000	1101
0000	1111	0001	0010
$S_5 = S_1 \oplus S_3$			$S_6 = S_2 \oplus S_4$
0000			1111
0001			0010
0001			1101
$S_7 = S_3 \times K_5$			$S_8 = S_6 + S_7$
0001			1101
0111			0111
0111			0100
$S_{10} = S_7 + S_9$			$S_9 = S_8 \times K_6$
0111			0100
0001			1101
1000			0001
$S_{11} = S_9 \oplus S_1$	$S_{12} = S_9 \oplus S_3$	$S_{13} = S_{10} \oplus S_2$	$S_{14} = S_{10} \oplus S_4$
0001	0001	1000	1000
0000	0001	1111	0010
0001	0000	0111	1010
$P_1 = S_{11}$	$P_2 = S_{13}$	$P_3 = S_{12}$	$P_4 = S_{14}$
0001	0111	0000	1010

Input to Round-3: 0001 0111 0000 1010

Round-3

$S_1 = P_1 \times K_1$	$S_2 = P_2 + K_2$	$S_3 = P_3 + K_3$	$S_4 = P_4 \times K_4$
0001	0111	0000	1010
1101	0110	1011	0001
1101	1101	1011	1010
$S_5 = S_1 \oplus S_3$			$S_6 = S_2 \oplus S_4$
1101			1101
1011			1010
0110			0111

$S_7 = S_5 \times K_5$			$S_8 = S_6 + S_7$
0110			0111
0101			1101
1101			0100
$S_{10} = S_7 + S_9$			$S_9 = S_8 \times K_6$
1101			0100
1001			1111
0110			1001
$S_{11} = S_9 \oplus S_1$	$S_{12} = S_9 \oplus S_3$	$S_{13} = S_{10} \oplus S_2$	$S_{14} = S_{10} \oplus S_4$
1001	1001	0110	0110
1101	1011	1101	1010
0100	0010	1011	1100
$P_1 = S_{11}$	$P_2 = S_{13}$	$P_3 = S_{12}$	$P_4 = S_{14}$
0100	1011	0010	1100

Input for Round-4: 0100 1011 0010 1100

Round-4

$S_1 = P_1 \times K_1$	$S_2 = P_2 + K_2$	$S_3 = P_3 + K_3$	$S_4 = P_4 \times K_4$
0100	1011	0010	1100
0111	0101	1010	1100
1011	0000	1100	1000
$S_5 = S_1 \oplus S_3$			$S_6 = S_2 \oplus S_4$
1011			0000
1100			1000
0111			1000
$S_7 = S_5 \times K_5$			$S_8 = S_6 + S_7$
0111			1000
0110			1000
1000			0000
$S_{10} = S_7 + S_9$			$S_9 = S_8 \times K_6$
1000			0000
1110			0011
0110			1110

$$\begin{array}{cccc}
 S_{11} = S_9 \oplus S_1 & S_{12} = S_9 \oplus S_3 & S_{13} = S_{10} \oplus S_2 & S_{14} = S_{10} \oplus S_4 \\
 1110 & 1110 & 0110 & 0110 \\
 1011 & 1100 & 0000 & 1000 \\
 0101 & 0010 & 0110 & 1110
 \end{array}$$

$$\begin{array}{cccc}
 P_1 = S_{11} & P_2 = S_{13} & P_3 = S_{12} & P_4 = S_{14} \\
 0101 & 0110 & 0010 & 1110
 \end{array}$$

Input to Round-5: 0101 0110 0010 1110

Round-5

$$\begin{array}{cccc}
 S_1 = P_1 \times K_1 & S_2 = P_2 + K_2 & S_3 = P_3 + K_3 & S_4 = P_4 \times K_4 \\
 0101 & 0110 & 0010 & 1110 \\
 1101 & 1101 & 0110 & 1011 \\
 1110 & 0011 & 1000 & 0001
 \end{array}$$

The ciphertext is: 1110 0011 1100 0001

Decryption

New keys are generated

K_j^i - j th encryption key for encryption round i

Z_j^i - j th decryption key for decryption round i

$Z_1^1 = (K_1^5)^{-1}$, $Z_2^1 = -K_2^5$, $Z_3^1 = -K_3^5$, $Z_4^1 = (K_4^5)^{-1}Z_5^1 = K_5^4$, $Z_5^1 = K_6^4$

For remaining rounds, decryption keys are similarly generated.

	Round-1	Round-2	Round-3	Round-4	Round-5
Key-1	0100	0101	0100	0111	1010
Key-2	0011	1011	1010	0110	1010
Key-3	1010	0110	0101	1000	1101
Key-4	1110	1010	0001	0100	0111
Key-5	0110	0101	0111	1111	
Key-6	0011	1111	1101	0111	

Round-1

$$\begin{array}{cccc}
 S_1 = P_1 \times Z_1 & S_2 = P_2 + Z_2 & S_3 = P_3 + Z_3 & S_4 = P_4 \times Z_4 \\
 1110 & 0011 & 1000 & 0001 \\
 0100 & 0011 & 1010 & 1110 \\
 0101 & 0110 & 0010 & 1110
 \end{array}$$

$S_5 = S_1 \oplus S_3$		$S_6 = S_2 \oplus S_4$	
0101		0110	
0010		1110	
0111		1000	
$S_7 = S_5 \times Z_5$		$S_8 = S_6 + S_7$	
0111		1000	
0110		1000	
1000		0000	
$S_{10} = S_7 + S_9$		$S_9 = S_8 \times Z_4$	
1000		0000	
1110		0011	
0110		1110	
$S_{11} = S_9 \oplus S_1$	$S_{12} = S_9 \oplus S_3$	$S_{13} = S_{10} \oplus S_2$	$S_{14} = S_{10} \oplus S_4$
1110	1110	0110	0110
0101	0010	0110	1110
1011	1100	0000	1000
$P_1 = S_{11}$	$P_2 = S_{12}$	$P_3 = S_{12}$	$P_4 = S_{14}$
1011	0000	1100	1000

Input to Round-2: 1011 0000 1100 1000

Round-2

$S_1 = P_1 \times Z_1$	$S_2 = P_2 + Z_2$	$S_3 = P_3 + Z_3$	$S_4 = P_4 \times Z_4$
1011	0000	1100	1000
0101	1011	0110	1010
0100	1011	0010	1100
$S_5 = S_1 \oplus S_3$		$S_6 = S_2 \oplus S_4$	
0100		1011	
0010		1100	
0110		0111	
$S_7 = S_5 \times Z_5$		$S_8 = S_6 + S_7$	
0110		0111	
0101		1101	
1101		0100	

$$S_{10} = S_7 + S_9 \quad S_9 = S_3 \times Z_6$$

1101	0100
1001	1111
0110	1001

$$S_{11} = S_9 \oplus S_1 \quad S_{12} = S_9 \oplus S_3 \quad S_{13} = S_{10} \oplus S_2 \quad S_{14} = S_{10} \oplus S_4$$

1001	1001	0110	0110
0100	0010	1011	1100
1101	1011	1101	1010

$$P_1 = S_{11} \quad P_2 = S_{13} \quad P_3 = S_{12} \quad P_4 = S_{14}$$

1101	1101	1011	1010
------	------	------	------

Input to Round-3: 1101 1101 1011 1010

Round-3

$$S_1 = P_1 \times Z_1 \quad S_2 = P_2 + Z_2 \quad S_3 = P_3 + Z_3 \quad S_4 = P_4 \times Z_4$$

1101	1101	1011	1010
0100	1010	0101	0001
0001	0111	0000	1010

$$S_5 = S_1 \oplus S_3 \quad S_6 = S_2 \oplus S_4$$

0001	0111
0000	1010
0001	1101

$$S_7 = S_5 \times Z_5 \quad S_8 = S_6 + S_7$$

0001	1101
0111	0111
0111	0100

$$S_{10} = S_7 + S_9 \quad S_9 = S_3 \times Z_6$$

0111	0100
0001	1101
1000	0001

$$S_{11} = S_9 \oplus S_1 \quad S_{12} = S_9 \oplus S_3 \quad S_{13} = S_{10} \oplus S_2 \quad S_{14} = S_{10} \oplus S_4$$

0001	0001	1000	1000
0001	0000	0111	1010
0000	0001	1111	0010

$$P_1 = S_{11} \quad P_2 = S_{13} \quad P_3 = S_{12} \quad P_4 = S_{14}$$

0000	1111	0001	0010
------	------	------	------

Input to Round-4: 0000 1111 0001 0010

Round-4

$S_1 = P_1 \times Z_1$	$S_2 = P_2 + Z_2$	$S_3 = P_3 + Z_3$	$S_4 = P_4 \times Z_4$
0000	1111	0001	0010
0111	0110	1000	0100
1010	0101	1001	1000
$S_5 = S_1 \oplus S_3$			$S_6 = S_2 \oplus S_4$
1010			0101
1001			1000
0011			1101
$S_7 = S_5 \times Z_5$			$S_8 = S_6 + S_7$
0011			1101
1111			1011
1011			1000
$S_{10} = S_7 + S_9$			$S_9 = S_3 \times Z_6$
1011			1000
0101			0111
0000			0101
$S_{11} = S_9 \oplus S_1$	$S_{12} = S_9 \oplus S_3$	$S_{13} = S_{10} \oplus S_2$	$S_{14} = S_{10} \oplus S_4$
0101	0101	0000	0000
1010	1001	0101	0000
1111	1100	0101	1000
$P_1 = S_{11}$	$P_2 = S_{13}$	$P_3 = S_{12}$	$P_4 = S_{14}$
1111	0101	1100	1000

Input to Round-5: 1111 0101 1100 1000

Round-5

$S_1 = P_1 \times Z_1$	$S_2 = P_2 + Z_2$	$S_3 = P_3 + Z_3$	$S_4 = P_4 \times Z_4$
1111	0101	1100	1000
1010	1010	1101	0111
1110	1111	1001	0101

The plaintext is: 1110 1111 1001 0101

Public Key Cryptosystems

7.1 INTRODUCTION

Encryption techniques are of two types, symmetric encryption and asymmetric encryption. In symmetric encryption same key is used by both the users for encryption as well as decryption. But the encryption is done by the sender and decryption is done by the recipient. So, both the users should know the key. As the sender and recipient are located at different physical locations, so the key should be transmitted securely from sender to recipient. This needs that the transmission channel should be secure so that the key transmitted securely between two users. But if the transmission channel is secure, then not only key, the message also transmitted securely. So, in this case encryption is not required. That means, this channel is not 100% secure. So, some mechanism is required for transmission of key. This secure transmission of key is called *key distribution*.

In asymmetric encryption, two different keys are used, one for encryption and another for decryption. These keys are mathematically related to each other. Sometime, in asymmetric encryption, two keys are used by each user, i.e., public key and private key. Public key is publicly available so that not only sender and recipient but anybody may know the key. Another key is private key which is a secret key known to the originator (owner) of the key. In all these techniques, some mechanism called key distribution technique is used.

For symmetric encryption, initially both the users should agree on the key. This key may be transmitted through e-mail, phone, postal system or some other medium. The security of the message is dependent on the security of the key. The key should be known to the sender and the recipient. If someone other than the sender and the recipient knows the key, then he/she can misuse it and decrypt the message. This key distribution is the most important part of the symmetric encryption system. To maintain the security of the key, only the secrecy of the key in transmission is not sufficient but one should take care of secrecy of key from the creation of the key to

distribution and storage of it. The different steps used for secrecy of key between the sender and the recipient are called *key management*. Key management includes:

1. Authentication of the users of the key
2. Generation of key
3. Distribution of key
4. Storage of key

In cryptography, there are different algorithms which support for key management. RSA algorithm, the well known key generation algorithm, provides the mechanism for securely generation of key. We will learn more detail of RSA in Section 7.3. Diffie-Hellman algorithm solves the problem of key distribution. In this algorithm, using the public key of the users, the session key is generated without transmitting the private key of the users. We will learn more detail about Diffie-Hellman algorithm in Chapter 8.

Asymmetric cryptography is also known as public key cryptography. We know that asymmetric key encryption algorithms are of two types:

- (i) There are two keys, one for each user;
- (ii) There are four keys, each user have a pair of keys; the public key-private key.

There is no need to keep the public key secret. For type 1 algorithms, both the keys must be kept secret. Symmetric encryption algorithms have some drawbacks. Cryptanalysis is easy. Weak key problem with some symmetric encryption blocks ciphers. Key distribution is the major issue with symmetric encryption. Authentication of the users cannot be done in symmetric encryption. Public key cryptography provides the solution for these issues. Asymmetric encryption algorithms cannot be decrypted easily. Key distribution is not required as each user have own keys. Therefore, public key cryptography provides more security as compared to symmetric encryption.

7.2 PUBLIC KEY CRYPTOGRAPHY

Asymmetric encryption system is also known as *public key cryptography*. It is different from symmetric encryption system. In asymmetric encryption system, a pair of keys is used to call public key and private key. A key which is freely available to all users is called *public key*. Whereas private is a secret key and it is never transmitted from the owner to any other users. Mathematically, it is not possible to determine the private key from the public key. From a pair of keys, either key is used for encryption and other key for decryption.

The components of asymmetric encryptions are:

- Plaintext
- Encryption algorithm
- Public key and private key
- Ciphertext
- Decryption algorithm

The working of public key cryptography is shown in Figure 7.1.

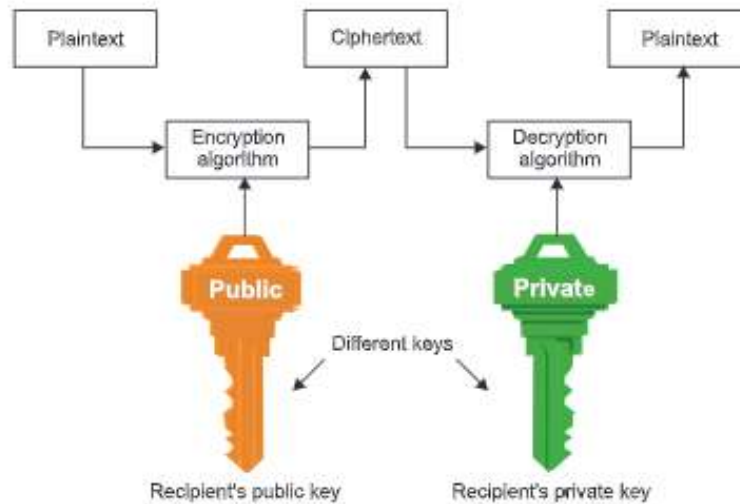


Figure 7.1 Public key cryptography—secrecy.

The scheme showed in Figure 7.1 shows that the user can broadcast his key without keeping it secret. Using the public key of recipient, the sender encrypts the message and sends the ciphertext to the recipient. When the message received, the recipient uses his private key and decrypts the ciphertext. As the private key is known to the recipient only, anybody else cannot decrypt the ciphertext. In this way the secrecy of the message is provided. As the public key of the recipient is publicly available to anybody, any person has the key and can encrypt the message and send it to the recipient. There is no way to check, whether the message is sent by the expected sender or somebody else. So, in this scheme authentication of the sender cannot be taken place.

Asymmetric encryption technique takes more computation time as compared to symmetric encryption technique. Therefore, asymmetric encryption technique is not suitable for large message. But as it is more secure, it is useful to send the key for symmetric encryption. This helps to solve the problem of key distribution in symmetric encryption. The same approach is used in secure socket layer (SSL) protocol.

Alternatively, the sender may use his own private key for encryption. Then the ciphertext is sent to the recipient. The recipient decrypts the ciphertext by using sender's public key. In this scheme, the recipient knows that the ciphertext (message) is sent by the expected sender. So sender's authentication takes place. But at the same time, the ciphertext can be decrypted by any person who knows sender's public key. That means, confidentiality of the message cannot be maintained.

We can summarise asymmetric encryption technique as:

- **Use of two keys:** *private key* and *public key*
- **Private key:** Key is known only by its owner.
- **Public key:** Key is known to everyone.
- **Relation between both keys:** One key is use for encryption and other key is use for decryption, and vice-versa

7.2.1 Authentication, Secrecy and Confidentiality

Authentication means to verify the origin and the integrity of a document. It verifies whether the message is sent by the intended recipient or not. It allows the recipient to verify about the origin of the message. Authentication can be done using different techniques. These techniques are based on secret information. This information should be known only to the users.

Authentication is provided by using a private key for encryption of the message by the sender so that the recipient knows that the message is encrypted by the sender and not anybody else because only the sender knows his own private key. In this, authentication (origin) of the sender is done. As the message is encrypted by the sender's private key and if the attacker captures the data and tries to modify it and sends it, he should know the private key of the sender. And the private key is secret and known to sender only, so the attacker cannot encrypt the message using the same key. If he uses some other key for encryption, then the message cannot be decrypted using sender's public key. Instead of it, attacker's private key is needed which is not available with recipient and the recipient knows that the message is not sent by the authenticated person. In this way authentication and origin of the message and sender can be verified. This also helps to keep the message unchanged, i.e., integrity of the message be maintained. For example, we have our email id and password for the same. This id and password are analogous to public key and the password respectively. One can send the message from his email account by using his password (analogous to private key). The recipient of the message knows that only the intended sender sends the message. So, authentication of the sender takes place. The senders can send the message to anybody using their email id (analogous to public key). The recipient opens this mail account using his password and then read the message. So only authenticate recipient read the message. This provides secrecy.

Figure 7.2 shows the authentication mechanism using private key for encryption and public key for decryption. This provides authentication and not the secrecy.

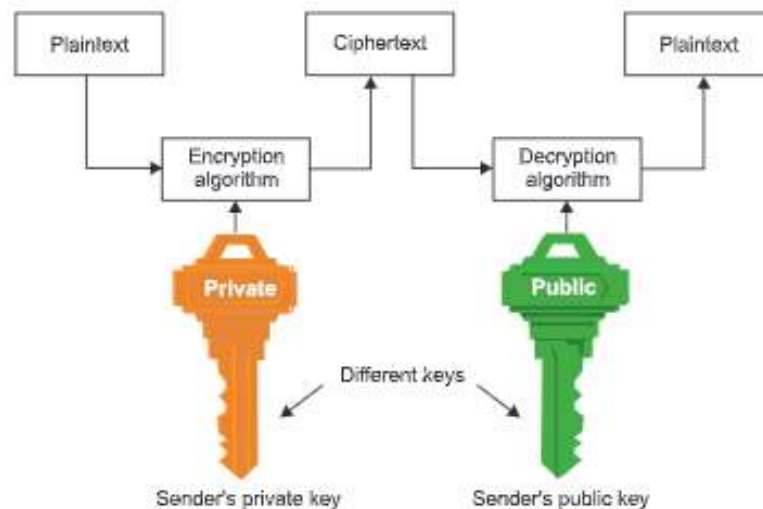


Figure 7.2 Public key encryption—authentication.

Because any third person can have a sender's public key and can decrypt the message sent by the authenticated sender. So, the message cannot be secret using this scheme. Depending on the application, i.e., whether authentication is required for secrecy, one of the above schemes is useful.

In some applications, both authentication and secrecy are required. That means the authentications of both the sender and the recipient should be done. This is called *confidentiality*. Confidentiality provides both the authentication of the users and the recipient and also the integrity (secrecy) of the data. Confidentiality can be provided using the following scheme as shown in Figure 7.3.

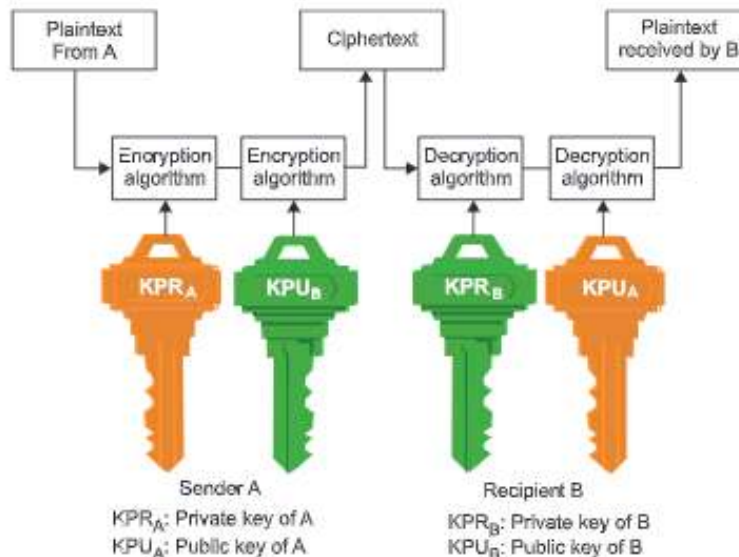


Figure 7.3 Public key encryption—secrecy and authentication.

In confidentiality, each user has two keys, i.e., public key and private key. Two algorithms are used for encryption and decryption. This is because the data or plaintext is encrypted twice before transmitting and also the ciphertext is decrypted twice. The sender first uses his private key to encrypt the message and again encrypts the ciphertext using recipient's public key. When the data is received at the receiving end the recipient's private key is required for decryption. This authenticates that only intended recipient can decrypt the message. Then public key of sender is used again to decrypt the message. This authenticates that the message is sent by the intended sender and not anybody else. In this way confidentiality is provided.

Figure 7.3, illustrated the use of private key and public key to provide authentication of the users and secrecy to the message. It is explained as:

Step 1 User A, i.e., sender, first encrypts the message by his/her own private key. As nobody knows A's private key, so nobody is able to send the encrypted message using A's private key. This provides authentication in the system.

- Step 2** A uses the public key of B, i.e., receiver, to encrypt the ciphertext generated in Step 1.
- Step 3** User A sends the encrypted message to user B.
- Step 4** The private key of B is required to decrypt the message as his public key is used for encryption. B's private key is known to him only so, only B can decrypt the message and nobody else able to do it.
- Step 5** Then decrypt the message using A's, i.e., sender's public key. Steps 3 and 4 provide secrecy to the message.

Therefore, the above scheme provides authentication and secrecy both but not confidentiality. To provide confidentiality one can use following scheme as shown in Figure 7.4.

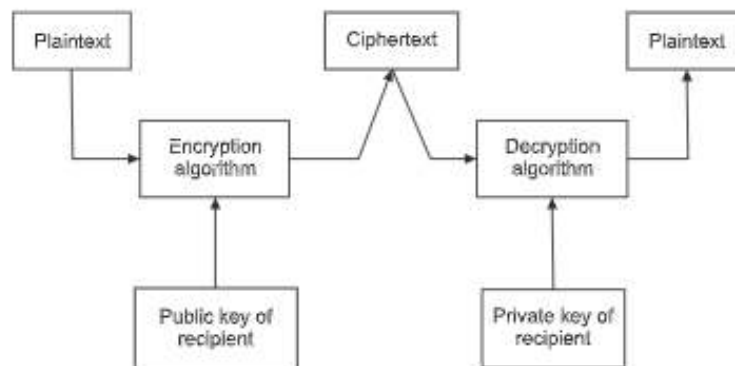


Figure 7.4 Public key encryption—confidentiality.

In this scheme, recipient's public key is used by the sender to encrypt the message. This provides the confidentiality that only recipient can decrypt the message because for decryption, recipient's private key is required. This key is known to recipient, the owner of the key. So other than recipient, nobody is able to decrypt the message and the confidentiality is achieved. At the same time, in this scheme, authentication of the sender cannot be done. So, following scheme is useful which provides authentication, confidentiality and secrecy as shown in Figure 7.5.

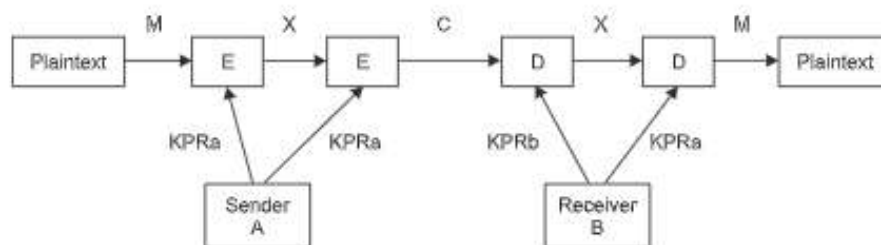


Figure 7.5 Authentications, secrecy and confidentiality.

Suppose A and B are the two users who want to communicate with each other. They want to maintain authentication, secrecy and confidentiality in their transmission. To achieve this, they follow the following steps:

Step 1 KPR_A and KPU_A are the private and public keys of A and KPR_B and KPU_B are the private and public keys of B

Step 2 User A encrypts a message M using his private key KPR_A , the encrypted ciphertext is

$$X = E[KPR_A(M)]$$

Step 3 User A again encrypts the message using B's public key KPU_B , the encrypted ciphertext is

$$\begin{aligned} C &= E[KPU_B(X)] \\ &= E [KPU_B [E[KPR_A(M)]]] \end{aligned}$$

Step 4 User A transmits this encrypted ciphertext to user B.

Step 5 At the receiving end, user B decrypts the message. For this, first B uses his private key KPR_B . The decrypted message is

$$D[KPR_B(M)] = X$$

Step 6 Then user B again decrypts the message using A's public key. The result is original message sent by user A.

$$D[KPU_A(X)] = M$$

This scheme provides authentication, secrecy and confidentiality. The major disadvantage of both the schemes is that these schemes are slow because in these schemes the encryption algorithms are used four times.

7.2.2 Key Length and Encryption Strength

The strength of any encryption algorithm is dependent on: the algorithm used and the length of the encryption key. Long key provides more security to the algorithm. The algorithm is said to be strong if it takes more time to find out the key by the cryptanalyst. RSA algorithm uses prime number theory which makes it difficult to find out the key by reverse engineering.

7.2.3 Applications of Public Key Cryptography

In *pretty good privacy* well known as PGP, public key cryptography is very useful. PGP is used for email application. In communication with a web server, public key cryptography is used to encrypt the message. This provides security to online transaction. In this communication, the communication is encrypted using a random number which is generated by the web browser and the public key of the server. Then the session key is generated by the server and sent securely. Public key cryptography can be used to send the secret key securely in symmetric encryption techniques. Public key cryptography is also used in digital signature where the user signs a message by encrypting using his own private key.

7.2.4 Strength and Weakness of Public Key

Asymmetric encryption technique can be used for key distribution in symmetric encryption. It can be used for providing confidentiality, authentication of the two parties in communication and secrecy to the message.

Asymmetric encryption technique is very slow, so cannot be suitable for encryption of large data. Misuse of public key is possible. Known ciphertext attack is possible against this technique.

7.2.5 Comparison of Asymmetric Encryption and Symmetric Encryption

We know about asymmetric encryption and symmetric encryption techniques. The comparison between asymmetric encryption and symmetric encryption is given in Table 7.1.

Table 7.1 Comparison between asymmetric and symmetric encryption

Asymmetric encryption	Symmetric encryption
Well-known as public key encryption.	Well-known as secret key encryption.
Two keys are used: public key and private key.	One key is used.
Public key is freely available to all. It is not secret key. Private key is a secret key.	The key should be kept secret.
One key is used for encryption and another key for decryption.	Same key is used for encryption as well as for decryption.
Slower than symmetric encryption.	Usually very fast as compared to asymmetric encryption.
It is generally used for encrypting small message because its computational time is more.	It is used for encrypting small or large message.

7.3 RSA ALGORITHM

RSA algorithm is invented by Ron Rivest, Adi Shamir and Leonard Adleman in 1977. It is currently the most widely used public key encryption method in the world. It is the example of public key cryptosystem. It is an asymmetric encryption technique. This algorithm is based on the property of modular exponentiation. It is the most secure algorithm if the key is sufficiently large. In RSA algorithm, exchange of key is not required. The same algorithm can be used for encryption and decryption.

RSA uses variable key length. The more commonly used secure key of size 1024. The speed of the algorithm depends on the size of the key. Large key makes the algorithm slow but provides more security.

7.3.1 Working of RSA

The working of RSA algorithm is divided into three parts:

1. Key generation

2. Encryption
3. Decryption

Key Generation

- Step 1** Generate two large prime numbers p and q randomly such that $p \neq q$.
- Step 2** Calculate the modulus $n = p \cdot q$ and $m = (p - 1)(q - 1)$
- Step 3** Select the key for encryption e such that e is relatively prime to m and $1 < e < m$.
- Step 4** Calculate the key for decryption d such that

$$d = e^{-1} \pmod{m} \text{ or } 1 = ed \pmod{m}$$

where, d is the multiplicative inverse of $e \pmod{m}$ and $1 < d < m$.

The public key is $KP_U = (e, n)$ is used for encryption.

The private key is $KP_R = (d, n)$ is used for decryption.

For better security the values of p , q and m should be kept secret.

n is the modulus.

e is the encryption exponent.

d is the decryption exponent.

Encryption

The encryption in RSA is exponential operation. The public key $KP_U = (e, n)$ is used for encryption. Suppose the message is represented by a positive integer P . The ciphertext is computed as:

$$C = P^e \pmod{n}$$

Decryption

The decryption in RSA is also exponential operation. The private key $KP_R = (d, n)$ is used for decryption. The plaintext is computed as:

$$P = C^d \pmod{n}$$

RSA Algorithm

- Step 1** Generate two large prime numbers p and q randomly
- Step 2** Calculate modulus $n = p \cdot q$ and $m = (p - 1)(q - 1)$
- Step 3** Select the key for encryption, e . This gives public key $KP_U = (e, n)$.
- Step 4** Calculate the key for decryption, d . This gives private key $KP_R = (d, n)$.
- Step 5** Ciphertext = (Plaintext) ^{e} mod n .
- Step 6** Plaintext = (Ciphertext) ^{d} mod n .

EXAMPLE 7.1 Use RSA to encrypt the message $M = 123$ using following parameters

Press Esc to exit full screen

$$p = 11, q = 3, e = 13.$$

Solution

Key Generation

$$n = p * q = 11 * 3 = 33$$

$$m = (p - 1)(q - 1) = 10 * 2 = 20$$

The given value of $e = 13$ as the encryption key. We use extended Euclidean algorithm to find out the multiplicative inverse of e , i.e., d .

$20 = 13(1) + 7$	$7 = 20 - 13(1)$
$13 = 7(1) + 6$	$6 = 13 - 7(1)$
$7 = 6(1) + 1$	$1 = 7 - 6(1)$
$6 = 1(6) + 0$	

$$1 = 7 - 6(1)$$

$$1 = 7 - [13 - 7(1)] (1)$$

$$1 = 7 - 13(1) + 7(1)$$

$$1 = 7(2) - 13(1)$$

$$1 = [20 - 13(1)](2) - 13(1)$$

$$1 = 20(2) - 13(2) - 13(1)$$

$$1 = 20(2) - 13(3)$$

$$1 = 20(2) + 13(-3)$$

Therefore, -3 is the multiplicative inverse of $13 \bmod 20$. So, $d = 20 - 3 = 17$.

$$\text{Public key } KP_U = (e, n) = (13, 33)$$

$$\text{Private key } KP_R = (d, n) = (17, 33)$$

Encryption

The given message $M = 123$, taking each number separately

$$\begin{aligned} C &= M^e \bmod n = (1)^{13} \bmod 33 = 1 \bmod 33 = 1 \\ &= (2)^{13} \bmod 33 = 8 \bmod 33 = 8 \\ &= (3)^{13} \bmod 33 = 27 \bmod 33 = 27 \end{aligned}$$

Therefore, the ciphertext $C = 1 \ 8 \ 27$.

Decryption

$$\begin{aligned} M' &= C^d \bmod n = (1)^{17} \bmod 33 = 1. \\ &= (8)^{17} \bmod 33 = 2 \bmod 33 = 2 \\ &= (27)^{17} \bmod 33 = 3 \bmod 33 = 3 \end{aligned}$$

The plaintext $P = 1 \ 2 \ 3$.

7.3.2 Key Length

The key size is the most important factor for any encryption algorithm. We know that the key sizes for various symmetric algorithms are different. For DES, it is 64-bit but actual it uses 56-bit whereas triple DES has 192-bit key. Due to long key triple DES is more secure than simple DES though the internal architecture is same for both the algorithms. AES have variable length of key such as 128, 192 and 256 bits. For IDEA algorithm, the key length is 128 bits. RSA algorithm has variable key length. If the user selects a short key, then the processing is fast and the algorithm works efficiently. But at the same time cryptanalysis is easier. If we select long key, it provides more security at the cost of reducing the performance of the algorithm. The better option is to select the key which can balance the performance as well as provides security. The key size 512-bit for RSA is no more secure. So, it is recommended that select 512-bit key. For more security, user can select 2048-bit key. As nowadays, hardware is easily available at low cost, it is easy to break even 2048 bits key.

7.3.3 Security

To provide strong algorithm and make cryptanalysis of RSA algorithm difficult, one should keep the values of p , q and m secret. This is important because if the attacker knows the values of p and q he can easily determine the value of m and if the attacker knows m and n he can compute p and q . Many attacks are possible against RSA algorithm. These include brute force attack, timing attack and mathematical attack. Selecting the large key size makes it difficult for brute force attack. The execution time of RSA depends on the key length. Different key size provides different execution time. The cryptanalyst examines the difference between the execution time for different keys. In the same way if the key size is same but the message is of different length, it also requires different execution time. This helps the cryptanalyst to know the message size. Using timing attack, the attacker tries to find out the secret information. Suppose user A decrypts the ciphertext and the opponent, i.e., cryptanalyst observes the process of decryption. Using his observation he finds out the execution time required for decryption for each ciphertext. From this time, the cryptanalyst may be able to find out the value of decryption key, "d".

Protecting Timing Attack

To protect timing attack, preventive measures should be taken. Some of the preventive measures are as follows:

1. Value of e should be large
 2. Value of n should be used by only one user
 3. Use of padding
 4. Use of delay during encryption
-

SOLVED PROBLEMS

7.1 Find the private key for RSA algorithm if the parameters given are:

$$p = 93, q = 47, e = 21.$$

Solution

$$n = pq = 93 * 47 = 4371$$

$$m = (p - 1)(q - 1) = 92 * 46 = 42328465$$

$$e = 21 \text{ encryption exponent}$$

$$\text{Public key} = (21, 4371)$$

We calculate decryption exponent d using Euclid's algorithm as given below

$4232 = 21(201) + 11$	$11 = 4232 - 21(201)$
$21 = 11(1) + 10$	$10 = 21 - 11(1)$
$11 = 10(1) + 1$	$1 = 11 - 10(1)$
$10 = 1(10) + 0$	

$$1 = 11 - 10(1)$$

$$1 = 11 - [21 - 11(1)](1)$$

$$1 = 11(2) - 21(1)$$

$$1 = [4232 - 21(201)](2) - 21(1)$$

$$1 = 4232(2) - 21(403)$$

$$1 = 4232(2) + 21(-403)$$

The multiplicative inverse of 21 is $-403 = 3829$

Therefore, decryption exponent $d = 3829$

$$\text{Private key} = (3829, 4371)$$

7.2 The parameters given are: $p = 5, q = 17$. Find out the possible public key and private for RSA algorithm. Also encrypt the message "4".

Solution

$$p = 5$$

$$q = 17$$

$$\text{Let } n = p * q$$

$$n = 5 * 17$$

$$= 85$$

$$\text{Let } m = (p - 1)(q - 1)$$

$$m = (5 - 1)(17 - 1)$$

$$= 4 * 16 = 64$$

We first find out the total number of possible values e .

As e is relatively prime to m and $e < m$.

Therefore, total number of possible values of $e = \Phi(m)$.

$$\Phi(m) = \Phi(64) = 64 * (1 - 1/2) = 64 * (1/2) = 32$$

Therefore, the total numbers of possible values of e are 32.

Key Generation

(a) Public key generation

Choose a small number, e which is relatively prime to m .

We use Euclid's algorithm to find the GCD.

$$e = 2 \Rightarrow \text{GCD}(e, 64) = 2$$

$$e = 3 \Rightarrow \text{GCD}(e, 64) = 1$$

$$e = 4 \Rightarrow \text{GCD}(e, 64) = 4$$

$$e = 5 \Rightarrow \text{GCD}(e, 64) = 1$$

$$e = 7 \Rightarrow \text{GCD}(e, 64) = 1$$

Now, we can select any one of these values whose GCD is 1. From above, suppose we select $e = 5$.

The public key is (5, 85).

(b) Private key generation

We can compute decryption exponent d , such that $d = e^{-1} \pmod{m}$.

We use extended Euclidean algorithm to find the value of d .

$$1 = 5 \pmod{64}$$

$64 = 5(12) + 4$	$4 = 64 - 5(12)$
$5 = 4(1) + 1$	$1 = 5 - 4(1)$
$4 = 1(4) + 0$	
$1 = 5 - 4(1)$	
$1 = 5 - [64 - 5(12)](1)$	
$1 = 5(13) - 64(1)$	

Therefore, 13 is the multiplicative inverse of 5 mod 64.

Therefore, $d = 13$ and

Private key is (13, 85).

Encryption

The message is $P = 7$

$$\begin{aligned} C &= P^e \pmod{n} \\ &= 7^5 \pmod{85} \\ &= 62 \end{aligned}$$

The ciphertext is 62.

Decryption

$$\begin{aligned} P &= C^d \pmod{n} \\ &= 62^{13} \pmod{85} \\ &= 7 \end{aligned}$$

We can regenerate the plaintext = 7 from the ciphertext.

- 7.3 The parameters given are: $p = 3$, $q = 19$. Find out the possible public key and private for RSA algorithm. Also encrypt the message "6".

Solution:

$$\begin{aligned} p &= 3, q = 19 \\ n &= 3 \times 19 = 57 \\ m &= \Phi(n) = 2 \times 18 = 36 \end{aligned}$$

Key Generation

- (a) Public key generation

Choose a small number, e which is relatively prime to m , i.e., $\text{GCD}(e, m) = 1$

We use Euclid's algorithm to find the GCD.

$$\begin{aligned} e = 2 &\Rightarrow \text{GCD}(e, 36) = 2 \\ e = 3 &\Rightarrow \text{GCD}(e, 36) = 3 \\ e = 4 &\Rightarrow \text{GCD}(e, 36) = 4 \\ e = 5 &\Rightarrow \text{GCD}(e, 36) = 1 \\ e = 7 &\Rightarrow \text{GCD}(e, 36) = 1 \end{aligned}$$

Now, we can select any one of these values whose GCD is 1. From above, suppose we select $e = 5$ or 7, here we select $e = 7$.

The public key is (7, 57).

- (b) Private key generation

We can compute decryption exponent d , such that $d = e^{-1} \pmod{m}$.

We use extended Euclidean algorithm to find the value of d .

$$1 = 7 \pmod{36}$$

$36 = 7(5) + 1$	$1 = 36 - 7(5)$
$7 = 1(7) + 0$	
$1 = 36 - 7(5)$	
$1 = 36 + 7(-5)$	

Therefore, -5 is the multiplicative inverse of $7 \pmod{36}$.

Therefore, $d = 31$ and (since $36 - 5 = 31$)

Private key is (31, 57)

Encryption

The message is $P = 6$

$$\begin{aligned} C &= P^e \pmod{n} \\ &= 6^7 \pmod{57} \\ &= 9 \end{aligned}$$

The ciphertext is 9.

Decryption

$$\begin{aligned}
 P &= C^d \pmod n \\
 &= 9^{31} \pmod{57} \\
 &= 6
 \end{aligned}$$

We can regenerate the plaintext = 6 from the ciphertext.

- 7.4 In asymmetric encryption using RSA, the ciphertext is $C = 4$ and the public key is: $e = 89$, $p = 11$, $q = 47$.

Find

1. The key for decryption
2. The plaintext P ?

Solution

$$\begin{aligned}
 n &= p * q \\
 n &= 11 * 47 = 517 \\
 m &= \text{Totient function } \phi(n) = 10 * 46 = 460 \\
 e &= 67
 \end{aligned}$$

Public key = (67, 517)

We have to find out the value of d in such a way that $ed \pmod m = 1$.

Therefore, $d = 103$

Private key = (d , n) = (103, 517)

Decryption

The ciphertext is 4.

Plaintext $P = C^d \pmod n$

$$P = 5^{103} \pmod{517} = 344$$

Use modulo arithmetic to compute the value of P

Message $P = 344$

SUMMARY

Encryption techniques are of two types, symmetric encryption and asymmetric encryption. In symmetric encryption same key is used by both the users for encryption as well as decryption. Asymmetric cryptography is also known as *public key cryptography*. In asymmetric encryption, two different keys are used, one for encryption and other for decryption. RSA algorithm, the well known key generation algorithm, provides the mechanism for secure generation of key. Asymmetric encryption technique takes more computation time as compared to symmetric encryption technique. Asymmetric encryption technique is not suitable for large message. But as it is more secure, it is useful to send the key for symmetric encryption. Authentication means to verify the origin and the integrity of a document. It verifies whether the message is sent by the intended recipient or not. RSA is currently the most widely used public key encryption methods in the world. It is the example of public key cryptosystem. To provide strong algorithm and make cryptanalysis of RSA algorithm difficult one should keep the values of p , q and m secret.

CHAPTER 8

Key Management

8.1 INTRODUCTION

In the last chapter we learn various symmetric encryption techniques. Apart from this technique, there is asymmetric encryption technique where two different keys which are mathematically related to each other are used for encryption and decryption. In both of these techniques, i.e., symmetric encryption and asymmetric encryption, the distribution of keys is the most important issues. In this chapter, we discuss different approaches for key distribution.

8.2 KEY DISTRIBUTION

Computer is a necessary part of our day to day life. People use computers to communicate their message. This is possible due to Internet. People can send e-mail, chat their friends, and even speak on phone through internet. Now, we know that this channel of communication is not secure. Every user expects authentication, confidentiality and integrity of the message which he/she sends through computer. To provide these security services, public key infrastructure is most important. Use of these infrastructures depends upon the required degree of security which depends on the applications. Up to last chapter, we discuss various symmetric and asymmetric encryption algorithms. We know that the security strength of these algorithms depends on the security of the key. In symmetric encryption techniques, key distribution or key handling is the major issue. While transferring the key from sender to recipient, if the attackers are able to capture the key, then he can decrypt the message. So, key handling is the most important domain in public key infrastructures. In this chapter, we will focus on this problem of key management.

Asymmetric encryption or public key cryptography has two issues related to keys. We know in public key cryptography, each user has a pair of keys, i.e., public key and private key. During communication, each user should know the public key of another

user. So, the first issue about key in public key infrastructure is the distribution of public keys among the users. The second issue is the private key distribution using public key encryption technique.

There are different approaches for the distribution of the public keys. The keys can be distributed using any one of the approaches given below:

- Public announcement
- Publicly available directory
- Public key authority
- Public key certificates

8.2.1 Public Announcement

In this method, the public key is broadcasted by the owner of the key. Many times users may use pretty good privacy (PGP) for broadcasting the key. The drawback of this method is that anyone can forge key and may misuse it for encryption or decryption of the data as there is no control on the accessing of the key. Figure 8.1 shows the public announcement.

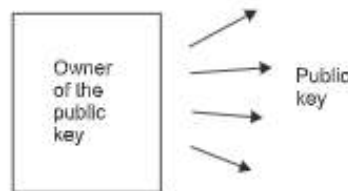


Figure 8.1 Announcement of the public key.

8.2.2 Publicly Available Directory

To avoid the drawback of the above approach, the next approach is used in which a directory of keys is maintained which is available publicly to anybody. In a directory, the public keys of all the users are maintained. This directory is available to all legitimate users of the system. As compared to first approach, this approach provides more security.

The directory is maintained by a trusted party. This party is responsible for the control of use of this directory. The user who wants to be a part of communication system should register his public key in the directory and keep his own private key secret. The directory contains the name of the user with his public key. Figure 8.2 shows the directory structure. The access to this directory is controlled by the trusted party. The directory is protected by password. This password is shared to all the registered users of the directory. So, only registered users of the directory can access it. If the user wants to change his public key, he should inform to the trusted party about the same and store the new public key by replacing the old public one. The advantage of this scheme is that only authorised users of the directory can access the directory and able to get the public key. This scheme can work just like our telephone directory where the telephone numbers of all the users with their names are stored.

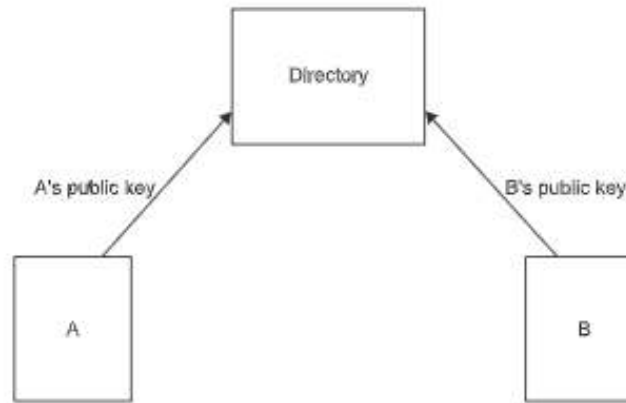


Figure 8.2 Distribution of public key using publicly available directory.

From Figure 8.2, two users A and B are the registered users of the directory. Both users A and B stored their keys in the directory. When the user A wants to communicate with user B, he accesses the directory and get B's public key. Using this key now he can encrypt the message and send it to user B.

If someone is able to get the password of this directory, then he will get the public key of any user stored in the directory and may misuse it.

8.2.3 Public Key Authority

We know that, the public key from the directory can be used if some is able to get the password of the directory. So, there is a need of new mechanism which can distribute the public keys more carefully. The public key authority approach provides the control on the distribution of the keys. The authority keeps the control on the distribution of keys by giving the password. The public key of authority is known to all the registered users. When any user makes a request for public key, the authority first verifies that whether a user is registered user or not. If he is a registered user then authority sends him the requested public key using authority's private key. The users can access the directory by using authority's password.

The communication between the authority and the users occurs as follows:

- Step 1** The sender A sends the request message to the authority. The message contains the request for B's public key with time of request.
- Step 2** Then the authority verifies the authentication of the user A. If user A is the authorised user then the authority responds request of A by sending the message. The message is encrypted with authority's private key. It contains the public key of B with the original request message sent by A.
- Step 3** After receiving the message from the authority, user A decrypts it with the authority's public key. Then A sends a message to user B requesting for communication. This message is encrypted by user B with his public key. The message contains the network address of user A and a random number called *nonce*.

- Step 4** After receiving the message from user A, user B sends the request message to the authority. The message contains the request for A's public key with time of request.
- Step 5** Then the authority verifies the authentication of the user B. If user B is the authorised user then the authority responds request of B by sending the message. The message is encrypted with authority's private key. It contains the public key of A with original request message sent by B.
- Step 6** Now user B sends reply message to the request of user A. The message is encrypted using A's public key. The reply message contains new random number in addition to the random number received from user A.
- Step 7** User A confirms the request by replying back. This time the message contains the random number sent by user B and the message is encrypted by B's public key.

After completion of above seven steps, authentications of both the users completed and they agree for communication. This complete process is illustrated in Figure 8.3. They communicate with each other by sending the message encrypted with each other's public key. For decryption, both the users use their own private key. For better security, each user should change their public key from the directory. This helps to avoid the reuse of public key.

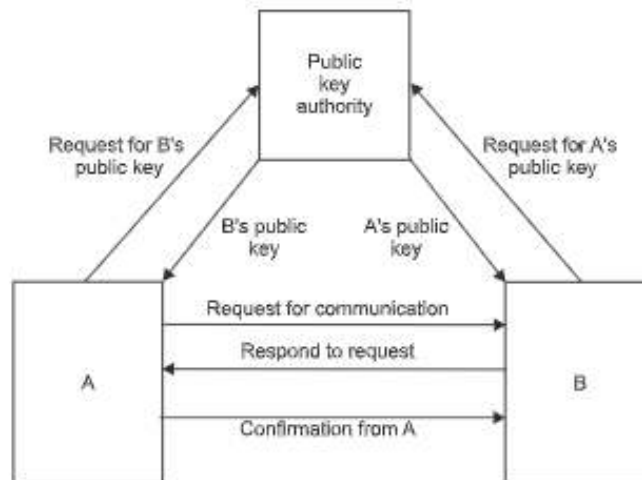


Figure 8.3 Key distribution by public key authority.

In this method, every time the users have to make a request for public key to the authority which makes the system slow.

8.2.4 Public Key Certificates

To avoid the drawback of above public key authority method, an alternative approach is used. In this approach, the certificates are used. This approach is called

public key certificates approach. It was suggested by Kohnfelder for the first time. In this approach, the public keys are exchanged by using certificates. This avoids the need to contact the public key authority for public key. Each certificate contains some other information in addition to public key. This other information includes time of the request and network address of the user who made the request. This time in the certificate helps to differentiate one request from other. Suppose user A's private key is captured by an attacker. Therefore, user A generates a new pair of public and private keys. Then he requested for the certificate by sending his public and private key pair. Meanwhile, the attacker uses the old certificate to communicate with user B. If user B have the old public key of A and encrypt the message using this old public key, the attacker can decrypt the message and read the message.

The information in the certificate is encrypted by the authority's private key. So, only the registered users who know the authority's public key can decrypt the message. When the user A sends the request for communication to user B with his certificate, user B first verifies the certificate of user A. The communication between the two users takes place as follows:

- Step 1** User A requests the certificate authority for a certificate. For this he sends his name and his public key.
- Step 2** The certificate authority responds to the request by sending encrypted certificate for user A. The certificate is encrypted using authority's private key. Each certificate contains some other information in addition to public key. This other information includes time of the request and network address of the user who made the request.
- Step 3** User A uses this certificate for communication with user B. Then user B decrypts the certificate using authority's public key and verifies the user A authentication. Also, user B stores the public key of A for further communication.
- Step 4** User B requests the certificate authority for a certificate. For this he sends his name and his public key.
- Step 5** The certificate authority sends the certificate to user B.
- Step 6** User B sends the certificate to user A. User A verifies the information by decrypting the certificates by authority's public key and stores the public key of user B.

After exchanging the certificates, user A and B start communication. They use each other's public for communication. The detail process is as shown in Figure 8.4.

8.3 DIFFIE-HELLMAN KEY EXCHANGE

Whether we use symmetric encryption or public key encryption technique, transfer of key among the users is the main issue. Key exchange in public key cryptography is simple as compared to symmetric encryption. In Section 8.2 we discuss different approaches of key transfer for public key cryptography. But for symmetric encryption, the key may be handled personally using different media. Symmetric encryption techniques are faster than public key cryptography.

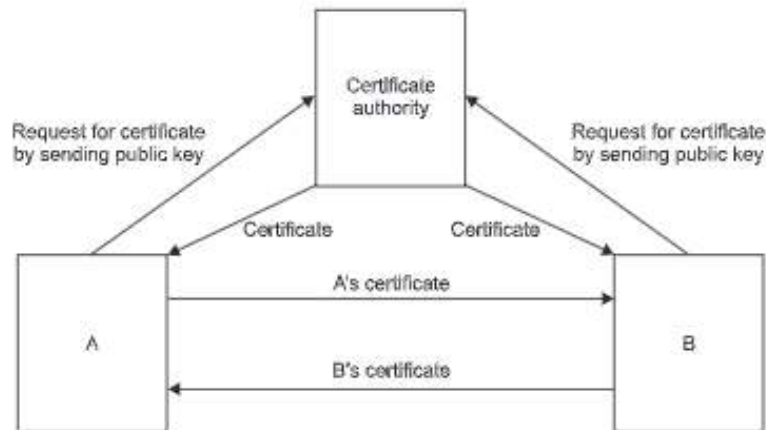


Figure 8.4 Public key certificates.

Diffie–Hellman designed an algorithm which helps for key transmission. This algorithm is known as *Diffie Hellman key exchange algorithm*. The name is derived from the two scientists Whitefield Diffie and Martin Hellman who developed this algorithm. They developed a protocol called *key agreement protocol* in 1976. In this, the private key is exchangeable among the users using public key techniques. This algorithm can be used for key exchange and not for encryption or decryption. Using exchangeable keys, both the parties generate a key called *shared secret key* which can be used for encryption and decryption. So, actual shared secret key is never transmitted through the insecure channel. The performance of this algorithm is better than any other algorithm.

8.3.1 Description

Suppose A and B are the two users who wish to communicate with each other. They decided to use Diffie–Hellman key exchange algorithm for sharing their keys. The working of Diffie–Hellman algorithm is illustrated from the following steps.

- Step 1** First, user A and user B both agree on two numbers, n and g . Where n is a large prime number and g is a primitive root of n .
- Step 2** User A selects X_A as his private key randomly, such that $X_A < n$. X_A is a large positive integer number.
- Step 3** Similarly, user B selects X_B as his private key randomly, such that $X_B < n$. X_B is a large positive integer number.
- Step 4** User A computes his public key, Y_A , using $Y_A = (g^{X_A}) \bmod n$.
- Step 5** Similarly, user B computes his public key, Y_B , using $Y_B = (g^{X_B}) \bmod n$.
- Step 6** Both the users now exchange their public keys over the insecure channel.
- Step 7** User A computes the key, k , called *shared secret key*, using the formula $k = (Y_B^{X_A}) \bmod n$.

Step 8 User B computes the key, k , called shared secret key, using the formula, $k = (Y_A^{X_B}) \bmod n$.

Step 9 Now, user A and user B communicate with each other using one of the symmetric encryption techniques. They use the shared secret key, k , as the encryption key for the selected algorithm. This key k was never transmitted over the insecure channel.

The Diffie–Hellman key exchange algorithm uses prime number n as modulo and primitive root of n . Table 8.1 illustrates the working of Diffie–Hellman algorithm.

Table 8.1 Working Of Diffie–Hellman algorithm

User A		User B	
Private key	Calculation	Private key	Calculation
X_A	n, g $Y_A = g^{X_A} \bmod n \rightarrow$ <i>A's public key</i> $(Y_B^{X_A}) \bmod n$ Shared key	X_B	n, g $\leftarrow Y_B = g^{X_B} \bmod n$ <i>B's public key</i> $(Y_A^{X_B}) \bmod n$ Shared key

User A and user B select n and g such that $n = 19$ and $g = 7$.

User A select his private key $X_A = 8$, then computes his public key $Y_A = g^{X_A} \bmod n$

$$Y_A = 7^8 \bmod 19 = 11$$

Therefore, A having private key $X_A = 8$ and public key $Y_A = 11$.

A sends his public key $Y_A = 11$ to B.

User B selects his private key $X_B = 10$, then computes his public key $Y_B = g^{X_B} \bmod n$

$$7^{10} \bmod 19 = 7$$

Therefore, B having private key $X_B = 10$ and public key $Y_B = 7$.

B sends his public key $Y_B = 7$ to A

A computes shared secret key using: $(Y_B^{X_A}) \bmod n$

$$(7^8 \bmod 19) = 11$$

B computes shared secret key using: $(Y_A^{X_B}) \bmod n$

$$11^{10} \bmod 19 = 11$$

Both users A and B have the same value of shared secret key. Note that only private keys, i.e., X_A, X_B and shared secret keys $Y_B^{X_A} = Y_A^{X_B}$ should keep secret. All the other values, i.e., g, n, Y_A and Y_B need not be secret. The shared secret key can be used for encryption. This key is known only to A and B. Larger values of X_A, X_B , and n provide more security and make difficult for cryptanalysis. If these values are small, it is possible for the cryptanalyst to find out shared secret key easily. In the above example, we know that the values of $n = 19, g = 7, Y_A = 11$ and $Y_B = 7$ are publically available. Therefore, one can try $g^{X_A X_B} \bmod 19$. There are only 18 values and this equation is possible and easily determined the shared secret key. Therefore, the values of n and g should be large.

Considering above problem, Table 8.2 will illustrate how the cryptanalyst tries to find the shared secret key.

Table 8.2 Shared Secret Key

User A		User B		Cryptanalyst	
Knows	Doesn't know	Knows	Doesn't know	Knows	Doesn't know
$n = 19$	$X_B = 10$	$n = 19$	$X_A = 8$	$n = 19$	$X_A = 8$
$g = 7$		$g = 7$		$g = 7$	$X_B = 10$
$X_A = 8$		$X_B = 10$			$k_s = 11$
$7^8 \text{ mod } 19 = 11$		$7^{10} \text{ mod } 19 = 7$		$7^{X_A} \text{ mod } 19 = 11$	
$7^{X_A} \text{ mod } 19 = 7$		$7^{X_B} \text{ mod } 19 = 11$		$7^{X_B} \text{ mod } 19 = 7$	
$7^8 \text{ mod } 19 = 11$		$11^{10} \text{ mod } 19 = 11$		$k_s = 7^{X_A} \text{ mod } 19$	
$11^{X_B} \text{ mod } 19 = 11$		$7^{X_A} \text{ mod } 19 = 11$		$k_s = 11^{X_B} \text{ mod } 19$	
$7^8 \text{ mod } 19 = 11^{X_B} \text{ mod } 19$		$11^{10} \text{ mod } 19 = 7^{X_B} \text{ mod } 19$		$7^{X_A} \text{ mod } 19 = 11^{X_B} \text{ mod } 19$	
$k_s = 11$		$k_s = 11$			

It should not be possible for user A or user B to compute each other's private key using all available information. For this, the values of n and g should be very large. If the values of n and g are small then it is possible for A and B to find out each other's private key. In this case, it is also possible for the cryptanalyst to find out fake shared secret key by guessing his private and public keys. Then he tries for A's and B's public key.

8.3.2 Security

Diffie-Hellman algorithm is secure for large values of n and g . For small values, it is possible for the attackers to find out the X_A and X_B using discrete logarithm. So, they can derive the shared secret key once they know the private key of the users. But for large values of n and g it is very difficult to obtain the private keys of the user. To calculate a large prime n , a Sophie Germain prime number m is used. It is called a *safe prime* as in that case the factors of n is only 2 and m . g is selected as the order of m instead of n so that it is difficult to compute X_A from g^{X_A} .

After computing the secret key, there is no need of X_A and X_B . So, the private keys of the users are destroyed once the secret key is obtained. This provides forward security. But man-in-the-middle attack is possible against Diffie-Hellman algorithm. We will discuss it in the next section.

8.3.3 Man-in-the-Middle Attack

Man-in-the-middle attack is also known as *bucket brigade attack*. Suppose the two users A and B are in communication with each other. They want to exchange their

public keys. Let Y_A and Y_B are their public keys. User A receives the public key Y_B indirectly. Suppose the public key of user B is computed by the attacker and send it to user A. There is no way for him to know whether Y_B sends by B or somebody else. User A calculates the shared secret key k_s from Y_B . Now, encrypt the message B uses the k_s and start communication. The attacker captures the encrypted message in between and able to decrypt it as instead of user B, the attacker's key is used by user A to compute the shared secret key.

Let us take an example. Suppose Ajay wants to communicate with Seema. Assume n and g are publicly known keys.

A writes a letter and send it through a courier as "Dear Seema, I would like to meet you in the garden tomorrow, 456, A. (Here 456 is treated as public key). Suppose there is Henry (an attacker) who works for the courier. Henry picks his own private key Z_H , and computes $Y_H = gZ_H \bmod n$, slightly modifies the letter by replacing the number 456 by 1234, prints the new version of the letter, and sends it to B. Later, B replies, "I will meet you tomorrow. My magic number is 14035, B." Henry again modifies the letter replaces 14035 by his own number and sends it again to A. Now, A computes the key using 456 and uses it for communication with A, and computes the key using 14035 and uses it for communication with B.

In this example, man-in-the-middle attack is established by Henry. He uses two distinct keys, one with "Seema" and the other with "Ajay", and then tries to masquerade as "Seema" to "Ajay" and/or vice-versa, perhaps by decrypting and re-encrypting messages passed between them.

8.3.4 Authentication

Authentication of the two users has not taken place in Diffie–Hellman algorithm. So, it is suffered by man-in-the-middle attack. To avoid this attack some mechanism should be used to authenticate the users to each other. For this, any authentication algorithm can be used with Diffie–Hellman algorithm. One of the solutions is to use digital signature during transmitting the public key.

8.4 ELLIPTIC CURVE ARITHMETIC

Victor Millar first time proposed the elliptic curve cryptography (ECC) in 1985. Also Neal Koblitz proposed the Elliptic curve cryptography in the same year. ECC is standardised in 1990 and used for commercial purpose. It is more secure even for a small key. Due to small key size, the performance of Elliptic curve cryptography is better as compared to other algorithms. There is no sub-exponential algorithm due to which the cryptanalyst can break it. For selecting a small prime numbers or smaller finite fields, greater degree of security can be achieved. This saves hardware implementations for the algorithm. Elliptic curve cryptography uses discrete logarithms which provide more challenging task for finding keys of equivalent length. Today ECC is used in ad-hoc wireless networks and mobile networks.

Elliptic curve groups are formed by using elliptic curves. A group is a set of similar elements with some user-defined arithmetic operations on these elements. For elliptic

curve groups, these user-defined operations are defined geometrically. The underlying fields can be created by the number of points on a curve.

8.4.1 Elliptic Curve Groups Over Real Numbers

Over a hundred people studied elliptic curves. An elliptic curve E over the real numbers is the set of points (x, y) . It is a graph of an equation of the form:

$$y^2 = x^3 + ax + b$$

where x, y, a and b are real numbers. It also includes a point at infinity.

Each choice of the numbers a and b yield a different elliptic curve. For example, $a = -3$ and $b = 3$ give the elliptic curve with equation $y^2 = x^3 - 3x + 3$; the graph of this curve is shown in Figure 8.5.

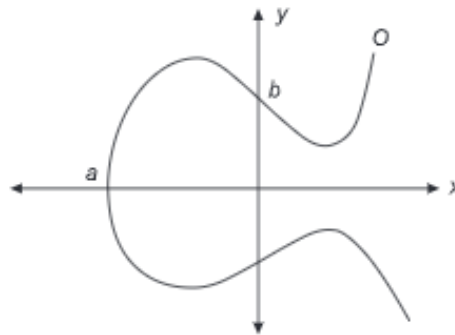


Figure 8.5 Elliptic curve.

If the given equation for elliptic curve has no repeated factors, then the given equation of elliptic curve can be used to form a group. The corresponding points on a curve form a group over real numbers with a special point O . This point O is called the *point at infinity*.

8.4.2 Elliptic Curve Addition: A Geometric Approach

The basic function of elliptic curve groups is addition, so it is additive groups. The addition of any two points on the elliptic curve can be defined geometrically.

The negative of any point $P(x_p, y_p)$ lies on the elliptic curve is $-P(x_p, -y_p \text{ mod } P)$. If any point P lies on the elliptic curve then point $-P$ also lies on the curve.

Adding Distinct Points P and Q

Suppose $P(x_p, y_p)$ and $Q(x_q, y_q)$ are two distinct points on the elliptic curve such that Q is not $-P$. The point where line PQ intersects the curve is $-R$ and its reflection against x -axis is R (Figure 8.6). Then

$$P + Q = R$$

where R is the point where line PQ intersects the curve.

$$m = (y_P - y_Q)/(x_P - x_Q) \pmod{P}$$

$$x_R = m^2 - (x_P + x_Q) \pmod{P} \text{ and } y_R = -y_P + m(x_P - x_R) \pmod{P}$$

where m is the slope of the line PQ .

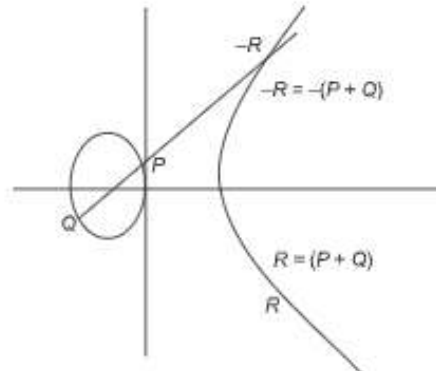


Figure 8.6 Adding distinct points P and Q .

Adding the Points R and $-R$

If the two points R and $-R$ join by a vertical line, it does not intersect the elliptic curve at any point other than R and $-R$. Therefore, we cannot add R and $-R$ as P and Q . Due to this, the point at infinity O is added to the elliptic curve group. O is the additive identity of the elliptic curve group. All the elliptic curves have an additive identity.

By addition property,

$$R + (-R) = O.$$

Therefore, we get

$$R + O = R \text{ is in the elliptic curve group.}$$

Figure 8.7 illustrated this property.

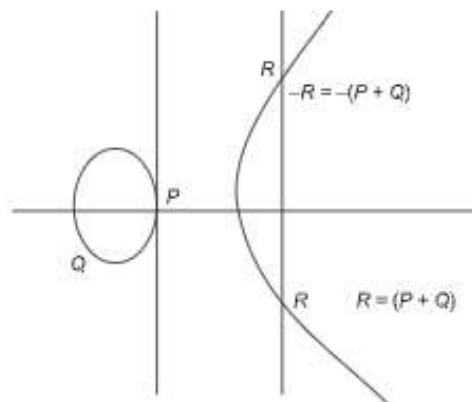


Figure 8.7 Adding the points R and $-R$.

Doubling the Point Q

Now, suppose we want to add a point Q in the group. Draw a tangent to the curve at point Q . If the y -coordinate of Q is not 0, then the tangent intersects the elliptic curve at exactly one other point. That point is $-R$. The reflection of $-R$ against x -axis is R . This is shown in Figure 8.8. This operation helps to double the point so it is called *doubling the point Q* .

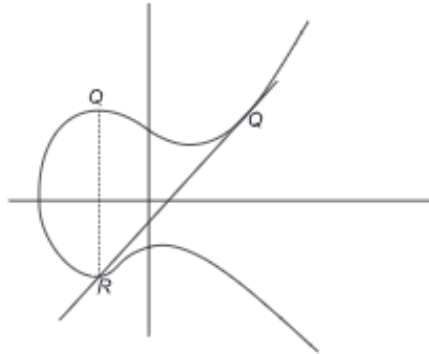


Figure 8.8 Doubling the point Q .

The law for doubling a point on an elliptic curve group is defined by:

- If y -coordinate y_Q is 0, the tangent from Q is always vertical.
- If $y_Q = 0$, then doubling the point Q .
- If $y_Q = 0$, then the tangent to the elliptic curve at Q is vertical and it does not intersect the elliptic curve at any other point as shown in Figure 8.9.

By definition, $2Q = 0$ for a given point Q .

If one wanted to find $3Q$ in this situation, one can add $2Q + Q$. This becomes $Q + 0 = Q$

Thus, $3Q = Q$.

$3Q = Q, 4Q = 0, 5Q = Q, 6Q = 0, 7Q = Q, 8Q = 0, 9Q = 0$, etc.

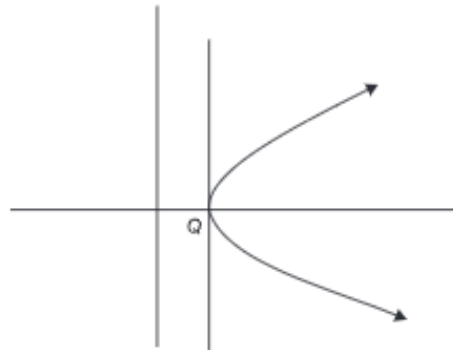


Figure 8.9 The tangent from Q is always vertical if $y_Q = 0$.

8.4.3 Elliptic Curve Addition: An Algebraic Approach

Above approach of elliptic curves provides an excellent method of illustrating elliptic curve arithmetic, but it is not a practical method for implementing arithmetic computations. So, there should be a method to construct algebraic formulae to efficiently compute the geometric arithmetic.

Adding Distinct Points P and Q

When two points on the elliptic curve, P and Q are not negative of each other, then

$$P + Q = R$$

where

$$m = (y_P - y_Q)/(x_P - x_Q)$$

$$x_R = m^2 - x_P - x_Q \text{ and } y_R = -y_P + m(x_P - x_R)$$

Note that m is the slope of line PQ .

Doubling the Point P

When y_P is not 0,

$$2P = R$$

where

$$m = (3x_P^2 + a)/(2y_P)$$

$$x_R = m^2 - 2x_P \text{ and } y_R = -y_P + m(x_P - x_R)$$

We know one of the parameters chosen with the elliptic curve is a and m is the slope of tangent on the point P .

8.4.4 Elliptic Curve Groups over F_P

Above approach use real numbers which make the execution of the algorithm very slow. At the same time rounding off the real number gives approximate results. Due to all these reasons, if we use this approach for cryptography, the performance of the cryptographic algorithms deteriorate. Cryptographic algorithms require fast and precise arithmetic. Thus, the finite fields of F_P and F_{2^m} are used in place of real number arithmetic. The field F_P uses the numbers from 0 to $P - 1$. The computations will result in an integer between 0 to $P - 1$.

For example, in F_{29} the field is composed of integers from 0 to 28, and any operation within this field will result an integer also in between 0 and 28.

An elliptic curve of F_P can be formed by selecting a and b as coefficients. The coefficients a and b are the integer numbers from 0 to $P - 1$, the field of F_P . The elliptic curve includes all points (x, y) which satisfy the elliptic curve equation modulo P (where x and y are numbers in F_P).

For example: if a and b are in F_P then $y^2 \text{ mod } p = (x^2 + ax + b) \text{ mod } P$ has an underlying field of F_P . The elliptic curve can be used to form a group if the term $x^3 + ax + b$ contains no repeating factors. An elliptic curve group over F_P consists of the points on the corresponding elliptic curve, together with a special point O called the point at infinity. There are finitely many points on an elliptic curve.

Example of an Elliptic Curve Group over F_p

Suppose, an elliptic curve over the field F_{13} . With $a = 1$ and $b = 0$, the elliptic curve equation is $y^2 = x^3 + x$. The point $(3, 11)$ satisfies this equation since:

$$\begin{aligned}y^2 \bmod P &= x^3 + x \bmod P \\121 \bmod 13 &= 27 + 3 \bmod 13 \\4 \bmod 13 &= 30 \bmod 13 \\4 &= 4\end{aligned}$$

Here $P = 13$, therefore, there are 13 points which satisfy the given equation. These points are:

$(0, 0), (2, 3), (2, 10), (3, 2), (3, 11), (6, 1), (6, 12), (7, 5), (7, 8), (9, 6), (9, 7), (11, 4), (11, 9)$

If we observe the above points, for every value of x , there are two points. The graph is symmetric about $y = 6.5$. Over the field of F_{13} , the negative components in the y -values are taken modulo 13, resulting in a positive number as a difference from 13. Here $-P = (x_p, (-y_p \bmod 13))$.

8.4.5 Arithmetic in an Elliptic Curve Group over F_p

Elliptic curve groups over F_p and over real numbers have following difference:

1. There are finite numbers of points in elliptic curve groups over F_p . As some of the points are discrete, there is a problem of connecting these points to get a smooth curve.
2. It is difficult to apply geometric relationships. As a result, the geometry used in one group cannot be used for other groups. But, the algebraic rules of one group can be applied for other groups.
3. Due to use of real number, there is round off error in elliptic curves over real numbers. In the field of F_p there is no round-off error.

Adding Distinct Points P and Q

The negative of the point P is $-P$ where $x_p = x_p$ and $y_p = -y_p \bmod p$. If P and Q are distinct points such that P is not $-Q$, then

$$P + Q = R$$

where

$$\begin{aligned}m &= (y_p - y_q)/(x_p - x_q) \bmod P \\x_R &= m^2 - x_p - x_q \bmod p \text{ and } y_R = -y_p + m(x_p - x_R) \bmod p\end{aligned}$$

Note that m is the slope of the line through P and Q .

Doubling the Point P

Suppose y_p is not 0.

$$2P = R$$

where

$$\begin{aligned}m &= (3x_p^2 + a)/(2y_p) \bmod P \\x_R &= m^2 - 2x_p \bmod p \text{ and } y_R = -y_p + m(x_p - x_R) \bmod P\end{aligned}$$

a is the parameter selected with the elliptic curve and m is the slope of the line PQ .

8.4.6 Elliptic Curve Groups over F_{2^n}

The rules for arithmetic in F_{2^n} can be defined by two ways:

1. Polynomial representation
2. Optimal normal basis representation.

With F_{2^n} , an elliptic curve is formed by selecting a and b within F_{2^n} (if $b \neq 0$). The elliptic curve equation for F_{2^n} having a characteristic 2 is:

$$y^2 + xy = x^3 + ax^2 + b$$

Elliptic curve equation over F_{2^n} satisfies for all points (x, y) . These points together with a point at infinity form the elliptic curve. On an elliptic curve, there are finitely many points. As these points are bits, addition is controlled by using XOR operation.

An Example of an Elliptic Curve Group over F_{2^n}

The field F_{2^4} , defined by $f(x) = x^4 + x + 1$.

The element $g = (0010)$ is a primitive root for the field.

The powers of g are:

$$\begin{aligned} g^0 &= (0001) & g^1 &= (0010) & g^2 &= (0100) & g^3 &= (1000) & g^4 &= (0011) & g^5 &= (0110) \\ g^6 &= (1100) & g^7 &= (1011) & g^8 &= (0101) & g^9 &= (1010) & g^{10} &= (0111) & g^{11} &= (1110) \\ g^{12} &= (1111) & g^{13} &= (1101) & g^{14} &= (1001) & g^{15} &= (0001) \end{aligned}$$

The large value of n generates the more efficient table which provides more security. For adequate security, $n = 160$. The pattern allows the use of primitive root notation (g^x) rather than bit string notation, as used in the following example.

$$f(x) = x^4 + x + 1.$$

Suppose the elliptic curve $y^2 + xy = x^3 + g^4x^2 + 1$.

Here $a = g^4$ and $b = g^0 = 1$. The point (g^3, g^3) satisfies this equation over F_{2^n} :

$$\begin{aligned} y^2 + xy &= x^3 + g^4x^2 + 1 \\ (g^3)^2 + g^3g^3 &= (g^3)^3 + g^4g^{10} + 1 \\ g^6 + g^6 &= g^{18} + g^{14} + 1 \end{aligned}$$

$$(1100) + (0101) = (0001) + (1001) + (0001)$$

$$(1001) = (1001)$$

The fifteen points which satisfy this equation are:

$$\begin{aligned} (1, g^{13}) & (g^3, g^{13}) & (g^5, g^{11}) & (g^6, g^{14}) & (g^9, g^{13}) & (g^{10}, g^8) & (g^{12}, g^{12}) \\ (1, g^6) & (g^3, g^8) & (g^5, g^3) & (g^6, g^8) & (g^9, g^{10}) & (g^{10}, g) & (g^{12}, 0) & (0, 1) \end{aligned}$$

8.4.7 Arithmetic in an Elliptic Curve Group over F_{2^m}

There is finite number of points for an elliptic curve group over F_{2^n} without round off error. Use of binary arithmetic makes the method very efficient.

The following algebraic rules are applied for arithmetic over F_{2^n} .

Adding Distinct Points P and Q

We know that the negative of $P = -P$. If P and Q are different points then, $P \neq Q$.

Then, $P + Q = R$
where

$$m = (y_P - y_Q)/(x_P + x_Q)$$

$$x_R = m^2 + m + x_P + x_Q + a \text{ and } y_R = m(x_P + x_Q) + x_R + y_P$$

We know that, $P + (-P) = 0$, the point at infinity and $P + 0 = P$ for all points P in the elliptic curve group.

Doubling the Point P

If $x_P = 0$, then $2P = 0$

Provided that x_P is not 0, $2P = R$

where

$$m = x_P + y_P/x_P$$

$$x_R = m^2 + m + a \text{ and } y_R = x_P^2 + (m + 1) * x_R$$

a is the parameter selected with the elliptic curve and m is the slope of the line through P and Q .

8.5 ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

ECC is an alternative method for public key cryptography digital signature algorithm and RSA. ECC provides same level of security using small prime numbers than conventional encryption systems. Due to the use of small key, ECC is faster than conventional encryption system. This improves the performance of ECC and makes it efficient for public key cryptography. ECC is useful for applications which have small hardware capacity such as smart cards, cellular phones, etc. ECC is now used for implementing digital signatures and key distribution protocols.

8.5.1 Elliptic Curve Diffie-Hellman

We discuss Diffie-Hellman key exchange algorithm in Section 8.3. Elliptic Curve Diffie-Hellman (ECDH) is same as algorithm using elliptic key cryptography. Key exchange in elliptic curve can be done as: Select an elliptic curve over a finite field in the form 2^n . Select four numbers p , q , a and b . Select $g = (x_1, y_1)$.

8.5.2 Key Establishment Protocol

Suppose users A and B want to communicate with each other. Initially they agree with basic parameters. Both the users have their own private-public key pair suitable for ECC. Suppose user A has X_A and Y_A as his private key and public key respectively.

User B also has X_B , and Y_B as his private key and public key respectively. User A and user B both share each other's public keys.

User A computes the point P having coordinates $(X_k, Y_k) = d_A Q_B$. Also user B computes $(X_k, Y_k) = d_B Q_A$. The shared secret key is the x -coordinate value of point P , i.e., X_k . Here, $d_A Q_B = d_A d_B G$, $d_B Q_A = d_B d_A G$ and $d_A d_B G = d_B d_A G$, therefore, $d_A Q_B = d_B Q_A$. Therefore, the number calculated by both the users is equal. Only the public key is transferred through insecure channel. Other information is not transferred, so this algorithm is secure.

The public keys of both the user's are either static (and trusted, say via a certificate) or ephemeral. *Static key* means key is authenticated via certificate. It provides neither forward secrecy nor key compromise impersonation resilience, among other security properties. *Ephemeral keys* are not necessarily authenticated. If authentication of key is required, it should be obtained by other means.

8.6 ELLIPTIC CURVE SECURITY AND EFFICIENCY

The security of different public key cryptographic algorithm depends upon the size of key used. Most of the algorithms like Diffie–Hellman and RSA use 1024-bit key. One can select the large key size for these algorithms to provide more security as the larger key size is difficult for the cryptanalysis of the algorithm. But this deteriorates the performance of the algorithms. To increase the performance and at the same time provide the required security, one should take advantage of the research taken place in the area of public key particularly in ECC.

Selection of key size for public key cryptosystem can be determined from the strength of symmetric encryption algorithms those using public key algorithms for transfer the key. There are many symmetric encryption algorithms such as data encryption standard (DES) and advanced encryption standard (AES), international data encryption algorithm (IDEA), and Blowfish. For all these symmetric encryption algorithms, the key size is the main parameter of security. The cryptanalysis of these algorithms with key size of n -bit, it will require approximately $2n - 1$ operations.

Table 8.3 gives the key sizes recommended by the national institute of standards and technology (NIST) for symmetric encryption algorithms and other approaches which provide equivalent security.

Table 8.3 Key sizes

Symmetric encryption key size (bits)	RSA and Diffie–Hellman key size (bits)	Elliptic curve cryptography key size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	334
256	15360	521

For the 128-bit AES, the equivalent key size for RSA or Diffie–Hellman is 3072-bit whereas for elliptic curves, it is only 256-bit. As compared to RSA and Diffie–Hellman,

the key size for elliptic curve increases slowly as shown in Table 8.3. Hence, elliptic curve systems offer more security per bit increase in key size than either RSA or Diffie–Hellman algorithms.

Not only security, ECC is more attractive due to its computational efficiency than other algorithms such as RSA and Diffie–Hellman. ECC uses arithmetic which takes more computational time per bit as compared to RSA and Diffie–Hellman algorithm. But the security provided per bit by ECC is more than the extra time required for computation. Table 8.4 shows the ratio of computation of Diffie–Hellman to elliptic curve for different key sizes (in bits) listed in Table 8.3.

Table 8.4 Relative computation costs of Diffie–Hellman and elliptic curves

Security level (bits)	Ratio of DH cost: EC cost
80	3:1
112	6:1
128	10:1
192	32:1
256	64:1

If we use large key size for transferring the keys, the channel overhead is increased. So, ECC provides better solution as compared to RSA and Diffie–Hellman algorithms. There are number of elliptic curves standardised by NIST. Out of these, ten curves are for binary fields and five are for prime fields.

8.7 ZERO-KNOWLEDGE PROOF

A disadvantage of the above encryption algorithms is that when user A gives his secret key to user B, user B can thereafter impersonate user A. But, zero-knowledge (ZK) protocols allow user A to demonstrate knowledge of a secret key to user B without revealing any useful information about that secret key. Zero-knowledge proofs are probabilistic and based on interactive method. The example of zero-knowledge proof is RSA algorithm in which the user can prove that he knows the secret associated with his public key without revealing his private key. A protocol between two users in which one user is called prover and the other user is called the verifier. Prover tries to prove a certain fact to the verifier. This protocol is called *zero-knowledge protocol* or *zero-knowledge proof*. In cryptography, it is used for authentication.

Properties of zero-knowledge proof:

1. *Completeness*: If the fact is true, the honest verifier always accepts this fact and both the users follow the protocol.
2. *Soundness*: If the fact is false, the honest verifier always rejects this fact except with some small probability.
3. *Zero-knowledge*: If the fact is true, no cheating verifier learns anything other than this fact. This is formalised by showing that every cheating verifier has

some simulator that, given only the fact to be proven (and no access to the prover), can produce a transcript that “looks like” an interaction between the honest prover and the cheating verifier.

The first two properties are the properties of interactive systems and the third property is what makes the zero-knowledge proof.

Research in zero-knowledge proofs has been motivated due to its usefulness for authentication. In authentication, one user wants to prove his identity to a second user through some secret information. But at the same time user A does not want to learn the second user to learn anything about this secret. This is called a *zero-knowledge proof of knowledge*. However, this secret information is typically too small or insufficiently random to be used in many schemes for zero-knowledge proofs of knowledge.

Considering in mathematical sense, zero-knowledge proofs are not proofs because there is some small probability of the error called the *soundness error*. However, one can reduce this error to negligibly small values.

8.7.1 Cave Story

There is a well-known story presenting some of the ideas of zero-knowledge proofs, first published by Joan-Jacques Quisquater et al. in their paper “How to Explain Zero-Knowledge Protocols to Your Children”. There are two users in a zero-knowledge proof, one is the prover of the fact and another is the verifier of the fact. Here we can call them user P and user V.

In this story, there are two users, P (she) and V (he). There is a cave which has a magic door which is open with a secret word. P claims that she knows the secret word to open a magic door of a cave. The cave is circular and there is an entrance in one side only. The other side is blocked by the magic door. User V wants the secret word and ready to pay for it. But he will pay to user P if he is sure that user P really knows the secret key. User P is also ready to tell the secret word to user V but after she will get the money. So, they plan a scheme so that P can prove that she knows the secret word without telling the secret word to V.

First, user V waits outside the cave and user P goes inside the cave. For convenience, we label the left path from the entrance as A and the right path from the entrance as B. User P randomly selects either path A or path B. Then, user V enters inside the cave and shouts the name of the path he wants user P to use to return, either A or B, chosen at random. Providing she really does know the magic word, this is easy: she opens the door, if necessary, and returns along the desired path. Note that V does not know which path P has gone down.

However, suppose P does not know the secret word. Then, she can only return by the named path if V gives the name of the same path that P entered by. Since V chooses A or B at random, P has at most a 50% chance of guessing correctly. If they repeat this trick many times, say 20 times in a row, P’s chance of successfully anticipating all of V’s requests becomes vanishingly small, and V should be convinced that P knows the secret word.

There is a question, why not just make P takes a known path that will force her through the door, and make V waits at the entrance. Certainly, that will prove

that P knows the secret word, but it also allows the possibility of eavesdropping. By randomising the initial path that P takes and preventing V from knowing it, it reduces the chances that V can follow P and learn not just that P knows the secret word, but what the secret word actually is. This part of the exchange is important for keeping the amount of information revealed to a minimum. This is explained in Figure 8.10.

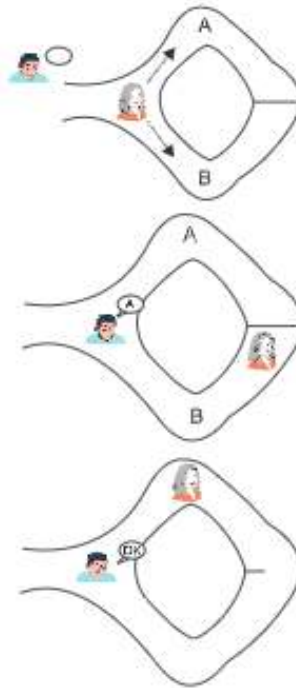


Figure 8.10 Cave story.

These ideas can be extended to a more realistic cryptography application. In this, P knows a Hamiltonian cycle for a large graph, G . She will prove that she knows this information without revealing the cycle itself. A Hamiltonian cycle in a graph is just one way to implement a zero-knowledge proof, in fact any NP-complete problem can be used. However, P does not want to simply reveal the Hamiltonian cycle or any other information to V; she wishes to keep the cycle secret.

To show that P knows this Hamiltonian cycle, she and V play several rounds of a game. At the beginning of each round, P creates H , an isomorphic graph to G . Since it is trivial to translate a Hamiltonian cycle between isomorphic graphs with known isomorphism, if P knows a Hamiltonian cycle for H , she also must know for graph G . For verification V can either ask P to show the isomorphism between H and G , or show a Hamiltonian cycle in H .

If P is asked to show that the two graphs are isomorphic, she provides the vertex translations that map G to H . V can verify that they are indeed isomorphic. If P is asked to prove that she knows a Hamiltonian cycle in H , she translates her

Hamiltonian cycle in G onto H and reveals it to V . V can then check the validity of the cycle. During each round, P does not know which question she will be asked by V until after giving H . Therefore, in order to be able to answer both, H must be isomorphic to G and P must have a Hamiltonian cycle in H . Because only someone who knows a Hamiltonian cycle in G would always be able to answer both questions, V becomes convinced that P does know this information. This takes after a sufficient number of rounds.

However, P 's answers do not reveal the original Hamiltonian cycle in G . Each round, V will learn only H 's isomorphism to G or a Hamiltonian cycle in H . He would need both answers for a single H to discover the cycle in G , so the information remains unknown as long as P can generate a unique H every round. Because of the nature of the isomorphic graph and Hamiltonian cycle problems, V gains no information about the Hamiltonian cycle in G from the information revealed in each round.

If P does not know the information, she can guess which question V will ask and generate either a graph isomorphic to G or a Hamiltonian cycle for an unrelated graph, but since she does not know a Hamiltonian cycle for G , she cannot do both. With this guesswork, her chance of fooling V is $n/2$ where n is the number of rounds. For all realistic purposes, it is infeasibly difficult to defeat a zero-knowledge proof with a reasonable number of rounds in this way.

Attacks

To break zero-knowledge protocol, following attacks are tried against it.

1. *Impersonation*: One entity pretends to be another entity
2. *Replay*: Capture the information from a single previous protocol and use on the same or different verifier
3. *Interleaving*: A selective combination of information from one or more previous protocol
4. *Reflection*: Sending information from an ongoing protocol execution back to the originator
5. *Forced delay*: An adversary that intercepts a message and relays it later
6. *Chosen-text*: When an adversary chooses specific challenges in an attempt to gain information about the secret

SOLVED PROBLEMS

- 8.1 Users A and B use the Diffie–Hellman key exchange technique. They agree with a common prime $n = 41$ and a primitive root $g = 13$.
- (a) If user A has private key $X_A = 27$, what is A 's public key Y_A ?
 - (b) If user B has private key $X_B = 18$, what is B 's public key Y_B ?
 - (c) What is the shared secret key?

Solution

User A		User B	
Private key	Calculation	Private key	Calculation
$X_A = 27$	For $n = 41$ and $g = 13$ $Y_A = g^{X_A} \bmod n$ $= 13^{27} \bmod 41$ $= 15$ $k_S = (Y_B)^{X_A} \bmod n$ $= (8)^{27} \bmod 41$ $= 2$	$X_B = 18$	For $n = 41$ and $g = 13$ $Y_B = g^{X_B} \bmod n$ $= 13^{18} \bmod 41$ $= 8$ $k_S = (Y_A)^{X_B} \bmod n$ $= (15)^{18} \bmod 41$ $= 2$

Therefore

- (a) A's public key $Y_A = 15$
 (b) B's public key $Y_B = 8$
 (c) The shared secret key $k_S = 2$
- 8.2 Users A and B use the Diffie–Hellman key exchange technique. They agree with a common prime $n = 67$ and a primitive root $g = 5$.
- (a) If user A has private key $X_A = 10$, what is A's public key Y_A ?
 (b) If user B has private key $X_B = 24$, what is B's public key Y_B ?
 (c) What is the shared secret key?

Solution

User A		User B	
Private key	Calculation	Private key	Calculation
$X_A = 10$	For $n = 67$ and $g = 5$ $Y_A = g^{X_A} \bmod n$ $= 5^{10} \bmod 67$ $= 40$ $K = (Y_B)^{X_A} \bmod n$ $= (25)^{10} \bmod 67$ $= 59$	$X_B = 24$	For $n = 67$ and $g = 5$ $Y_B = g^{X_B} \bmod n$ $= 5^{24} \bmod 67$ $= 25$ $K = (Y_A)^{X_B} \bmod n$ $= (40)^{24} \bmod 67$ $= 59$

Therefore,

- (a) A's public key $Y_A = 40$
 (b) B's public key $Y_B = 25$
 (c) The shared secret key = 59
- 8.3 We use the Diffie–Hellman key exchange with private keys X_A and X_B and public keys $Y_A = g^{X_A} \bmod n$ and $Y_B = g^{X_B} \bmod n$. We assume $n = 71$ and $g = 7$.
- (a) Give two possible pairs (X_A, X_B) such that the common key $k = 1$.
 (b) An attacker knows that the product $Y_A * Y_B = 7 \bmod g$.
 Give two possible pairs (X_A, X_B) that satisfy the attacker's knowledge.

Solution

(a) $k = (Y_B X_A) \bmod n$

$$1 = 7 X_B X_A \bmod 71 \quad (i)$$

Using Fermat's Little theorem $a^n = 1 \bmod n$

$$\begin{aligned} a^{n-1} \bmod n &= 1 \\ 7^{(71-1)} \bmod 71 &= 1 \end{aligned} \quad (ii)$$

From equation (i) and (ii)

$$7 X_B X_A \bmod 71 = 7^{(71-1)} \bmod 71$$

Therefore, $X_A Y_A = 70$.Therefore, the possible values of X_A and Y_A are 10 and 7 or 14 and 5

(b) $Y_A * Y_B = 7 \bmod n$

$$Y_A * Y_B = g^{X_A} \bmod n * g^{X_B} \bmod n = 7 \bmod 71$$

$$7^{X_A} * 7^{X_B} \bmod 71 = 7 \bmod 71$$

$$7^{X_A + X_B} \bmod 71 = 7 \bmod 71$$

Using Fermat's Little theorem $a^n = a \bmod n$

$$X_A + X_B = 71$$

We know that $Y_A * Y_B = 78 \bmod 71 = 7 \bmod 71$.Factorise 78, we get (2, 39), (3, 26) and (6, 13) are the possible values of Y_A and Y_B .

$$6 = 7^{X_A} \bmod 71 \text{ and } 13 = 7^{X_B} \bmod 71$$

Solving we get $X_A = 39$ and $X_B = 32$

$$3 = 7^{X_A} \bmod 71 \text{ and } 26 = 7^{X_B} \bmod 71$$

We get, $X_A = 26$ and $X_B = 45$ **SUMMARY**

For public key cryptography, two important issues are: the distribution of public keys and the use of public key encryption for distribution of secret keys. In this chapter, we discuss different approaches for public key distribution. These include: the public announcement, publicly available directory, public key authority, and public key certificates.

Diffie-Hellman key exchange algorithm is used by two parties to generate a shared secret key. In Diffie-Hellman algorithm, there is no need of transferring the shared secret key for encryption. But it is suffered by man-in-the-middle attack. The Diffie-Hellman algorithm by itself does not provide authentication of the users. Elliptic curve cryptography is more secure even for a small key. Due to small key size, the performance of elliptic curve cryptography is better as compared to other algorithms. There is no sub-exponential algorithm due to which the cryptanalyst can break it. For selecting a small prime numbers or smaller finite fields, greater degree of security can be achieved.

A disadvantage of Diffie-Hellman or RSA algorithms is that when user A gives his secret key to user B, user B can thereafter impersonate user A. But, zero-knowledge (ZK) protocols

CHAPTER 9

Authentication

9.1 INTRODUCTION

In the last chapter, we discuss about zero-knowledge proof which is useful for authentication of the users against each other. Authentication is one of the key aspects of cryptography. It can be used to guarantee that communication end-points, i.e., sender and receiver are the parties who they claim. It is the process of establishing or confirming a proof of identities, i.e., the claims made by or about the things are true. In cryptography and network security, authentication is done by verifying digital information of the sender or the recipient. Traditional method of authentication is user id and passwords. If someone knows the password, then it is assumed that the he/she is the authentic person.

For example, in some application, for the first time the user first registered by using an assigned user id and password or he can declare his own user id and password. For the subsequent use of that application, he must use that previously used user id and password. But the password is not the secure mechanism for authentication as it may be forgotten, or stolen or accidentally revealed. It is possible that while using the password, Eve may capture it. Nowadays, alternative techniques are used for password authentication. These techniques use encryption of the password, so that if it is stolen, it is difficult to decrypt and capture the password. Alternative to password authentication, new techniques are emerged which include: token-based authentication, biometric-based authentication and certificate-based authentication. Authorisation of any application needs the authentication of the user. Authorisation gives the full access to a specific application. To provide the security, the prerequisite to authorisation is authenticated. It depends on the identification of the user.

9.1.1 OBJECTIVES

The main objectives of authentication requirement are to:

- ensure that the claimants are really what they claim to be,
- avoid compromising security to an imposter.

9.1.2 Measurements

The degree of authentication which is required is dependent on the security expected for the application. The requirements of authentication are typically specified using following parameters:

- Minimum percentage of valid identities [by role, group] authenticated.
- Maximum percentage of invalid identities [by role, group] authenticated.
- Average time required for an attacker to become authenticated. It may be manually or using computer.

Authentication and authorisation are two different mechanisms. Many times people confused between them and treated both as the same. In many host-based systems, the same hardware is used for authentication as well as for authorisation. In some systems, same software is used for authentication as well as for authorisation. We can take a simple example, your computer has two login, one is admin and another is guest login. When the user wants to use your computer, he has to enter his password which is used for authentication. It is authenticated that whether he is guest user or admin user. If he is a guest user, he has limited access to the computer system. He cannot change the different setting in the system as he is not the authority to do it. But if he is admin, he can do it. Authorisation depends on the user type and not on the password.

So in authentication, the system may securely identify the users. Authentication systems provide following information about the user:

- Who is the user? i.e., identity of the user.
- Is the user really who he/she claims himself to be? i.e., authentication of the user.

An authentication system may be as simple or complicated system. Simple system includes systems which are using a plain-text password. It is the insecure authentication mechanism. Complicated system includes system like the Kerberos system where authentication process is complex. It is more secure authentication mechanism. Irrespective of type of system, i.e., whether it is a simple or complicated, authentication systems depend on some unique bit of information which is known only to the individual user who is being authenticated and the authentication system. Advanced authentication system use thumb impression, iris image or hash values derived from data related to users. For verification, the authenticating system typically challenges the user to provide his unique information. If the information provided by the user and the information stored with the authenticating system is matched, the user is considered an authenticated user of the system.

Whereas authorisation, is the mechanism through which a system determines what level of access to a particular authenticated user should have. It used to secure the resources controlled by the system. For example, in the above example, guest login allows the user to use the limited applications whereas admin user can use and change the setting of the system.

So, in authorisation of the system may control the access of the users to different applications of the system. Authorised systems provide following information about the user:

- Is user A authorised to access some particular resource in the system?
- Is user A authorised to perform some particular operation in the system?
- Is user A authorised to perform some particular operation on some particular resource?

In this way authentication and authorisation are somewhat related to each other. Authorisation depends on authentication systems for security. Figure 9.1 graphically illustrates the inter-relationship between authentication and authorised systems and a typical client/server application.

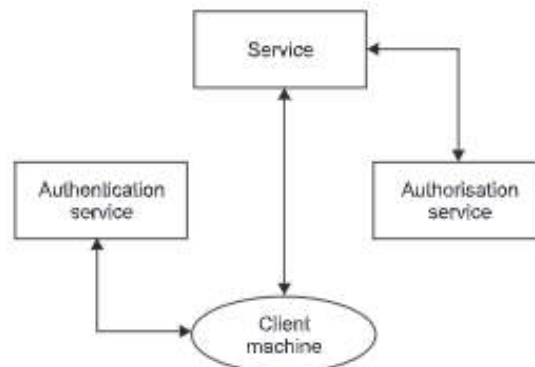


Figure 9.1 Authentication vs authorisation.

In Figure 9.1, a user working on a client system first login to the authentication system to prove his identity and request to work on the server. In turn, the server system contacts to an authorised system to know about the rights and privileges, the client's user have on the server.

9.2 AUTHENTICATION METHODS

There are different authentication methods available. But the simplest and the most common method of authentication is the traditional local authentication method.

9.2.1 Password-based Authentication Method

The password-based authentication method refers to a secret information which the user has to prove that he knows it. A password is a pattern of characters contains alphabets, numbers and special characters. It is used to verify the user who requests for accessing the computer system is actually that particular user. For a multiuser or securely protected single-user system, each user has a unique identity called *user id* which is publically known to all. For authentication of such users, additional identification is needed is called *password*. This password is secret and known only to

that user and the computer system. When the user wants to use the computer system, he uses his user id and password. Then the authentication system verifies the user id and password with the database available with the authentication system. Once the verification is completed and the information provided by the user matches with the information available in the database, authentication system allows the user to access the necessary applications. The major problem with this method is that password may be captured by the eavesdropper.

Guidelines for good password:

- Password should be a combination of alphabets, numbers and special characters.
- It should not be a word or information related to you such as your PAN number, date of birth, name of pet, etc.
- Do not select the word that can be found in the dictionary or be a keyword.
- Do not select the password which is related to current news.
- It should not be similar to previous password.
- It should be easy to remember.
- Do not write the password on a paper or share it with anybody.

It is recommended that user should change his password periodically. Due to this, if the eavesdropper knows the password, he cannot misuse it. To provide the security, user id and password are the basic and simplest method of authentication. It is an efficient and cost-effective method. The major problem with this system is that there are number of such systems where a same user has his different user id and password. For example, nowadays many users have different passwords for online banking, e-commerce application, different e-mail account, etc. It is difficult to remember the password for all these systems.

The security of the application depends on the identification of a user. After identification only, the user can use various applications and resources of the system. Figure 9.2 provides a graphical overview of the traditional authentication method. In this system initially, the user login the client system by entering his user id and password. This password is a plaintext format. Then the client system sends user id and password to the server. Server has a database of user id and password. The user id and password sent by the client is verified by the server with this database called *password table*. If the user id and password matches, the authentication completed and the user is permitted to use different applications on the client.

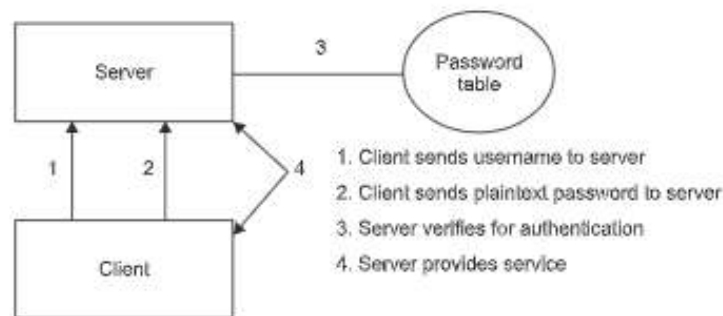


Figure 9.2 Traditional authentication method.

Weaknesses of traditional authentication method:

- In this model, the passwords are stored in plaintext form. If the attacker is able to capture the password, he can access the user id and password from the database.
- The client to server network is connected by insecure channel. Therefore, if the password is sent in plaintext form, then there is a chance that the attacker may capture it during transmission.
- Nowadays, a single user may have different accounts on different systems to use different applications. For this, single user has different id and password for different machines. To remember all this user id and password it is a difficult task for any user. So, a user may select either same id and password or less secure password which is easy to remember.
- Using different malicious software, attacker tries to capture the password. These software includes, key logger, Trojan horse, etc. which when installed on the system, it sends all the information to the attacker.

9.2.2 Two-factor Authentication Method

Traditional authentication method has different weaknesses as we discussed. So, another method called two-factor authentication is used. In this system, identification and authentication of the user take place in two different ways to establish his identity and privileges. In this method, two factors are used for identification and authentication. For example, the user wants to withdraw money from the ATM machine. For these two factors are required: the ATM card issued by the bank and the PIN number of the card. This method is called *strong authentication method*. In the above example, the ATM card is issued by the bank to the authorised account holder of the bank by taking necessary security measures. PIN for the card is initially provided by the bank which the user can change periodically. This provides necessary security. Another example is online banking where physical card is not required. Instead of that user id is required as one of the factors of authentication. The other factor is PIN which sends to the registered mobile number of the user. This PIN number is OTP (one time password). OTP is the most secure way of encryption because PIN is no longer valid to give access to the system. Its lifetime is small and for every new transaction user gets new OTP. This provides strong authentication and identification of the user. This provides more security. This system also has some drawbacks, particularly the use of new technology. Therefore, many times we read the news about hacking of bank accounts and transfer of money from victim's account. This method suffers by Trojan horse and man-in-the middle attack.

9.2.3 Biometric Authentication Method

The two-factor authentication method based on physical devices such as in the first example the ATM card and in the second example the mobile phone on which the OTP is received. The lost of ATM card or block of mobile phone by the attacker are the weakness of this method. One more method of authentication is *biometric authentication*. This method uses thumb impression, iris or voices for authentication.

You may know that in Aadhar card project of Government of India, before issuing the card, the fingerprint and the iris impression of the user is taken. This data is used for the authentication of the user. The most common and widely used authentication method is *thumb impression*. Authentication of employees in different organisation using thumb impression is the most common application of this method. Thumb impression authentication method is also used in college for taking the attendance of the students. In this method, the finger impression is used for authentication because it has distinctive and unique feature. Throughout the life span of the user his fingerprint remains invariant. In this method, the scan image of fingers is taken and stored in the database. Apply different image processing on this image which includes edge reduction feature extraction and matching algorithms. This is shown in Figure 9.3. If this step is not processed properly, there is a chance of introducing some noise in the image which creates a serious problem during authentication. The next step is similarity operations between two sets carried out. This can be done by performing different transformation operations on the image. These include, scaling, translation and rotations of the image. Using the score for an image is calculated which is used to take final decision. This decision is based on some threshold value of the score.

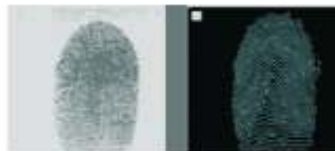


Figure 9.3 Fingerprint authentication.

If the score is below the threshold, the decision is that fingerprints are not matched; if the score is above the threshold, a correct match is declared.

There are different issues in fingerprint authentication system. First issue is performance issue. It is known as *fail to enroll rate*. This occurs as some people has no finger, or may have very faint fingerprint impression. The authentication system cannot work for such users. The second issue is a 'reject' option in the system. This issue is based on the quality of the input image. The poor quality input is not accepted by the system during authentication. And in such cases, the authentication of the user failed. This authentication is failed due to non-cooperative users, dirt on the fingers and improper usages. The performance of the authentication also depends on the quality of scanner used for this purpose.

The drawback of this method is that if there are cuts and bruises on the fingers, then match of the current finger impression and the stored finger impression in the database never match and failure in authentication of the particular user. This method also failed if noise is added during the preprocessing of the finger impression images. Due to human skin elasticity, the authentication may fail as the new finger impression cannot match with the images in the database. Apart from all these drawbacks, thumb impression is the widely used method for authentication due to its low cost and high performance.

There are two types of error in fingerprints and biometrics in authentication system. These recognition errors are the *false accept ratio* (FAR) and the *false reject ratio*

(FRR). The FAR is the probability that the wrong user is identified and authenticates by a system. FRR is the probability that a true user is treated as non-authenticate user. These ratios are exactly opposite of each other, i.e., when we try to reduce one error, other error will increase. The threshold is shown with respect to the error rates in Figure 9.4. This graph is called the ROC (receiver operating characteristic) curve. If the threshold is changed to reduce one of the error rates, other error rate would increase.

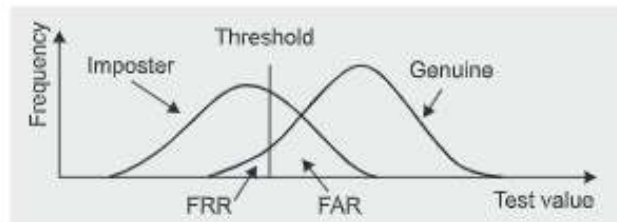


Figure 9.4 Error rates.

Biometrics-based authentication is more secure as compared to password-based authentication system. In password-based authentication, loss of physical card, forget the password, and capture the password by the attacker are the main issues. Whereas in biometric, there is no question of loss or forget at the same time it is difficult to steal or forge the biometric signals.

9.2.4 Extensible Authentication Protocol (EAP)

For more security, most of the organisations do not depend on the user id and passwords. Such organisations use new extensible authentication protocol (EAP) to support the point-to-point protocol (PPP). EAP is standard which provides an infrastructure to clients and servers for authentication.

Microsoft Windows operating system uses extensible authentication protocol to authenticate network access for point-to-point protocol (PPP) connections and for IEEE 802.1X-based network access to authenticate ethernet switches and wireless access points (APs).

With EAP, the PPP peers negotiate to perform EAP during the connection authentication phase. The peers negotiate the use of a specific EAP authentication scheme when the connection authentication phase is reached. This method is known as an *EAP method*.

Then EAP allows for an open-ended exchange of messages between the client and the authenticating server. This exchange is based on the parameters of the connection. The client first requests for the service. Once the authentication takes place, the authentication server sends the response to the client. EAP provides the flexibility to allow for secure authentication methods.

An EAP infrastructure includes:

- *Access client*: The computer that sends the request to access the network.
- *Authenticator*: It is an access point or network access server.
- *Server*: A computer system which is responsible for authentication.

The client and the authentication server exchange the messages using software and a data link layer transport protocol such as PPP or IEEE 802.1X. The EAP authenticator and the authentication server send EAP messages using RADIUS. The result is that EAP messages are exchanged between the EAP components on the client and the authentication server. Figure 9.5 shows EAP working.

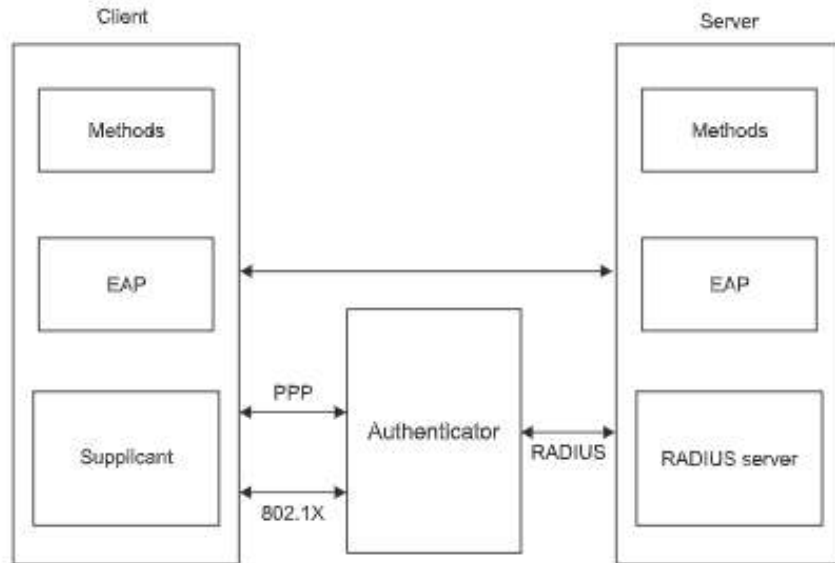


Figure 9.5 EAP infrastructure and information flow.

There is no need of the support of EAP authenticator to any specific EAP methods. User can use EAP to support authentication schemes such as Generic Token Card, One Time Password (OTP), Message Digest 5 (MD5)-Challenge, Transport Layer Security (TLS) for smart card and digital certificate-based authentication, and future authentication technologies. EAP is a critical technology component for establishing secure connections.

EAP methods, such as MD5, CHAP, are used for dial-up remote access or site to site connection, VPN. Table 9.1 lists the different types of access and the available EAP methods user can use.

Table 9.1 EAP methods for network access

Type of network access	EAP methods
Dial-up connections	MD5 CHAP, TLS
802.1X to wireless AP	PEAP-MS-CHAP v2, TLS
802.1X to an authenticating switch (wired)	TLS, MD5 CHAP, PEAP_MS-CHAP v2, PEAP-TLS
VPN	TLS, MD5 CHAP

9.3 MESSAGE DIGEST

9.3.1 MD2

Encryption techniques provide the confidentiality to the message. Authentication techniques provide the access control. But if the third party captures the message and modifies it during transmission, the integrity checking of the message is necessary. Integrity of a message is checked using the hash value or message digest calculated from the message. The hash value of the message is derived from the message which remains same if the message is unaltered. For a small change in the message, its hash value is also changed. Therefore, hash value is used to check the integrity of the message. There are various algorithms available which can generate the hash value of the message. Two of them are message digest and secure hash algorithm.

In 1989, Ronald Rivest developed a method to calculate the hash value of the message. That hash value is called *message digest* (MD) and the algorithm is called *message digest algorithm*. The initial version of message digest algorithm is MD2. It is a cryptographic hash function. This algorithm is optimised for 8-bit computers. Later on other message digest algorithms have been proposed those are MD4 and MD5. One more hashing algorithm is developed called secure hash algorithm (SHA). MD2 is used in public key infrastructures as part of certificates generated with MD2 and RSA. Each of these algorithms generate different hash value called message digest. For a message multiple of 128 bits (each block is of 128 bits), MD2 produces a message digest of 128 bits long.

The working of MD2:

1. The input is a message of an arbitrary length.
2. Pad necessary number of bits to the message to make it to a multiple of 16 octets.
3. Add a checksum of 16-byte.
4. The 128-bit message digest or hash value is generated.

Padding

Padding is always used for MD2 even if the message is of multiple of 16 octets. If the message is a multiple of 16 octets, 16 octets of padding are added. The number of padding octets is from 1 to 15 necessary to make the message a multiple of 16 octets is added. Each pad octet specifies that the total number of octets was added.

Checksum

MD2 checksum is a 16-octet. It is just like a message digest, but is not secure by itself. It is always appended to the message. For the actual calculation, a 48-byte auxiliary block and a 256-byte table generated indirectly from the digits of the fractional part of pi are used. Once all of the blocks of the modified message have been processed, the first partial block of the auxiliary block becomes the hash value of the message.

Final Pass

It is similar to the checksum computation.

In 2004, MD2 was shown to be vulnerable to a pre-image attack with time complexity equivalent to 2^{104} applications of the compression function.

9.3.2 MD4

In 1990, Ronald Rivest developed a new version of message digest called MD4. Its operations are 32-bit word oriented. This makes it faster than MD2. Like MD2, it can also handle a message of arbitrary length.

The padding to the message is used to ensure that its length in bits plus 64 is divisible by 512. The length of the original message which is 64-bit is concatenated to the original message. The message digest generated using MD4 is a 128-bit. The message is processed in 512-bit blocks in the iterative structure. There are three distinct rounds and each block is processed in these three rounds.

MD4 is vulnerable to an attack developed by Boer, Bosselaers and others. The attack was developed on versions of MD4 with either the first or the last rounds missing. Dobbertin shows many more attacks on MD4 which prove that MD4 should now be considered broken.

9.3.3 MD5

MD4 is broken and treated as non-secure hashing algorithm. In 1991, the new version of message digest was designed by Ronald Rivest. The next and secure version of message digest is MD5 (Message-Digest algorithm 5). It is a widely used hashing function. It generates message digest with a 128-bit hash value. MD5 gains popularity by its use as an internet standard, in a wide variety of security applications, and mostly used to check the integrity of files.

In 1996, a flaw was found with MD5 design, but it is not treated as a complete weakness of MD5. But the flaws discovered in 2004, rises the questions on the use of MD5 for integrity checking of the message.

Generation of Message Digest

For MD5

The length of input message	: Arbitrary
Block size	: 512-bit: K bits
Padding	: 1 to 512 bits
Message length	: $K \bmod 2^{64}$
The length of the output:	128-bit message digest (MD)

The message is padded with 1 to 512 bits as shown in Figure 9.6(a). Padding is always used. That means there is a minimum of 1 bit padding is used. The overall processing of a message to produce a digest is shown in Figure 9.6(b).

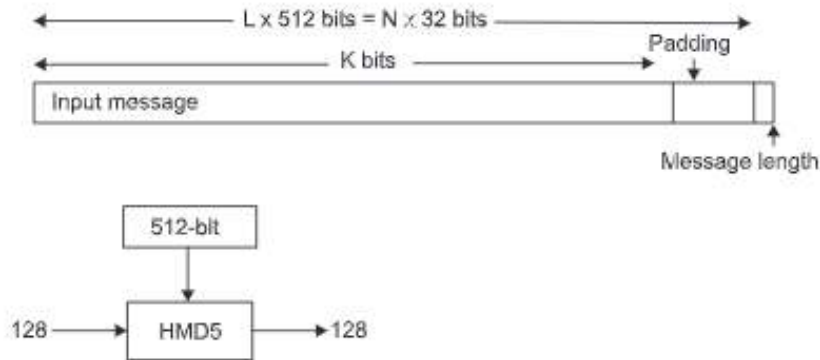


Figure 9.6(a) MD5 message with padding.

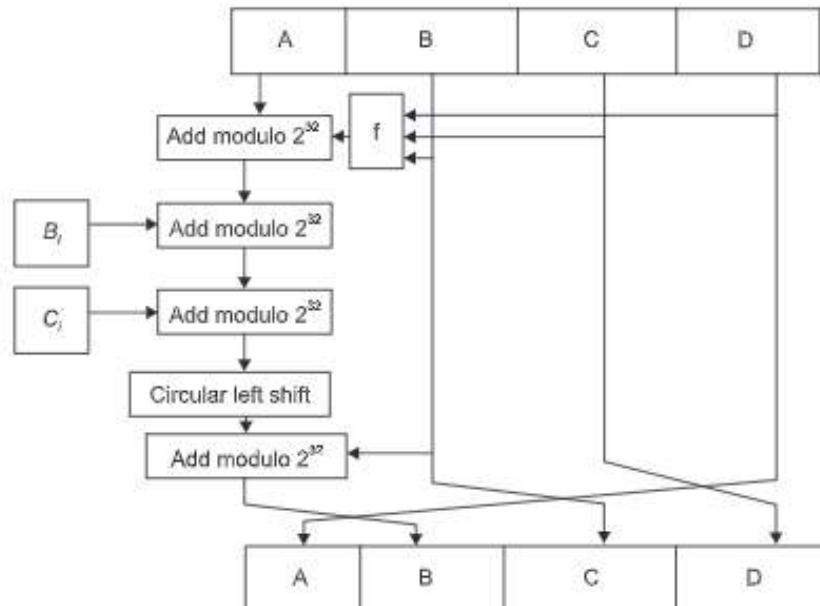


Figure 9.6(b) MD5 operation.

In MD5, there are 4 rounds. Each round has 16 operations. There are total 64 operations. Each operation uses different constants of 32 bits. Each round uses one non-linear function called f function. At a time, a block of 128 bits called state is processed in each round, which is divided into 32 bits sub-blocks each. Let the input of sub-block of the message is denoted by B_i and a 32-bit constant is denoted by C_i . Circular left shift operation is used which rotate the bits by s positions to the left. The shift s varies from round to round. Addition modulo 2^{32} operations are performed.

MD5 processes a variable-length message which is divided into 512-bit block. At a time, one block of 512 bits is processed which produces a 128-bit message digest.

The input message of arbitrary length is appended by 1 to 512 bits called padding and then the 64 bits are used to show the length of the message. This message is divided into a block of 512-bit each. The message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeroes as are required to bring the length of the message up to 64 bits less than a multiple of 512. The remaining bits are filled up with a 64-bit integer representing the length of the original message. For example, the size of the message is 734 bits. We know that the block size for MD5 is 512 bits. Therefore, the message should be divided into two blocks of 512 bits. But 734 is not the multiples of 512 bits so, padding is required. Now, we have to decide how many bits are appended. So, $512 * 2 = 1024$ bits actually required to split the message into two blocks. We know that last 64 bits are used for message length. Therefore, 960 bits ($1024 - 64$) should be the length of the message. But original message is 734 bits long, so we have to append 226 bits ($960 - 734$) to get the message of desired size. So, the padding of 226 bits gives us the message size of 1024 bits. ($734 + 226 + 64$). This 226 bit padding includes the first bit from LHS, i.e., MSB is 1 and other 225 bits are all zeroes, i.e., 100000.....0. If the message is of size 448, with 64 bits which are reserved for length gives the message of size 512. In this case padding is also required and it is of 512 bits so that the total size is 1024 bits. If the message is of size 447 bits, then with 64 bits the size of the message is 511, so 1 more bit is used as padding to get the desired size, i.e., 512 bits. This concludes that whatever be the size of the message, the padding is always required. There is minimum 1 bit to maximum 512 bits are used for padding.

The MD5 algorithm operates on a 128-bit state which is split into four 32-bit words. It is denoted by A, B, C and D. These are initialised to certain fixed constants. Then the 512-bit message block is taken as input which modifies the state. There are four rounds in MD5 and each round has different operations such as non-linear function f , modular addition 2^{32} and shift circular left. One operation within a round is shown in Figure 9.6(b). The possible function used in each round is as shown in Table 9.2.

Table 9.2 Functions used in MD5

Round	f	$f(b, c, d)$
1	$F(p, q, r)$	$(p \text{ AND } q) \text{ OR } ((\text{NOT } p) \text{ AND } r)$
2	$G(p, q, r)$	$(p \text{ AND } r) \text{ OR } (q \text{ AND } (\text{NOT } r))$
3	$H(p, q, r)$	$p \text{ XOR } q \text{ XOR } r$
4	$I(p, q, r)$	$q \text{ XOR } (p \text{ AND } (\text{NOT } r))$

MD5 algorithm consists of 5 steps:

Step 1 *Padding*: Required numbers of bits append to the right hand side of the message. The original message is “padded” (extended) so that its length (in bits) is congruent to 448, modulo 512. The padding rules are:

- In MD5 padding to the original message is always used. Minimum 1 bit and maximum 512 bits are used for padding. The first bit of padding (MSB) is always ‘1’.
- The remaining bits of the padding are always zero.

- Step 2** *Appending length:* At the end of message after padding, the 64 bits are appended which show the length of the original message in bytes. The rules of appending length are:
- The length of the original message is in bytes. Convert it binary format of 64 bits. In case of overflow, use only the low-order 64 bits.
 - Divide the 64-bit length into 2 words (32-bit each).
 - Appended first the low-order word and followed by the high-order word.
- Step 3** *Initialising MD buffer:* MD5 algorithm has a buffer of size 128 bits. This buffer is divided into 4 words each has a specific initial value. The rules of initialising the buffer are:
- The buffer is divided into 4 words, named as A, B, C and D.
 - Word A is initialised to: 0x67452301.
 - Word B is initialised to: 0xEFCDAB89.
 - Word C is initialised to: 0x98BADCFE.
 - Word D is initialised to: 0x10325476.
- Step 4** *Processing message:* The message is split in 512-bit blocks. Each block of 512-bits undergoes 4 rounds of operations. Total 16 operations are performed on each block in each round.
- Step 5** *Output:* The contents in buffer words A, B, C, D are stored in sequence with low-order byte first. The 128-bit message digest is produced.

These 128 bits act as state for the next block of 512 bits. After processing of all the blocks, the contents of buffer A, B, C, D is 128-bit message digest.

Pseudocode for MD5

```

A, B, C, D: initialized buffer words

F(P, Q, R) = (P AND Q) OR ((NOT P) AND R)
G(P, Q, R) = (P AND R) OR (Q AND (NOT R))
H(P, Q, R) = P XOR Q XOR R
I(P, Q, R) = Q XOR (P OR (NOT R))
const[ 1, 2, ..., 64]: Array of special constants (32-bit integers) as:
const[ i] = int(abs(sin(i)) * 232)
M[ 1, 2, ..., N]: Array of blocks of message of size 512-bits

//Round 1
Round1(a, b, c, d, P, s, i)
a = b + ((a + F(b, c, d) + P + const[ i]) shift left by 's' positions)

// Round 2
Round2(a, b, c, d, P, s, i)
a = b + ((a + G(b, c, d) + P + const[ i]) shift left by 's' positions)

// Round 3
Round3(a, b, c, d, P, s, i)

```

```

a = b + ((a + H(b, c, d) + P + const[i]) shift left by 's' positions)
// Round 4
Round4(a, b, c, d, P, s, i)
a = b + ((a + I(b, c, d) + P + const[i]) shift left by 's' positions)

Algorithm:
for k = 1 to N do the following
AA = A
BB = B
CC = C
DD = D
(H[0], H[1], ..., H[15]) = M[k]          /* Divide M[k] into 16 words */
/* Round 1 operations. */
Round1(A, B, C, D, H[0], 7, 1)
Round1(D, A, B, C, H[1], 12, 2)
Round1(C, D, A, B, H[2], 17, 3)
Round1(B, C, D, A, H[3], 22, 4)
Round1(A, B, C, D, H[4], 7, 5)
Round1(D, A, B, C, H[5], 12, 6)
Round1(C, D, A, B, H[6], 17, 7)
Round1(B, C, D, A, H[7], 22, 8)
Round1(A, B, C, D, H[8], 7, 9)
Round1(D, A, B, C, H[9], 12, 10)
Round1(C, D, A, B, H[10], 17, 11)
Round1(B, C, D, A, H[11], 22, 12)
Round1(A, B, C, D, H[12], 7, 13)
Round1(D, A, B, C, H[13], 12, 14)
Round1(C, D, A, B, H[14], 17, 15)
Round1(B, C, D, A, H[15], 22, 16)
/* Round 2 operations. */
Round2(A, B, C, D, H[1], 5, 17)
Round2(D, A, B, C, H[6], 9, 18)
Round2(C, D, A, B, H[11], 14, 19)
Round2(B, C, D, A, H[0], 20, 20)
Round2(A, B, C, D, H[5], 5, 21)
Round2(D, A, B, C, H[10], 9, 22)
Round2(C, D, A, B, H[15], 14, 23)
Round2(C, D, A, B, H[15], 14, 23)
Round2(A, B, C, D, H[9], 5, 25)
Round2(D, A, B, C, H[14], 9, 26)
Round2(C, D, A, B, H[3], 14, 27)
Round2(B, C, D, A, H[8], 20, 28)
Round2(A, B, C, D, H[13], 5, 29)
Round2(D, A, B, C, H[2], 9, 30)

```

```
Round2 (C, D, A, B, F[ 7] , 14, 31)
Round2 (B, C, D, A, F[ 12] , 20, 32)
/* Round 3 */
Round3 (A, B, C, D, F[ 5] , 4, 33)
Round3 (D, A, B, C, F[ 8] , 11, 34)
Round3 (C, D, A, B, F[ 11] , 16, 35)
Round3 (B, C, D, A, F[ 14] , 23, 36)
Round3 (A, B, C, D, F[ 1] , 4, 37)
Round3 (D, A, B, C, F[ 4] , 11, 38)
Round3 (C, D, A, B, F[ 7] , 16, 39)
Round3 (B, C, D, A, F[ 10] , 23, 40)
Round3 (A, B, C, D, F[ 13] , 4, 41)
Round3 (D, A, B, C, F[ 0] , 11, 42)
Round3 (C, D, A, B, F[ 3] , 16, 43)
Round3 (B, C, D, A, F[ 6] , 23, 44)
Round3 (A, B, C, D, F[ 9] , 4, 45)
Round3 (D, A, B, C, F[ 12] , 11, 46)
Round3 (C, D, A, B, F[ 15] , 16, 47)
Round3 (B, C, D, A, F[ 2] , 23, 48)
/* Round 4*/
Round4 (A, B, C, D, F[ 0] , 6, 49)
Round4 (D, A, B, C, F[ 7] , 10, 50)
Round4 (C, D, A, B, F[ 14] , 15, 51)
Round4 (B, C, D, A, F[ 5] , 21, 52)
Round4 (A, B, C, D, F[ 12] , 6, 53)
Round4 (D, A, B, C, F[ 3] , 10, 54)
Round4 (C, D, A, B, F[ 10] , 15, 55)
Round4 (B, C, D, A, F[ 1] , 21, 56)
Round4 (A, B, C, D, F[ 8] , 6, 57)
Round4 (D, A, B, C, F[ 15] , 10, 58)
Round4 (C, D, A, B, F[ 6] , 15, 59)
Round4 (B, C, D, A, F[ 13] , 21, 60)
Round4 (A, B, C, D, F[ 4] , 6, 61)
Round4 (D, A, B, C, F[ 11] , 10, 62)
Round4 (C, D, A, B, F[ 2] , 15, 63)
Round4 (B, C, D, A, F[ 9] , 21, 64)

A = A + AA
B = B + BB
C = C + CC
D = D + DD
End of for loop

// Output of contents A, B, C, D
128-bit Message digest
```

Applications

MD5 digests have been widely used to provide the integrity of the message receiving at the receiver's end. This can be done as follows.

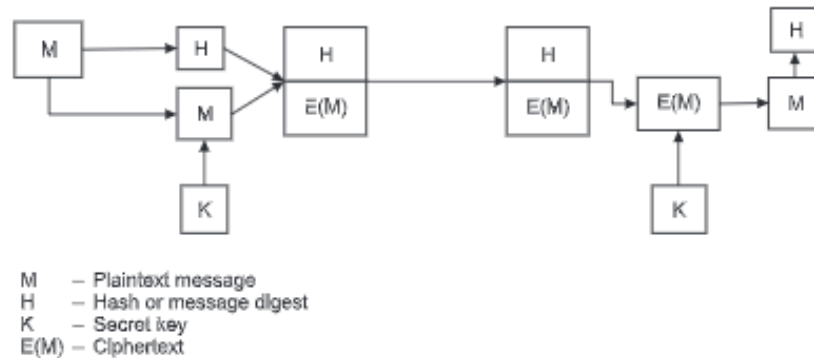


Figure 9.7 Verification of message using message digest.

For example, from Figure 9.7, user sends message M. First the hash value or message digest is computed by the sender. Then the message is encrypted using the secret key. Then the ciphertext with hash value is sent to the receiver. The receiver decrypts the message using the secret key and computes the hash value or message digest of the plaintext. Then compare the newly computed hash value with the hash value sent by the sender. If both hash values are same, the receiver received the same message sent by the receiver. More security is provided by encrypting the message digest with ciphertext by using public key of the receiver.

The most widely use of MD5 is to store passwords. The attack is made by decrypting the password using reverse lookup table. A number of such reverse lookup tables exist for MD5 which make it easy to decrypt password hashed with plain MD5. To prevent such attacks, we can apply the MD5 for hashing the password. Due to this, the time, for capturing the password is increased and it provides more security.

Brute Force Attack

It is not possible to get back the plaintext from the message digest. There is no known way to generate plaintext from the message digest. But the cryptanalyst may be able to get back the plaintext from the message digest using brute force attack. This attack can be made by assuming some plaintext. Calculate the message digest for this assumed message. Compare the calculated message digest with the actual message digest. If both the message digest matches, the assumed message corresponding to computed message digest is the required message. MD5 message digest values are unique. There is no chance to get same message digest for two different messages using MD5. The secure hash algorithm is the replacement for MD5.

Spoofing MD5

A hash technique is repetitive. Collisions between two distinct messages have studied by a Chinese researcher, Xiaoyun Wang. He used these differences to suggest changes that would increase the chances of the two messages digest generating the same result. Another successful spoofing attack is done by Benne de Weger, a Dutch scientist against MD5. He took two colliding documents and adds non-random data. Due to this the documents collide continues. He suggests an example, suppose user A could prepare a document and he has to take a sign of his higher authority on that document. Simultaneously, he prepares one more document having the same hash value. After the signature of his higher authority on the document, he could adjust his original document with the document signed by his higher authority having the same hash value. (From *The Economist Technology Quarterly*, March 8, 2008, page 10.)

Comparison of MD5 with MD4

A comparison of MD5 with MD4 is given in Table 9.3.

Table 9.3 Comparison between MD5 and MD4

Algorithm	MD5	MD4
Number of steps	64	48
Additive constants	For each step, a different additive constant is used.	For the first round no additive constant is used. For each step of the second round same additive constant is used. Another additive constant is used for each of the steps of the third round.
Number of logical functions	4	5
Speed	Slower	Faster
Security	More secure	Less secure

9.3.4 SHA-1

The possible attacks against MD5 are possible. The more secure hash algorithm is SHA (secure hash algorithm). The hash value generated by SHA is unique for a given message. NIST proposed a secure hashing algorithm known as SHA-1. For a small change to a message will have a very high probability to get different message digests. These algorithms are called secure because, for a given algorithm, it is computationally infeasible to

- find a message which have a same message digest or
- find two different messages those have the same message digest.

There are five different variants of secure hash algorithms. The five algorithms are SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512. The latter four variants are sometimes collectively referred to as SHA-2.

SHA-1 is widely used in security applications and protocols such as: PGP, TLS, IPsec, SSL, SSH, S/MIME. It has variable hash values as per its variants. No attack has been detected against SHA-2.

SHA-1 Algorithm

The message digest or hash value generated using SHA-1 is 160-bit. For a message with length $< 2^{64}$ bits, it produces a message digest of 160-bit. Like MD5, it also processes a block of 512-bit input message. The message digest or hash value generated by SHA-1 is used as input to the digital signature algorithm. This hash value is used for generation or verification of message using digital signature. This improves the performance and efficiency of the algorithm as hash value is smaller than original message. The same digital signed hash value is used to check the integrity of the message.

The hash values for two different messages are never same. It is computationally infeasible to generate the message from the hash value. This proves that SHA-1 is more secure. If there is any change happened in a message, it is possible to identify due to hash value of the message. The architecture of SHA-1 is based on MD4 architecture.

Features

SHA-1 generates a message digest of 160-bit for a message of arbitrary size. It uses padding as MD5 algorithm to make the message size is of multiples of 512-bit. But, it is not defined for the message longer than 2^{64} . Then the length of the message is added to the end of the message. The length indicates the number of bits present in the message. For empty message, the length is 0-bit. For compactness, the message is represented in hexadecimal. The working of SHA-1 is shown in Figure 9.8.

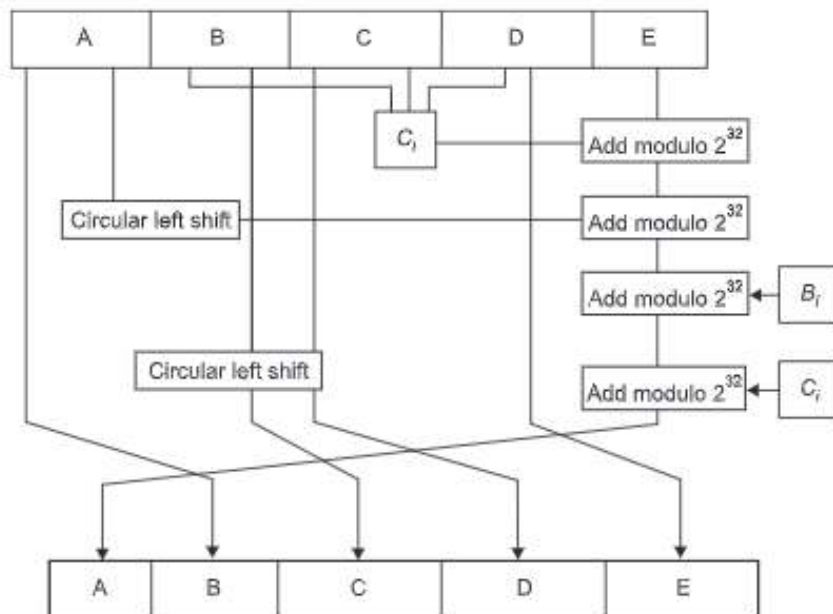


Figure 9.8 SHA-1 operation

From Figure 9.8, there are 5 states A, B, C, D, E and each has 16 operations. So, there are 80 operations in SHA-1. The bits are rotated by circular left shift operations by s bit positions. The value of s varies for each operation. The size of each state is 32-bit. A non-linear function f is used which varies for different rounds. Addition modulo 2^{32} is used for modular additions. A constant C_i is used in each round. Each block of message of size 512-bit is indicated as B_i . The cryptanalysis of SHA-1 is more difficult than its predecessor SHA-0.

NIST has published four variants of SHA are: SHA-224, SHA-256, SHA-384, and SHA-512. The number indicates the message digest generated in bits. The SHA-256 uses 32 words; SHA-512 uses 64-bit words. They use different shift amounts and additive constants. Other operations are identical for both SHA-256 and SHA-512. The number of rounds is different for SHA-256 and SHA-512. But SHA-1 is more popular than other variants of SHA family.

SHA-1 Working

The following example illustrates the padding process in SHA-1. It is similar to MD5.

Message is appended by 1 and necessary number of "0" followed by a 64-bit integer are appended to the end of the message to produce a padded message of length $512 * n$. The 64-bit integer shows the length of the original message. The padded message is then processed by the SHA-1 as 512-bit blocks at a time. Suppose a message has length $l < 2^{64}$. Before it is input to the SHA-1, the message is padded on the right as follows:

- (a) "1" is appended.
For example, if the original message is "01010110", this is padded to "010101101".
- (b) "0"s are appended. The number of "0"s will depend on the original length of the message. The last 64 bits of the last 512-bit block are reserved for the length of the original message.
For example, suppose the original message is:
11100101 01100110 01101011 01100100 01101101 01101101.
Since there are 48 bits in the message, total 400 bits are needed to append the original message to get $448 \bmod 512$. First append "1" to the message, we get:
1110 0101 0110 0110 0110 1011 0110 0100 0110 1101 0110 1101 1.
Remaining 399 bits append should be "0". The result in hexadecimal is:
E5666764 6D80000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000.
- (c) Obtain the 2-word representation of length of the original message. If $l < 2^{32}$, then the first word is all zeroes. Append these two words to the padded message.
For example, suppose the original message is above having 48 bits length. Length should be computed before padding so it is 48 bits. The two-word representation of 48 is hex 00000000 00000030. Hence, the final message after padding and appending message in hexadecimal is:
E5666764 6D80000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000030.

The new message is now multiples of 512. As each block is of 512 bits, there are n blocks of 512 bits in the new message. Then process each block sequentially one by one.

Functions and constants: SHA-1 uses 80 functions and operates on three 32-bit words and produces a 32-bit output word. The working is as follows:

$$f(i; B, C, D) = (B \text{ AND } C) \text{ OR } ((\text{NOT } B) \text{ AND } D) \quad (0 \leq i \leq 19)$$

$$f(i; B, C, D) = B \text{ XOR } C \text{ XOR } D \quad (20 \leq i \leq 39)$$

$$f(i; B, C, D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D) \quad (40 \leq i \leq 59)$$

$$f(i; B, C, D) = B \text{ XOR } C \text{ XOR } D \quad (60 \leq i \leq 79).$$

$$\text{Const}(i) = 5A827999 \quad (0 \leq i \leq 19)$$

$$\text{Const}(i) = 6ED9EBA1 \quad (20 \leq i \leq 39)$$

$$\text{Const}(i) = 8F1BBCDC \quad (40 \leq i \leq 59)$$

$$\text{Const}(i) = CA62C1D6 \quad (60 \leq i \leq 79).$$

Generation of the Hash Value (Message Digest)

There are many methods for the generation of the message digest. Here we discuss two methods for the generation of message digest. The execution time requires for second method is more as compared to first method.

Method 1: The message is prepared as above by appending padding bits and length. Two buffers are used. Each buffer has five 32-bit words in addition of a sequence of 80 words. The words are labelled as A, B, C, D and E for the first buffer and H_0 to H_4 for the second buffer. The sequence of 80 words is labelled from W_0 to W_{79} . A temporary buffer is also used called TEMP.

Before processing any blocks, the H words are initialised in hexadecimal as:

$$H_0 = 67452301$$

$$H_1 = \text{EFCDA}889$$

$$H_2 = 98BADCFE$$

$$H_3 = 10325476$$

$$H_4 = \text{C3D2E1F0}$$

The 16-word blocks B_1, B_2, \dots, B_n are processed in order. This is shown in Figure 9.7. The processing of each B_n involves 80 steps.

- B_n is divided into 16 words W_0 to W_{15} where W_0 is the left-most word.
- For $n = 16$ to 79, let $W_n = S^1(W_{(n-3)} \text{ XOR } W_{(n-2)} \text{ XOR } W_{(n-14)} \text{ XOR } W_{(n-16)})$.
- Let $A = H_0, B = H_1, C = H_2, D = H_3, E = H_4$.
- For $n = 0$ to 79 do

$$\text{TEMP} = S^5(A) + f(n; B, C, D) + E + W_n + C_n;$$

$$E = D; D = C; C = S^{30}(B); B = A; A = \text{TEMP};$$
- Let $H_0 = H_0 + A, H_1 = H_1 + B, H_2 = H_2 + C, H_3 = H_3 + D, H_4 = H_4 + E$.

After processing B_n , the 160-bit message digest is generated as:

$$H_0 \ H_1 \ H_2 \ H_3 \ H_4.$$

Method 2: In the above method an array is used to implement the sequence W_0, \dots, W_{79} . This reduces the execution time. But it occupies more space. To reduce the space, one has to use a circular queue using an array of 16, 32-bit words from W_0 to W_{15} . Let M stands for masking bits.

$M = 0000000F$. Then processing of B_n is as follows:

- (a) Split B_n into 16 words W_0 to W_{15} , where W_0 represents the leftmost word.
- (b) Let $A = H_0$, $B = H_1$, $C = H_2$, $D = H_3$, $E = H_4$.
- (c) For $n = 0$ to 79 do
 - $s = n \text{ AND } M$;
 - if $(n \geq 16)$ $W[s] = S^1(W[(s + 13) \text{ AND } M] \text{ XOR } W[(s + 8) \text{ AND } M] \text{ XOR } W[(s + 2) \text{ AND } M] \text{ XOR } W[s])$;
 - $TEMP = S^3(A) + f(n; B, C, D) + E + W[s] + C(n)$;
 - $E = D$; $D = C$; $C = S^{30}(B)$; $B = A$; $A = TEMP$;
- (d) Let $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$, $H_4 = H_4 + E$.

After processing B_n , the 160-bit message digest is generated as:

$$H_0 \ H_1 \ H_2 \ H_3 \ H_4.$$

Comparison of various SHA is given in Table 9.4.

Table 9.4 Comparison of various SHA

Algorithm	SHA-0	SHA-1	SHA-256/224	SHA-512/384
Size of message digest in bits	160	160	256/224	512/384
Size of internal state in bits	160	160	256	512
Size of block in bits	512	512	512	1024
Length of message	$2^{64} - 1$	$2^{64} - 1$	$2^{64} - 1$	$2^{128} - 1$
Size of word	32	32	32	64
Number of steps	80	80	64	80
Collision	Yes	2^{62} attack	None yet	None yet

A comparison of SHA-1 and MD5 is given in Table 9.5.

Table 9.5 Comparison between SHA-1 and MD5

Algorithm	SHA-1	MD5
Size of message digest in bits	160	128
Cryptanalysis attack	Vulnerable to attack	Not yet vulnerable
Speed	Slow	Fast
Number of steps	80	64
Buffer size in bits	160	128

9.3.5 HMAC

Message authentication code (MAC) allows the receiver to check the integrity of the message. Suppose user A sends a message to receiver B. At the receiving end how user B knows that it is the same message send by user A or it is a modified message by a

third party. With the help of MAC receiver can check the message and decide whether it is the same message sent by user A or not. This is called *integrity checking*. MAC is used to check the integrity of the message. At the same time MAC helps to check the authentication of the sender. To provide the security hash function is used along with the encryption techniques. This is called *hash message authentication code (HMAC)*. Hash function is used basically to check the integrity of the message and also makes digital signature technique more efficient. It also concerned about the non-repudiation of origin and validating identity of originator. The hash value of a message is indicated by $h(m)$. The requirements of hash functions are as follows:

- It can be applied to a message M of any size.
- It produces fixed-length output message digest or hash value h .
- For any message M , one can easily compute $h = H(M)$
- The hash value h is public and if someone knows the value of h , it is infeasible to find the original message from hash value such that $H(x) = h$, i.e., it works one way.
- If the value of x is given, it is infeasible to find y such that $H(y) = H(x)$, this shows weak collision resistance.
- It is not possible to find any messages x and y such that, $H(y) = H(x)$, this shows strong collision resistance.

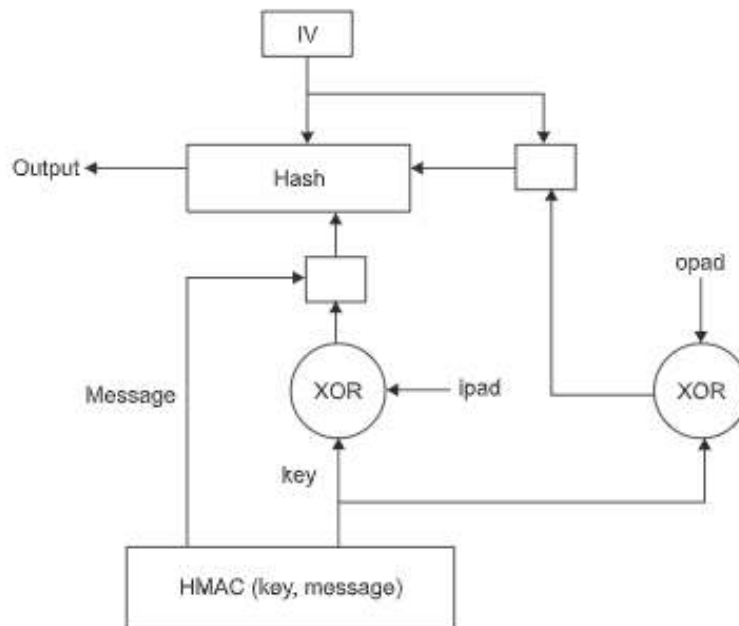


Figure 9.9 Illustration of HMAC.

A keyed-hash message authentication code, or **HMAC**, is a type of message authentication code (MAC) calculated using a cryptographic hash function in combination with a secret key. As with any MAC, HMAC may be used to verify simultaneously

both the integrity and the authenticity of a message. One can use any message digest algorithm such as MD5 or SHA-1, for the calculation of hash value in an HMAC. If we use MD5 then the resulting MAC algorithm is called HMAC-MD5. If we use SHA-1 then the resulting MAC algorithm is called HMAC-SHA-1. The security provided by the HMAC depends upon the strength of the hashing algorithm used in addition to the size and quality of the key.

An iterative hash function splits up a given message into a block of fixed size and process them with a compression function. This is shown in Figure 9.9. For example, MD5 and SHA-1 operate on 512-bit blocks. The size of the output of HMAC is the same as that of the underlying hash function (128 or 160 bits in the case of MD5 and SHA-1). It can be truncated if desired but it reduces the security of the MAC.

$$\text{HMAC}_{\text{key}}(\text{message}) = \text{HASH}(\text{key} \oplus \text{opad} \parallel \text{HASH}(\text{key} \oplus \text{ipad} \parallel \text{message}))$$

where HASH is an iterated hash function, key is a secret key padded with extra zeroes to equalise it with the block size of the hash function, message is the message to be authenticated, \parallel denotes concatenation, \oplus denotes exclusive OR, and the outer padding opad = 01011100 or 0 X5c5c5c...5c5c (5c in hexadecimal repeated for b/8 times) and inner padding ipad = 00110110 or 0 X 363636.....3636(36 in Hexadecimal repeated for b/8 times) are two one-block-long hexadecimal constants.

9.3.6 RIPEMD-160

RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digest) is a message digest algorithm developed by Hans Dobbertin, Antoon Bosselaers and Bart Preneel in 1996. It generates the hash value of 160-bit. It is somewhat similar to MD5 and SHA. It gives the performance similar to SHA-1.

There are different variants of RIPEMD such as RIPEMD-128, RIPEMD-256 and RIPEMD-320 having hash value 128-bit, 256-bit, and 320-bit, respectively. The 128-bit version was the replacement for the original RIPEMD. Its hash value was 128-bit which is same as RIPEMD and it is not secure. RIPEMD-128 and RIPEMD-160 provide more security as compared to RIPEMD-256 and RIPEMD-320-bit versions.

The overviews of RIPEMD-160 are given as follows:

1. *Padding*: Padding procedure is similar to MD5. The length of the message after padding should be equal to 448 modulo 512. The padding is always required though the length of the original message is multiples of 512 bits. The number of padding bits is from minimum 1 to maximum 512. The padding consists of a single 1-bit and remaining number of 0 bits.
2. *Append*: The message is always appended by 64-bit block to indicate the length of the message. It is an unsigned 64-bit integer and contains the length modulo 2^{64} of the original message.
3. *Initialisation*: Initialise 5-word buffer A, B, C, D and E as:
 - A = 67452301
 - B = efcdab89
 - C = 98badcfe

D = 10325476

E = c3d2e1f0

These values are same as used for SHA-1 algorithm.

4. *Message processing:* There are 10 rounds of 16 operations each. The structure of all the rounds is similar, but each round uses different primitive logical function. Each round makes use of an additive constant. The output of the fifth round is added to the chaining variable input to the first round.
5. *Output:* The final value of the buffer is the output. It is 160-bit message digest.

Comparison of RIPEMD-160 with SHA-1 and MD5

Let us summarise the hashing algorithms about their differences and similarities between them. MD5, SHA-1 and RIPEMD-160 are invulnerable to attacks against weak collision resistance. Cryptanalysis of RIPEMD-160 is more difficult than SHA-1. RIPEMD-160 and SHA-1 are slower than MD5. A comparison of these algorithms are given in Table 9.6.

Table 9.6 Comparison of RIPEMD-160 with SHA-1 and MD5

Algorithm	RIPEMD-160	SHA-1	MD5
Message digest	160-bits	160-bits	128-bits
Cryptanalysis attack	Vulnerable to attack	Vulnerable to attack	Not to be vulnerable
Speed	Slow	Slower	Fast
Rounds	5	4	4
Operations	160	80	64
Buffer size (bits)	160	160	128
Endian architecture	Little endian	Big endian	Little endian

9.4 KERBEROS

Authentication of the user is very important to provide the security to any application. Under project Athena at MIT, an authentication system is developed known as *Kerberos*. The objective of this project was to provide a huge network of computer workstations so that undergraduate students can access their files stored on any workstation easily from anywhere in the campus.

For this project, a symmetric encryption is used to provide authentication and security. These authenticate and identify the users and the services in the network to each other. The most common way of authentication is the use of password. The server has a database of userid and password. When any user wants the service in the network, he has first log in by giving his user id and password. The server verifies the user id and password from the database available with it. Once it matches with the database, the user is permitted to use the necessary service. But as we already studied about the security issues of password authentication, they thought about some new system for authentication. So, they developed Kerberos, which address the security issues using a password for authentication.

The key innovation in Kerberos is that there is no need to reveal the secret about the key by the user. They can identify without revealing the secret. Users can prove their identity without sending the secret or password over the network. In Kerberos, an encryption key is used. Timestamp helps to prove that when the request is sent, the user created a timestamp and encrypted with shared secret, the request is sent to service. The service decrypts the request with shared secret key and recovers the timestamp. Then authentication succeeds and the necessary service is provided to the user. If the user uses wrong key and encrypts the timestamp with that key, then the timestamp will not decrypt properly and authentication fails. In this case, the service rejects the request of the user. This system is secure as shared secret key is not transmitted over the network. Kerberos architecture is more complex due to the use of a secret key in a more convenient manner and to patch some of the problems.

9.4.1 Basics of Kerberos

As discussed above, Kerberos is designed for authentication in client-server architecture. In this architecture, the server is not only dependent on the information given by the client but it also verifies the same from its database. The components of Kerberos are: a server, clients, an authentication server, and a ticket-granting server.

Suppose there is no secret key shared by the client and the server. In this case, identifying the client is done using the Kerberos protocol, and a session key is generated for a communication between the server and a client. The Kerberos working is shown in Figure 9.10.

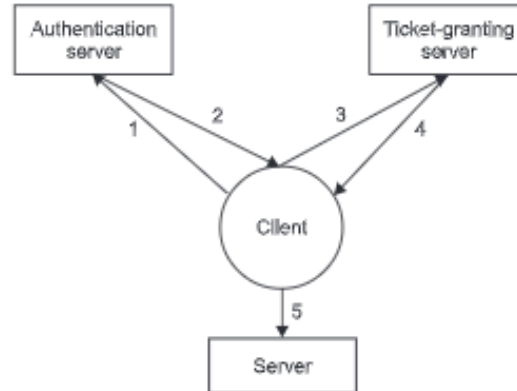


Figure 9.10 Kerberos.

The client sends his requests for a ticket to the ticket-granting service from the authentication server. The authentication server (AS) has a database of user id and password information for all the clients. Authentication server returns an encrypted ticket to the client. The client decrypts the ticket using his secret key. Now, client would like to use the service from the server. But before this, client must be allowed to communicate with the server. So, client submits the ticket to the ticket-granting server. The ticket-granting server verifies the ticket for identifying client and after verification gives a new ticket to client. This ticket will allow the client to make use of

service. Client now submits the ticket to the server. He sends the service ticket and an authentication credential to the server. Server checks the ticket and the authentication credential to make sure whether it is a valid client or not. After verification, server will provide the service to client.

The main objective of Kerberos is authentication. So, there is no need to store the password by the client. The AS serves as an introducer for them. Both the client and the service must have a shared secret key which is stored with the AS. Such keys are used for long time. Figure 9.11 shows the Kerberos authentication.

Authentication can be done using the following steps:

- Step 1** The user sends a request to the authentication server for a service.
- Step 2** Authentication server generates a secret key that will be shared only by the user and the service. This key is called the *session key*. Authentication server replies to the user's request by sending a message. The message has two parts. The first part contains the secret key along with the service's name which is encrypted with the user's key. The other part contains that same secret key along with the user id encrypted with the server's key. In Kerberos phrasing, the first part is called the *user's credentials* and the second part is called *ticket*.

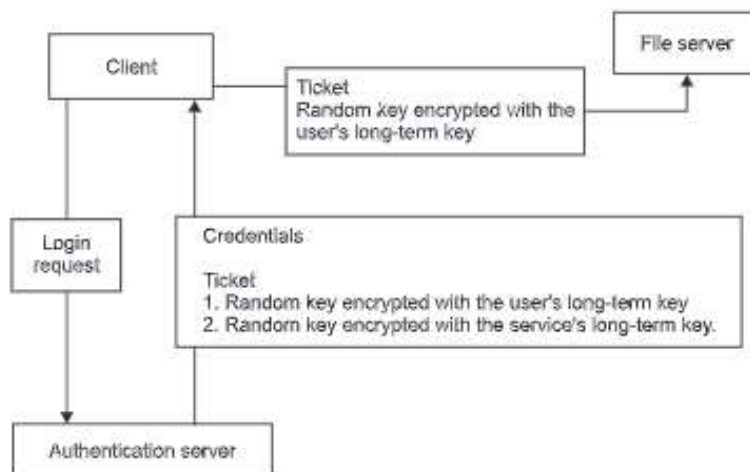


Figure 9.11 Authentication.

- Step 3** Then the user generates a timestamp, and encrypts it with the secret key. User sends the timestamp called *authenticator* along with the ticket, to the service. The service decrypts the ticket with its secret key, recovers the session key. This key is then used to decrypt the authenticator. The service trusts the authentication server for the authentication of the user. So, at the service no further authentication process of the user takes place. Sometimes, the user may want the authentication of the service so the service takes the timestamp from the authenticator, adds the service's own name to it, and encrypts it with the session key. This encrypted message is sent to the user.

For every service if the user contacts the authentication server, the load on AS is increased and it takes time for authentication of the users. To reduce the load on the authentication server, one more server is introduced called *ticket-granting server* (TGS). Now, in this case the user requests his first service is for TGS, which then grants additional tickets for other services. Thus, the database of user id and password is located with authentication server and the trust is with TGS.

9.4.2 Kerberos Ticket-granting Approach

Here, the term “client” refers to a user or to a client-side application program operating on behalf of a user. The term “application” server refers to a remote computer which provides a shared service. The term “password” refers to the user’s password or to a cryptographic key derived from it. The term “session” key refers to a cryptographic key issued for use in communication between a client and an application server, which is valid for some defined interval of time.

A ticket and an authenticator are the two basic credentials used in Kerberos. Both the credentials are based on private key but both use different private key. When the client or user login on a computer, he has to enter his password. He gets a ticket from the authentication server and send this ticket to the application server as a part of the request for a service. Then the client sends an authenticator to the application server along with the ticket. The application server uses the authenticator and the ticket to verify that the request was sent by the client to whom the ticket was issued.

To achieve these objectives, after the initial exchange with the authentication server, the AS issues a “ticket-granting ticket” (TGT) to the user. TGT contains a session key to be used for subsequent ticket requests. Each TGT have fixed life span generally it is set to 8 hours. The use of TGT reduces the load on AS and also there is no need for the user to authenticate himself every time for different services in a network.

The initial message was sent automatically at the time of login from the client to the authentication server. The initial message contains the user’s identification number and request for a TGT as shown in Figure 9.12. The authentication server

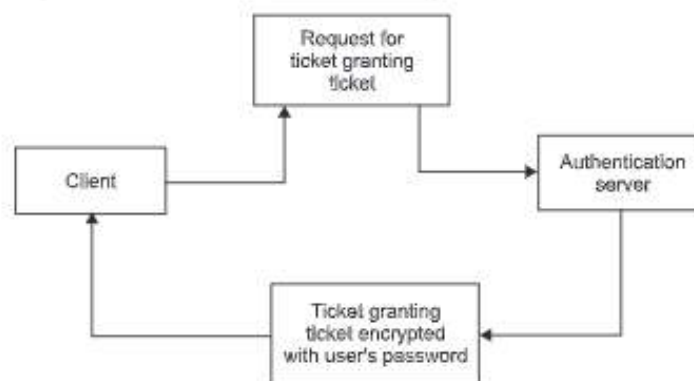


Figure 9.12 Ticket-granting ticket.

replies back with a message encrypted with the user's password and containing a TGT encrypted with the user's password (Figure 9.11). After receiving the message, the client decrypts the TGT message to get the session key. The user uses this session key for subsequent ticket requests. This session key for ticket-granting requests is referred to as the TG key.

9.4.3 Ticket-Granting Server

To reduce the load on the authentication server, one more server is introduced called *ticket-granting server* (TGS). TGS resolves the problem of password re-entry every time for new service requests. The TGS is located on the same server where authentication server located but logically it is different from AS. The purpose of the TGS is to provide the ticket and session key so that user has to enter his password only once and obtained additional services in the network by the use of ticket and session key.

Initially, the user sends the request for a ticket from the AS to talk to the TGS. This ticket is called the *ticket-granting ticket*, or TGT. The session key is encrypted using the user's secret key. After receiving the TGT and the session key, the user requests a TGS for a ticket. This can be done at any time if he wants to use any service. The reply from TGS is encrypted with the session key. The user already has a session key, so there is no need of his own secret key. It is sort of like when you visit some industry or organisation. You have to show your regular ID to the receptionist at the counter to get a guest ID (visitor card) for visit the industry or organisation. This is work like client's request to AS. After verifying the user ID, the receptionist issue a guest ID or visitor card just like AS replies to the user by sending TGT and the session key. Now, when you want to enter various rooms in the industry or organisation, instead of showing your regular ID over and over again, which might make it vulnerable to be dropped or stolen, you have to show your guest ID, which is only valid for a short time anyway. This is like user does not have to use his password once AS gives him the TGT and the session key. If it was stolen, you could get it invalidated and be issued a new one quickly and easily, something that you could not do with your regular ID.

The advantage of above scheme is that session key and ticket are used instead of user's secret key. Therefore, if the session key is captured by the attacker, less damage is happened as session key and TGT are valid only for a limited time period. But if the user has to use his secret key and the key is captured by the attacker, then the more damage is happened than session key and the TGT as the life of secret key is more. This TGT, as well as any tickets that you obtain using it, is stored in the credentials cache. The term "credentials" actually refers to both the ticket and the session key in conjunction.

Once the client gets a TG key, then the client requests for a specific service. This is shown in Figure 9.13. The client sends a request to the TGS to obtain a ticket for the service. The TGS can verify the client identification information encrypted in the message with its database of the TG key. Each ticket has a timestamp. The timestamp protects from reuse of the message.

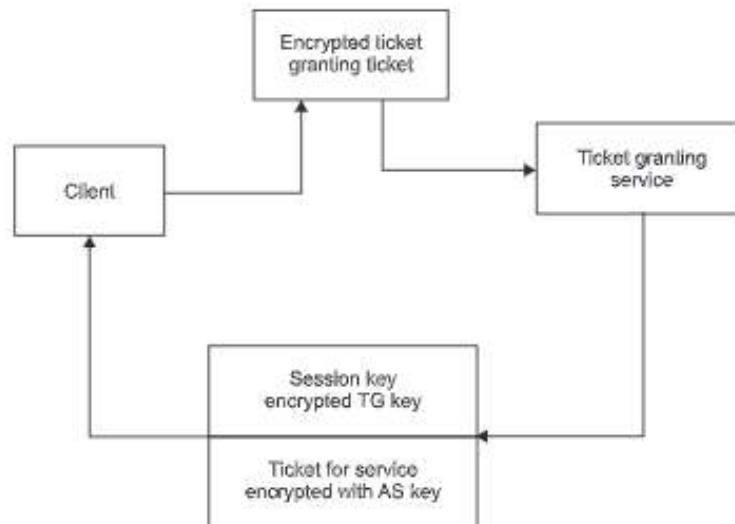


Figure 9.13 Subsequent requests for services from the ticket-granting service.

After checking the client's identity, the ticket-granting service sends a ticket to the user. The ticket contains information about the identity of the client and the newly generated session key. The information with the ticket cannot be changed or altered by the client. He can only forward it to the authentication server.

The client then sends a message to the application server. This message contains the ticket and an authenticator as shown in Figure 9.14. As noted above, the authenticator is constructed by the client, and contains identifying information and a timestamp. After receiving the message, the application server can decrypt the ticket, because it is encrypted with the authentication server's own key. The ticket contains the session key which is used to decrypt the authenticator. For the valid request, the data embedded in the authenticator must match that in the ticket. Further messages between the client and the application server may be encrypted using the session key.



Figure 9.14 Communication between the client and the application server.

9.4.4 Kerberos Third-party Authentication Model

In the Kerberos system, one or more trusted *authentication servers* may be used (termed KDCs or key distribution servers). This is used to provide third-party authentication services which are helpful for cooperating systems and applications. Client acquires *tickets* from the trusted authentication server(s) which can be used to provide the

proof of identification for subsequent request for service and applications. This ticket is encrypted so it is secured in transmission. The detail of the Kerberos authentication is as follows:

1. The user wants some service so he first sends his request to an authenticate server. This request contains the user's name and the name of the service granting server that he will use.
2. The user login on the client and requests for a ticket-granting ticket.
3. After authentication using password and username, the initial authentication ticket is granted by the AS to the client.
4. The client then submits this ticket to the ticket-granting service for a particular service.
5. The ticket-granting service issues a ticket to the client.
6. The client now submits the ticket to the particular server for the desired service.

The details of the communications between a client, the KDCs, and the various services used by the client are rather complex. Figure 9.15 graphically illustrates the interactions between different systems involved in the Kerberos network.

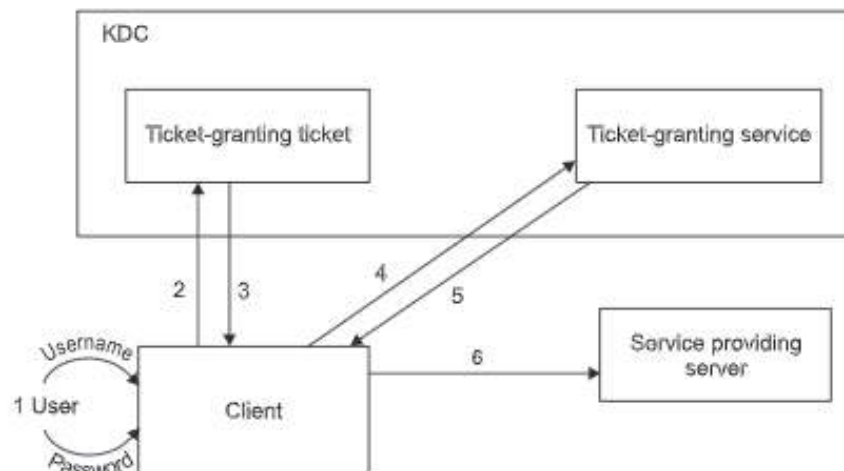


Figure 9.15 Authentication model.

9.4.5 Kerberos Authentication Model: Definitions and Notational Conventions

Some terms and notational conventions used in Kerberos authentication model are:

1. Authentication ticket: It is a record of authentication issued by an authentication server to a client system as a proof of that client's user being authentic.

2. Authenticated service: A service which is only provided to authenticated users via Kerberos and whose clients can present valid authentication tickets as proof of authentication.

3. Target service: The service for which a client is requesting a ticket or to which the client is presenting a ticket.

4. Initial ticketing service: It is the service by which the clients receive their initial tickets.

5. Ticket-granting service: The service by which clients receive tickets to specific target services.

6. Ticket-granting ticket: A ticket provided on demand by the initial ticketing service which must be presented to the ticket-granting service in order to request a service ticket.

9.4.6 Kerberos Authentication Model

The Kerberos authentication model uses a symmetric key encryption technique. Kerberos IV uses DES algorithm and Kerberos V uses DES and IDEA algorithms. To provide more security, double encryption technique is used. For encryption two keys are used, i.e., user password and the session key. The user password have a long life span and used only for initial authentication whereas session key has a life span of 8 to 10 hours and used for requesting different services after initial authentication.

The user first login on the client by using his user id and password. Client sends the request for a ticket to the authentication server for the particular user by providing his user id to AS and not the password. The authentication server verifies the user id and sends the encrypted ticket to the client. If the user is able to decrypt the ticket by his password then the user is considered as authenticate. Then the user sends the ticket to the service, he wants to use. If a service is able to decrypt a ticket using its own secret key, the service may presume that the user is authentic.

In this way, without passing the password information over the insecure channel, the authentication takes place in Kerberos environment. So, it is difficult for the attacker to capture the secret information about the user. The authentication in Kerberos takes place in 6 steps as shown in Figure 9.16.

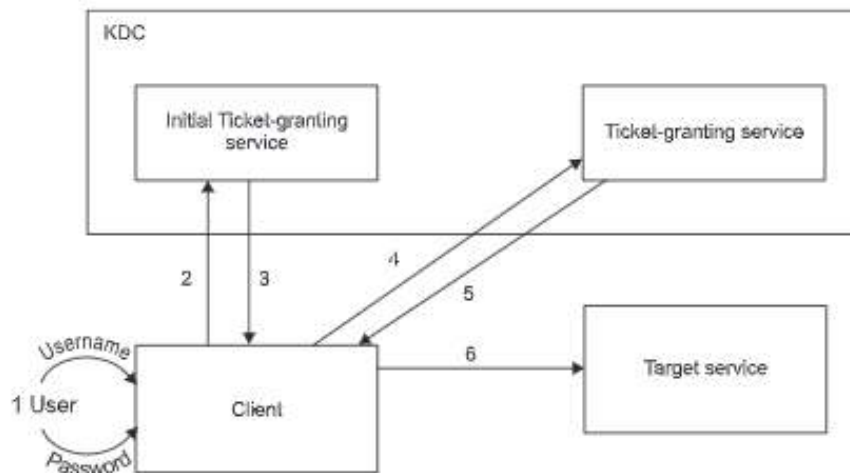


Figure 9.16 Kerberos authentication model.

- Step 1** The user first login on the client by using his user id and password.
- Step 2** The client sends a request to the AS requesting a ticket for the user. This request is totally unauthenticated and it contains only user id and not the password of the user.
- Step 3** The ticketing service verifies the user's name in its database. If user name is in the database then he is an authenticate user and ticketing service generates a unique session key for later use during the user's authenticated session. This ticket sends to the client a double-encrypted ticket-granting ticket and the session key in the form:

$$\mathbf{K_{user}\{K_s, K_{tgs} \{T_{tgs}, K_s\}}}$$

The client then decrypts the ticket-granting ticket using the user's password. If the client successfully decrypts the ticket using the user's password, then user is authentic. Then the client stores the ticket $TGT(K_{tgs} \{T_{tgs}, K_s\})$ for later use.

- Step 4** Then the client sends a ticket request to ticket-granting service (TGS) for a particular service requested by the user. This request for ticket is in the form:

$$\{\mathbf{TGT, K_s\{request, client-IP, timestamp\}}\}$$

(where $TGT = K_{tgs}(T_{tgs}, K_s)$)

- Step 5** The ticket-granting service decrypts the TGT using its own secret key (K_{tgs}) and the rest of the part of the message is decrypted by using the session key. If the ticket-granting service successfully decrypts the ticket, it gets the following information:

- The TGT was issued by authenticate ticketing service.
- The request for the service is from the authenticate user.

Once the authentication is completed the TGS generates a session key and the ticket for a requested service. The TGS sends the session key and the ticket to the client machine in the form:

$$\mathbf{K_s\{K_{session}, K_{ser}\{T_{service}, K_{session}\}}}$$

- Step 6** The client machine decrypts the service ticket using the session key (K_s) and yield the session key ($K_{session}$) and an encrypted service ticket ($K_{ser}\{T_{service}, K_{session}\}$).

The client then submits the encrypted ticket to the requested service. The service decrypts the ticket using its own secret key (K_{ser}). If the decryption is successful, the target service authenticates the user. The communication between the client and the service now start. They can use the session key for secure communication.

User can access other services in the Kerberos using steps 4, 5 and 6 repeatedly from the client machine.

9.4.7 Cross-Realm Authentication

So far we have discussed the client using the service in one Kerberos environment. If the user from one Kerberos environment wants to use the services from other

Kerberos environment, the authentication of that user should be done by the Kerberos which the user belongs is called *cross-realm authentication*. So, the user can use the services from other Kerberos environment without authentication by that other Kerberos environment. The users or clients of one realm use Kerberos to authenticate to services which belong to a realm other than their own. This property known as *cross-authentication*. It is based on a trust between the Kerberos involved. This relationship may be mono-directional, or bi-directional. *Mono-directional* means the users of Kerberos environment A can access the services of Kerberos environment B but not vice-versa. *Bi-directional* means the users of Kerberos environment A can access the services of Kerberos environment B and vice-versa.

We discuss the case where there is only one authentication server and only one ticket-granting server. These servers may or may not be installed on the same machine. This can work well if the requests are small. If the number of clients is more on the network, there are more number of requests to the AS and TGS. This deteriorates the performance of AS and TGS. If the AS or TGS fails, the whole system fails. Therefore, single KDC cannot work properly for the whole network. This is just like to work in a small group which always give better performance. In the same way, the large Kerberos environment divides into distinct small realms. Each realm has its own authentication server and ticket-granting server. This helps to improve the performance and also avoid the failure problem due to single AS and TGS.

To allow the user from one Kerberos environment to access the service in another Kerberos environment, the user should first register with TGS in the service's realm. In some cases, if there are many Kerberos realms, it is difficult for the user to register each realm in every other realm. Instead of above method, there is a network of realms, so that, the user sometimes contact to the RTGS in one or more intermediate realm. These realms are called the *transited realms*. Also the names of the realms are included in the ticket. Due to this, the end service knows all of the intermediate realms that were transited, and can decide whether or not to accept the authentication. Kerberos version 4 had only peer to peer realm authentication while Kerberos version 5 support for scaling.

There are three types of cross-realm authentication based on trust: direct, transitive and hierarchical.

1. Direct relationship: It occurs when the KDC of one realm has direct trust in the KDC of another realm (Figure 9.17). It allows the users of the latter realm to access its services. This can be done by using a shared key.

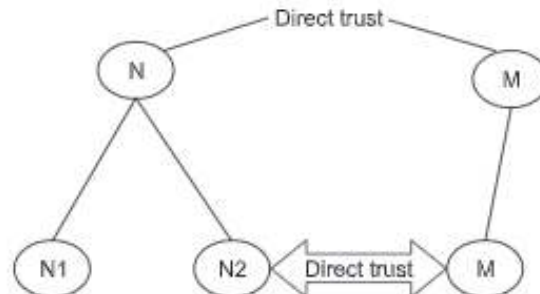


Figure 9.17 Direct trust.

2. Transitive trust relationship: In the above case, if the number of realms increases, the shared secret keys also increased. Transitivity trust relationship solves this problem. Here if realm A has a trust on realm B, and realm B has a trust on realm C, then realm A has a trust on realm C. This helps to reduce the number of shared secret keys required.

3. Hierarchical trust relationship: If, within organisations, the convention of naming realms uses upper case letters with the name of DNS domains and it belongs to a hierarchy, then Kerberos version 5 will support adjacent realms having a trust relationship.

Steps follows by cross-realm authentication are:

Step 1 Client requests to local KDC for a cross-realm ticket.

Step 2 Client submits a cross-realm ticket to the another KDC for a service ticket for the target service.

Step 3 Client submits the service ticket to the another AS server.

9.4.8 Kerberos and Public Key Cryptography

Kerberos uses symmetric key encryption techniques. But as we know that asymmetric key encryption techniques are more powerful than symmetric key encryption can be used for authentication, key distribution and non-repudiation. In this case, the public key is available for all the users in the Kerberos realms whereas the private key is known only by the owner user. KDC also not knows about the user's private key.

When the KDC generates the ticket after authentication, it encrypts the session key with the random key generated by the KDC. Again it encrypts it with user's public key. While at the user end the user can decrypt it using his private key and then obtains the random key, which he decrypts the message and get the session key. We can use one of the methods of public key cryptography as discussed earlier in this chapter.

9.4.9 Advantages of Kerberos

The Kerberos authentication model offers a number of advantages over more traditional authentication schemes.

1. User login on the client machine with his id and password. His passwords are never transmitted across the network in any form, i.e., encrypted or plaintext form. Only shared secret key is transmitted across the network in the encrypted form. This provides more security.
2. Client machine and server mutually authenticate each other during each communication.
3. Kerberos authentication model use timestamp and lifetime information in the ticket which limit the duration of their users' authentication. After specific lifetime, ticket is invalid for authentication.
4. Once the authentication of the user takes place, he can use different services across the Kerberos network without re-entering his personal information like password.

5. The shared secret may be used for encrypting the communication between the client and the service. This improves the security in Kerberos environment.
6. Kerberos is entirely based on open Internet standards.

9.4.10 Weaknesses of Kerberos

The weaknesses of Kerberos authentication model are as follows:

1. In Kerberos IV, DES algorithm is used for encryption. But DES is not a secure algorithm today. So in Kerberos V, 3DES or IDEA is used for encryption which is more secure.
2. For a multi-user client system, the Kerberos authentication scheme fails due to variety of ticket-stealing and replay attacks.
3. The AS model is vulnerable to brute-force attacks.

9.4.11 Attacks on Kerberos

Many attacks are successfully carried out against Kerberos. These attacks include: replay attacks, password guessing attacks, and inter-session chosen plaintext attacks.

1. Replay attacks: A replay attack occurs when an attacker captures a packet from the network and sends that packet to a service as a user of that service. When the packet is authenticated by the service, then the attacker can use the replay on behalf of other user and access other user's resources. Use of authenticator in Kerberos helps to prevent replay attack.

2. Password-guessing attacks: The reply to the request of a user, the ticket-granting ticket is sent in encrypted form. To decrypt the reply user's password is required. If user's password is not strong, then it is possible for the attacker to guess the password and attacker is able to decrypt the message.

3. Inter-session chosen plaintext attacks: As per Kerberos version 5 draft, inter-session chosen plaintext attacks are possible against it.

9.4.12 Applications and Limitations of Kerberos

To provide authentication, authorisation and confidentiality within a network or small set of networks, Kerberos is used. However, one cannot use Kerberos for generating digital signature.

The main assumptions about the Kerberos environment is that there should be trust on the hosts. But if the host is compromised, the attack can occur and the security of the Kerberos may be broken. Ticket is stored in host's cache may be used for such attack. But there is less possibility of such attacks. Dictionary attack is possible if the user password is guessable. Timestamp prevents such attacks. If the user needs more time for using the different services, then the small timestamp creates problem and again authentication is required. In Kerberos 4, for long processes, tickets having small timestamp can have this problem. Kerberos version 5 solved this problem by renewing the ticket after the end of time span allotted to a ticket.

9.4.13 Comparisons of Kerberos with SSL

Secure socket layer protocol (SSL) is also used for authentication. Table 9.7 shows the comparison between Kerberos and SSL.

Table 9.7 Comparison between Kerberos and SSL

SSL	Kerberos
Encryption is done using public key.	Encryption is done using private key.
Authentication is based on certificate.	Authentication is based on a trusted third party.
Ideal for secure communications with a large, variable user base that is not known in advance.	Ideal for networked environments where all services and users are known in advance.
Key revocation must be achieved either by sending certificates to all related servers or by having a centralised sever.	Key revocation can be achieved by disabling a user at the KDC.
Probability of cracking the certificates is more as it is stored in user's hard-disk.	Probability of cracking the password is less as it is not stored in written form.
One has to pay for the service as it is patented.	Freely available as Kerberos has open source.

9.5 X.509 AUTHENTICATION SERVICE

ITU-T recommends X.509, the authentication service. It specifies the authentication service for X.500 directories. It also specifies syntax for X.509 certificate. The first version of X.509 was published in 1988. The second version was published in 1993. The third version was proposed in 1994 and considered for approval in 1995. There were some security issues in the first two versions of X.509. These issues are addressed in version 3. Secret key or public key is used for directory authentication. The standard does not specify about the algorithms used for certificates but RSA is the most popular choice for this. In this, every user has a certificate whose validity depends on a chain of trust.

An X.509 certificate consists of the following fields:

1. *Version*: This gives information about the version of the X.509 standard applies to the certificate. Currently three versions of X.509 certificates are available. Version indicates the information available with the certificate.
2. *Serial number*: A serial number of the certificate distinguishes it from other certificates issued by the same party. Certificate's serial number is placed in a certificate revocation list (CRL) when a certificate is revoked.
3. *Signature*: This identifies the algorithm used to compute the signature on the certificate.
4. *Issuer name*: It is X.500 name of the entity who signed the certificate. Generally, it is a certificate authority (CA). Using issuer name certificate implies trusting the entity who signed the certificate.
5. *Validity*: Each certificate has its life span. Validity gives the information about this life span. The life span can be as short as a few seconds or almost as long as a century. This contains two types of information: a start date and

time, and an end date and time. The validity period chosen depends on a number of factors: the private key used or the amount one is willing to pay for a certificate. This is the expected period that entities can rely on the public value, if the associated private key has not been compromised.

6. *Subject*: The name of the user whose public key the certificate identifies. This name uses the X.500 standard, so it is intended to be unique across the Internet. This is the Distinguished Name (DN) of the entity, for example,

CN = Ram, OU = COEP, O = PIET, C = INDIA

CN: Subject's Common Name, OU: Organisational Unit, O: Organisation, and C: Country.

7. *Subject public key information*: This contains two types of information: the public key and an algorithm identifier which specifies which public key cryptosystem this key belongs to and any associated key parameters.
8. *Issuer unique identifier (versions 2 and 3 only)*: This is an optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.
9. *Subject unique identifier (versions 2 and 3 only)*: Each user has one number which is unique across the internet. This provides the unique identity of the user.
10. *Extensions*: It is a set of one or more extension fields. Extensions were added in version 3.
11. *Signature on the above fields*: Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifier.

Most generic version of X.509 is version 1. X.509 version 2 introduced the concept of subject and issuer unique identifiers are introduced in version 2. These concepts help to handle the possibility of reuse of subject and/or issuer names over time. Most certificate profile documents strongly recommend that names not be reused, and that certificate should not make use of unique identifiers. Version 2 certificates are not widely used.

The most recent version of X.509 is version 3. It supports the notion of extensions, whereby anyone can define an extension and include it in the certificate. The most common extension in use is: KeyUsage. It restricts extension in use is: AlternativeNames. It allows other identities to also be associated with this public key. e.g., DNS names, E-mail addresses, IP addresses. Extensions can be marked critical to indicate that the extension should be checked and enforced/used. For example, if a certificate has the KeyUsage extension marked critical and set to "keyCertSign" then if this certificate is presented during SSL communication, it should be rejected, as the certificate extension indicates that the associated private key should only be used for signing certificates and not for SSL use. This certificate is signed by the issuer to authenticate the binding between the subject (user's) name and the user's public key. The major difference between versions 2 and 3 is the addition of the extensions field. This field grants more flexibility as it can convey additional information. Standard

extensions include subject and issuer attributes, certification policy information, and key usage restrictions, among others. X.509 also defines syntax for certificate revocation lists (CRLs).

SUMMARY

Authentication is one of the key aspects of cryptography. It can be used to guarantee that communication end-points, i.e., sender and receiver are the parties who they claim. The main objectives of authentication requirement are to ensure that the claimants are really what they claim to be, and avoid compromising security to an impostor. Authorisation is the mechanism through which a system determines what level of access to a particular authenticated user should have. It is used to secure the resources controlled by the system. The password-based authentication method refers to a secret information which the user has to prove that he knows it. In two-factor authentication system, identification and authentication of the user take place in two different ways to establish his identity and privileges. Biometric authentication method uses thumb impression, iris or voices for authentication. Integrity of a message is checked by using the hash value or message digest calculated from the message. The hash value of the message is derived from the message which remains same if the message is unaltered. There are various algorithms available which can generate the hash value of the message. Two of them are message digest and secure hash algorithm.

MD5 generates message digest with a 128-bit hash value. The possible attacks against MD5 are possible. The more secure hash algorithm is SHA (secure hash algorithm).

Authentication of the user is very important to provide the security to any application. Under project Athena at MIT, an authentication system is developed known as Kerberos. The key innovation in Kerberos is that there is no need of user should reveal the secret. They can identify without revealing the secret. If the user from one Kerberos environment wants to use the services from other Kerberos environment, the authentication of that user should be done by the Kerberos which the user belongs is called cross-realm authentication.

EXERCISES

- 9.1 What is authentication? What are the objectives of authentication?
- 9.2 What is the difference between authentication and authorisation?
- 9.3 List the different methods of authentication. Explain each in brief.
- 9.4 Explain the password-based authentication method.
- 9.5 What are the guidelines for choosing a password or setting up a password?
- 9.6 Discuss the weaknesses of the password-based authentication method.
- 9.7 Explain the two-factor authentication method.
- 9.8 Compare the password-based authentication with the two-factor authentication methods.
- 9.9 Explain the fingerprint authentication method.
- 9.10 Discuss the weaknesses of the two-factor authentication method.
- 9.11 What is Kerberos? How TGS works?

CHAPTER 10

Digital Signatures

10.1 INTRODUCTION

In our day-to-day life, we use signature for various purposes. Every person has his own style of doing signature. Signature indicates the identity of the person. It helps in proving the authentication of a particular person. Use of signature in various forms for identification of documents is a practice from ancient times. In the middle age, wax imprint of insignia was used by the noblemen to seal the documents, which proved their authentication. In the history, every kingdom had its own flags, which was synonymous with signature, as it also used for authentication. Artists use to do their signatures on their paintings, which helps in authenticating the owner of the painting. Nowadays, after using the credit cards for the purpose of paying bills, we have to sign a slip which is supposed to be verified by the salesperson by comparing the same with the signature on the card. But due to advanced technology such as online trading, all these methods of authentication are of no use. The new method of authentication in electronic form has emerged. This new technique is called *digital signature*. Digital signature may be in the form of text, symbol, image or audio.

In our discussion, we use digital signature, a term encompassing only cryptographic signature. Digital signature is a strong method for authentication used today. This includes message authentication code (MAC), hash value of a message and digital pen pad devices. It also includes cryptographically-based signature protocols. All these different techniques ensure that no unauthorised person of some document or record could have done so. Therefore, digital signature is used for authentication of the message and the sender and to verify the integrity of the message. Many schemes have been proposed for generation of digital signature. Some of these schemes are patented. Some are freely available and some are failed to pass the security test. In today's world of electronic transaction, digital signature plays a major role in authentication. For example, one can fill his income tax return online using his digital signature, which avoids the use of paper and makes the process faster.

Asymmetric key encryption techniques and public key infrastructure are used for digital signature. Digital signature algorithms are generally divided into two parts—signing part and verification part. The first part, i.e., signing part allows the sender to create his digital signature. The second part of the signature is used by the receiver for verifying the signature after receiving the message. Digital signature is different from electronic signature. Electronic signature is not necessarily cryptographic-based for identification of the sender. Sometimes, phone, fax transmission and telephone addresses are treated as electronic signatures. Many electronic signatures use digital signature technology to ensure that the legal intent is also cryptographically secure.

The requirements for a digital signature are as follows:

1. It must be in the form of pattern of bits.
2. Information unique to the sender should be used for the generation of signature. This information helps in preventing forgery and denial.

Digital signature is used for communications to verify

1. Authentication of the sender
2. Integrity of the message received
3. Non-repudiation

We will discuss all these in brief below:

1. **Authentication:** Authentication, as we have discussed in the last chapter, is the most important part of security. There are two issues related to authentication—confidentiality and time-span. Confidentiality of the secret key is the most important issue as the security of the communication between two parties depends on it. So, we first take a look on this issue.

In the public key infrastructure, each user has two keys—private key and public key. The private key is used for encryption of the message by the sender. The decryption of the received message is done using sender's public key. But in this scheme, recipient is not sure whether the sender has himself signed the message or somebody else has used sender's private key for encrypting the message. This may happen if the private key of the sender is captured by the attacker and used it to send the message on the sender's name using his private key.

The second issue is time-span. This is related to the hazard of message replay attacks. There are replay attacks which allow the attackers to compromise a session key. In financial communication, confidentiality and integrity of the message is very important. Small time-span helps in protecting the replay attacks. If the time-span is more and the sender's key is compromised, then replay attack is possible.

2. **Integrity:** It helps in checking whether the message is the same message as sent by the sender or a modified message. This can be achieved by the use of message digest techniques. Encryption provides the confidentiality to the message and protects it from the cryptanalyst to read. If the key gets compromised, then it is possible for the cryptanalyst to modify the message. This can be done perhaps maliciously, without actually reading it. Integrity of the message can be verified with the help of message digest. Initially, the message digest or hash value of the message is generated using any message digest algorithm (such as MD5 or SHA-1). Then, the sender uses his private

key to encrypt the ciphertext and the message digest of the message. Then, the sender sends the encrypted message to the receiver. If the attacker or third party captures the message and modifies it, then due to hash value or message digest, the receiver can easily know about it because it is not possible to modify the message digest or hash value and no two messages can have the same hash value. The receiver decrypts the message digest using the public key of the sender. He also calculates the message digest of the message received. The receiver compares the two message digests. If the two messages digests match, then it proves that the message is unaltered and it is the same as sent by the sender.

3. Non-repudiation: Repudiation means that a person, who signs a document, is always able to disclaim a signature that has been credited to him or her. After receiving the message, the recipient asks the sender to attach his signature with the message so that it makes later repudiation more difficult. If the sender refuses that he is not the sender of the message, then the recipient can show the signed message to a third party (e.g., a court) to reinforce a claim that the message is sent by the signatories and not anybody else. However, if the sender's private key is compromised that means somebody else (i.e. attacker) know that key then all digital signatures generated using such private key may be generated by the attacker. In this case such digital signature cannot be helpful for non-repudiation. Notice that compromising the key is not the drawback of any cryptographic algorithm, but it is a human space and is unsolved. Digital signatures alone cannot provide inherent non-repudiation.

10.1.1 Implementation of Digital Signatures

Digital signature schemes have the following three algorithms:

1. A key generation algorithm
2. A signing algorithm
3. A verification algorithm

For example, user A wants to communicate with user B by sending him a message. User B wants to verify if the message has surely come from user A. User A signs her message and then sends to user B. For this, user A has generated the digital signature for the said message using her private key. Digital signature is in the form of a string of bits. After receiving the message, user B wants to know that whether the message has really been sent by user A or somebody else. For this, user B uses the verification algorithm. She uses the message and the digital signature as input for the verification algorithm and decrypts the message using user A's public key. If the digital signature matches, then user B can be confident that the message has really been sent by user A.

10.1.2 Association of Digital Signatures and Encryption

Message digest is used to generate the signature. The message digest (MD) is calculated from the plaintext or message. The message digests for two different messages are never same. The message digest is encrypted using user's private key. Then, the sender sends this encrypted message digest with the plaintext or message to the receiver.

The receiver calculates the message digest from the plaintext or message he received. He decrypts the encrypted message digest using the sender's public key. If both the MDs are not same, then the plaintext or message is modified after signing. This is explained in Figure 10.1.

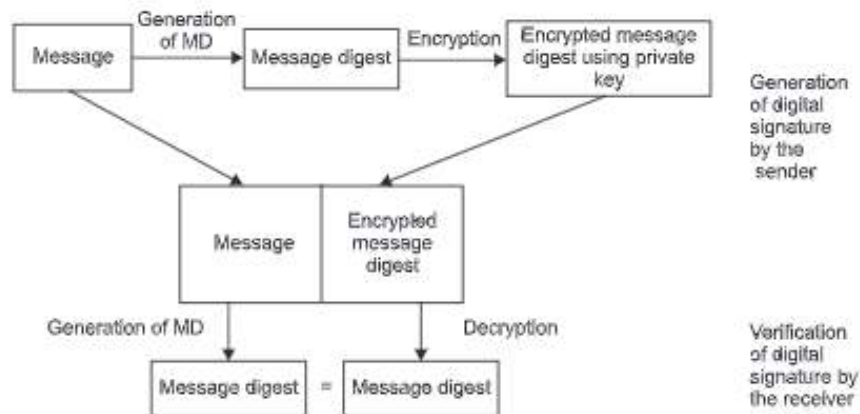


Figure 10.1 Generation and verification of digital signatures.

In digital signatures, the signature or hash value of the message is encrypted using encryption techniques. But the signature or hash value is small. In digital signature, the encryption techniques cannot be used for large messages. As compared to digital signature, more efficient methods are available. Signed document may be sent after encryption over an insecure channel like any ciphertext.

Generally, user A first uses a message digest algorithm like MD5 or SHA-1 and generates the hash value (MD) of the message. Then, he digitally signs the resulting MD (see Figure 10.1). An insecure MD can compromise the digital signature. For example, if it is possible to generate MD collisions (it is really difficult), it might be feasible to forge digital signatures.

In digital signature a message digest is signed instead of the complete message. This has the following advantages:

1. **Efficiency:** The size of digital signature is small, and thus, it improves the performance of the algorithm.
2. **The document is intended to be read by others:** The document includes degree certificates, birth date certificates, driving licenses of the users, rent agreement, contract agreement, etc. These documents are in the plaintext form. The accompanying digital signatures are in encrypted form and these can be used to verify that the documents are neither forged nor modified.
3. **Integrity:** Without the MD, the message 'to be signed' must be divided into a number of blocks. The blocks size should be smaller than the length of the private key. Then, each block is signed separately and sent to the receiver. The receiver can verify the individual block with the signature, but he would not be able to verify if all the blocks are received or not. If a block (or blocks) is lost during transmission,

the receiver cannot know it. To know about this loss, the message digest should be computed for a complete message and not for individual blocks.

10.1.3 Using Different Key Pairs for Signing and Encryption

Nowadays, digital signature is the most common tool for verification. In several countries, digital signature is accepted like traditional signature methods. These provisions mean that a person cannot deny the responsibility of the ownership of the message or the documents. Due to this, one should take care to protect one's message and digital signature through proper encryption techniques using proper public key and private key. This makes it difficult to the attacker to modify the message and also to change the digital signature. Digital signature can be protected using following measures.

1. Use of time stamp with digital signatures: There are different digital signature algorithms for the generation of digital signature. But these algorithms do not provide any information about the date and time of signature generation. If the user includes a date and time with the message, then also it is not sufficient to check whether the signature is generated by the user or the attacker. To avoid the misuse, as above mentioned, we can use time stamp in addition to digital signatures.

2. Additional security precautions: The security of digital signature is based on the private key used for encryption of the signature. If this private key is compromised, the attacker may be able to capture the digital signature. It can be stored on the user's computer/laptop or notebook, and protected by a password, but it has two disadvantages—only the documents on that particular computer/laptop or notebook can be signed and the private key is secure if the computer/laptop or notebook is secure from threat. To provide security to the local computers is difficult due to hardware or operating systems.

So, better option for the security of private key is store it on a smart card. Smart cards are designed so that the data remain safe if, by mistake, it is tampered. The message digest calculated from the message is sent to the smart card. The smart card is connected to some computer, which encrypts the message digest using the user's private key and sends it back. Personal identification number (PIN) is required to activate the smart card. If the smart card is stolen, and the user's private key is located on the smart card, then also it does not make any harm, as PIN is needed to activate the smart card. So, without PIN, digital signature cannot be generated.

3. Use of smart card readers: To activate the smart card, PIN is entered using a numeric keypad. Some card readers have their own keypad, while some card readers are integrated into a computer. Sometimes, the PIN may be captured by the attacker. Nowadays, attackers use scanner machine with smart card reader so that when the smart card is swiped through the smart card reader, complete data on the card is copied, and later on, it is used by the attacker.

10.2 ALGORITHMS FOR DIGITAL SIGNATURE

There are many digital signature algorithms. Some of them are listed below:

1. Full Domain Hash, RSA-PSS, etc. based on RSA

2. Digital Signature Algorithm (DSA)
3. Elliptic Curve Digital Signature Algorithm (ECDSA)
4. ElGamal signature scheme
5. Undeniable signature
6. SHA (typically SHA-1) with RSA
7. Rabin signature algorithm
8. Pointcheval–Stern signature algorithm
9. Schnorr signature

10.2.1 Digital Signature Algorithm (DSA)

To generate the digital signature, the most widely used algorithm is Digital Signature Algorithm (DSA). In 1991, the National Institute of Standards and Technology (NIST) proposed the Digital Signature Algorithm to use it in their Digital Signature Standard (DSS). DSA was adopted as a standard in 1994. In 1996, a minor revision in DSA was issued. DSA standard was expanded further in 2000. DSA generates the message digest of length 160 bits. The algorithm is split into three steps—key generation, signing and verification. Now, we discuss all these steps of DSA.

Key Generation

1. Select a prime number q , which is 160-bit long. Select a prime number p , which is L -bit long such that $512 \leq L \leq 1024$, and L is divisible by 64, and $p = qk + 1$ for some integer number k .
2. Select h , where $1 < h < p - 1$ and generate g such that $g = h^k \bmod p > 1$.
3. Select private key x , where $0 < x < q$ and compute $y = g^x \bmod p$ (x should be kept secret).
4. The public key is (p, q, g, y) .
5. p, q, g should be shared by different users.

Signature Generation

1. Select a secret integer number k , where $0 < k < q$.
2. Compute r and s such that

$$r = (g^k \bmod p) \bmod q \text{ and}$$

$$s = (k^{-1}(\text{SHA-1}(m) + x * r)) \bmod q,$$

where $\text{SHA-1}(m)$ is the message digest of the message m using SHA-1 algorithm.

3. If $r = 0$ or $s = 0$, repeat the above procedure.
4. The digital signature for the message m is (r, s) .

Signature Verification

1. If either $0 < r < q$ or $0 < s < q$ is not satisfied, reject the signature.
2. Compute $w = (s)^{-1} \bmod q$.

3. Compute $u_1 = (\text{SHA-1}(m) * w) \bmod q$.
4. Compute $u_2 = (r * w) \bmod q$.
5. Compute $v = ((g^{u_1} * y^{u_2}) \bmod p) \bmod q$.
6. If $v = r$, then the signature is valid.

Correctness: The DSA scheme is correct, as the verifier always accepts genuine signatures. This can be illustrated as below:

We know that $g = h^k \bmod p$ and suppose $g^a \bmod p = h^{ak} \bmod p$. Applying Fermat's little theorem, we get $g^a = h^{ak} = h^{p-1} = 1 \pmod{p}$. Since $g > 1$ and q is prime, it follows that g has order q .

The signer computes

$$s = (k^{-1}(\text{SHA-1}(m) + x * r)) \bmod q$$

Thus,

$$\begin{aligned} k &= \text{SHA-1}(m)s^{-1} + xrs^{-1} \\ &= \text{SHA-1}(m)w + xrw \pmod{q} \end{aligned}$$

Since g has order q , we have

$$\begin{aligned} g^k &= g^{\text{SHA-1}(m)w} g^{xrw} \\ &= g^{\text{SHA-1}(m)w} y^{rw} \\ &= g^{u_1} g^{u_2} \pmod{p} \end{aligned}$$

The correctness of DSA algorithm is checked as below:

$$r = (g^k \bmod p) \bmod q = (g^{u_1} g^{u_2} \bmod p) \bmod q = v$$

Digital signature algorithms have a number of prior requirements (mentioned below). Without these requirements signatures do not have any meaning.

1. Quality algorithms: Quality of the digital algorithm also depends on the quality of public key algorithms used.

2. Implementations: Implementation of any algorithm is very important. If a very good algorithm is implemented with some mistake, it will never work properly.

3. Private key: The security of any algorithm depends on the secrecy of private key. If the private key of a user compromises (known by the attacker), then the attacker also generates the same digital signature irrespective of how strong the algorithm can be used for the generation of signature.

4. Distribution of public keys: In public key cryptography, the distribution of public key is also very important. For this, there exist different key distribution methods. The distribution of public key is commonly done using a public key infrastructure (PKI) and certificate authority (CA). During distribution, if the public key of a user is given to non-authenticate user, i.e., attacker, then it compromises with the security.

5. Handling of signature protocol: Signature algorithms should be used properly by the users.

If all the above conditions are satisfied by any digital signature scheme, then the digital signature will have the information regarding the sender of the message and user's consent about the message.

10.2.2 ElGamal Signature

One more digital signature scheme described by Taher ElGamal in 1984 is the ElGamal signature. It is based on discrete logarithms. A variant of ElGamal signature is the Digital Signature Algorithm, which is much more widely used. There are several other variants of ElGamal signature. The ElGamal signature scheme is different from ElGamal encryption technique. The main feature of ElGamal signature scheme is that there are many different signatures, which are valid for a given message.

The ElGamal signature scheme allows a verifier that he can confirm the authenticity of a message m sent by the signer to him over an insecure channel.

The user should select parameters p and g as below:

1. Select a large prime p such that the discrete log problem in F_p is infeasible.
2. Recommended size of p is 1024 bits.
3. g is generator of the multiplicative group F_p .

Key Generation

1. Select private key x as $1 < x < p - 1$.
2. Compute $y = g^x \pmod p$.
3. Public key is (p, g, y) .
4. Private key is x . (Recommended size of x is 160 bits.)

Signature Generation

The signature for message m is calculated as below:

1. Select a secret integer number k , where $0 < k < p - 1$ and $\gcd(k, p - 1) = 1$.
2. Compute $r = g^k \pmod p$.
3. Compute hash value of the message as $s = (H(m) - xr)k^{-1} \pmod{p - 1}$.
4. If $s = 0$, repeat the above procedure.
5. The digital signature is (r, s) .

Signature Verification

The digital signature for the message m is (r, s) and it is verified as follows:

$$0 < r < p \text{ and } 0 < s < p - 1$$

$$G^{H(m)} = y^r r^s \pmod p$$

If all conditions are satisfied, the verifier accepts a signature and rejects it otherwise.

Correctness: The above scheme is correct, as the verifier always accepts genuine signatures. The signature generation implies

$$H(m) = xr + sk \pmod{p - 1}$$

Applying Fermat's little theorem,

$$\begin{aligned} G^{H(m)} &= g^{xr} g^{ks} \\ &= (g^x)^r (g^k)^s \\ &= (y)^r (r)^s \pmod p \end{aligned}$$

Security: An attacker can forge signatures either by capturing the signer's private key x or by finding collisions in the hash function $H(m) = H(M) \pmod{p-1}$. Capturing either of this information is believed to be difficult.

For each signature, the signer must select the value of k very carefully. Also, he should keep the value of k secret. If the value of k compromises, an attacker may be able to deduce the private key x with reduced difficulty. The practical attack is possible using the value of k . If two messages are sent using the same value of k and the same key, then an attacker can compute x directly.

10.2.3 Elliptic Curve Digital Signature Algorithm (ECDSA)

A variant of the Digital Signature Algorithm (DSA) is Elliptic Curve Digital Signature Algorithm (ECDSA). It is based on elliptic curve groups. The use of elliptic curve provides smaller key sizes for the same security level, with roughly, the same execution time. It also generates the signature of exactly the same size.

Signature Generation

Suppose there are two users A and B, user A wants to send a digitally signed message m to user B. Both should agree for some curve parameters such as $q, FR, \alpha, b, G, n, h$. Also, user A selects two keys such as private key d_A and a public key Q_A . These keys are suitable for elliptic curve cryptography.

User A signs a message m as below:

1. Calculate $e = H(m)$, where H is a message digest or hash value of message m . MD can be calculated using secure hash algorithm SHA-1.
2. Select a secret integer number k , where k is from $1 \leq k \leq n-1$.
3. Calculate $kG = (x_1, y_1)$.
4. Calculate $r = x_1 \pmod{n}$. If $r = 0$, go to step 2.
5. Calculate $s = k^{-1}(e + d_A r) \pmod{n}$. If $s = 0$, go to step 2.
6. The signature of message m is (r, s) .

Signature Verification

Using user A's public key Q_A , user B verifies user A's signature as below:

If r and s are integers in $[1, n-1]$, signature is said to be valid, otherwise it is invalid.

1. Compute $e = H(m)$, where H is the same hash function used in the signature generation.
2. Compute $w = s^{-1} \pmod{n}$.
3. Compute $u_1 = ew \pmod{n}$ and $u_2 = rw \pmod{n}$.
4. Compute $(x_1, y_1) = u_1G + u_2Q_A$.
5. The signature is valid if $x_1 = r \pmod{n}$, otherwise invalid.

10.3 DIGITAL SIGNATURE STANDARD (DSS)

For performing digital signature of any message or documents, some standard is required. This standard is called *Digital Signature Standard (DSS)*. The NIST published this standard (i.e., DSS) in 1991. One should note that DSS is a standard, whereas DSA is an algorithm. It has been developed for performing digital signatures. For different digital applications, this standard specifies an appropriate Digital Signature Algorithm (DSA). As per this standard, the message digest of a document is calculated using secure hash algorithm-1 (SHA-1).

Using this algorithm, a signature is generated, which includes a pair of large number which is represented in the form of strings of binary digits. A set of rules and parameters are used to compute the signature. The DSA algorithm has three parts—key generation, signature generation and signature verification. Using user's private key, a signature is generated. Sender's public key is used for the verification of signature. Anyone can verify the signature of a sender.

In signature generation, the message digest of a message is computed using secure hash algorithm, as shown in Figure 10.2. Then, the signature is generated using this message digest. After that, the sender sends the signature with the message to the receiver. The receiver is first intended to verify the signature using sender's public key. The recipient should use the same hash function to calculate the message digest of the received message. The hash function is specified in a separate standard, the Secure Hash Standard (SHS), FIPS 180.

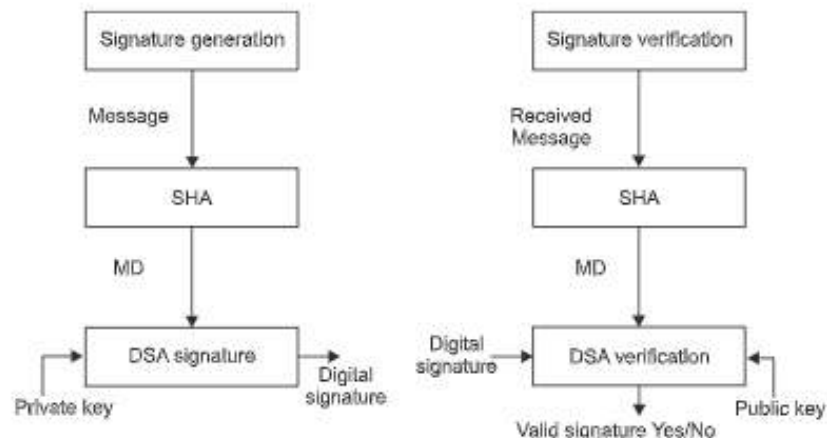


Figure 10.2 Using the SHA with the DSA.

10.3.1 Applications of Digital Signature

Digital signature provides the integrity and the authentication. Third party also verifies the authentication of the sender using digital signature. The applications of digital signature is intended in various areas such as in e-mail, online transactions,

e-commerce, e-billing, interchanging the data electronically, and applications where data integration, and authentication of data origin are required. Nowadays, e-registration of flats and other properties also use digital signature for authentications. Even one can file income tax return using his/her digital signature.

10.4 AUTHENTICATION PROTOCOLS

There are various authentication protocols. Some of them are listed below:

1. **Challenge Handshake Authentication Protocol (CHAP):** It is a three-way handshake protocol and it is much secure than PAP.
2. **Extensible Authentication Protocol (EAP):** It is used as a dial-in between the client and the server. It is used to determine what authentication protocol will be used.
3. **Password Authentication Protocol (PAP):** It is a two-way handshake protocol. It is used with point to point protocol (PPP). It uses a plaintext password like older SLIP systems. It is not secure.
4. **Shiva PAP (SPAP):** Only NT RAS server supports this for clients dialing in.
5. **Data Encryption Standard (DES):** It is used for older clients and servers.
6. **Remote Authentication Dial-In User Service (RADIUS):** It is used in organisation's network to authenticate users dialing in remotely to servers.
7. **S/Key:** It is a RFC 2289 Authentication Protocol. It is secure against replays attacks. It is a one-time password system.
8. **Terminal Access Controller Access Control System (TACACS):** It is used for authentication, accounting and authorisation.
9. **MS-CHAP (MD4):** It is used to authenticate remote workstations and developed by Microsoft. It uses MD4 for computing the message digest and DES for encryption.
10. **SKID-SKID2 and SKID3:** It uses symmetric encryption technique. Privacy of the users are not maintained and a man-in-the-middle attack is possible against it.

SUMMARY

Digital signature is a strong method for authentication used today. Digital signature includes message authentication codes (MAC), hash value of a message and digital pen pad devices. The Digital Signature Algorithm (DSA) is the standard algorithm for digital signatures. The algorithm is split into three parts—key generation, signing and verification. One more digital signature scheme described by Taher ElGamal in 1984 is the ElGamal signature. It is based on discrete logarithms. A variant of the Digital Signature Algorithm (DSA) is Elliptic Curve Digital Signature Algorithm (ECDSA). It is based on elliptic curve groups. The use of elliptic curve provides smaller key sizes for the same security level, with roughly, the same execution time. It also generates the signature of exactly the same size. The NIST published the standard for digital signature, called *DSS*, in 1991. One should note that *DSS* is a standard, whereas *DSA*

CHAPTER 11

Electronic Mail Security

11.1 INTRODUCTION

We learn about various encryption techniques which secure data. Most of today's communication is performed through e-mail. The message is in the form of plaintext. So, if somebody captures this message, it may cause harm to the sender and the recipient. So, there is a need to provide security to this message. Pretty Good Privacy (PGP) is used to protect the message, which we can send securely through e-mail. In this chapter, we are going to discuss Pretty Good Privacy (PGP), Multipurpose Internet Mail Extensions (MIME) and Secure/Multipurpose Internet Mail Extensions (S/MIME).

11.2 PRETTY GOOD PRIVACY (PGP)

Today most of the personal communication is done using e-mail through internet. We know that internet is very insecure medium of transmission. If we send the message as plaintext using e-mail, the attacker can easily capture our message. So, there is a need to protect our message. One can use Pretty Good Privacy (PGP) to protect the message. It is used to encrypt and decrypt e-mail message over the Internet. PGP uses a combination of symmetric and asymmetric encryption algorithms for encryption and decryption of the message. It was developed in the late 1980s and early 1990s by Phil Zimmermann. An encrypted digital signature can be sent to the receiver using PGP. This helps the receiver in verifying the sender's identity and also in checking the integrity of the message. PGP is the most popular program used by individuals and for commercial applications. It is available as freeware and also in a low-cost commercial version. For e-mail security, now PGP has become a de facto standard. We can store our files in encrypted form using PGP to protect it from the attackers or unauthorised person. In PGP, each user has his certificate, but there is no certificate authority. The authentication of certificate is verified by the other users.

PGP is used for authentication of the sender. It is also used for encrypting documents. This secures the documents, as only intended recipients can decrypt the files. The public keys of the users are distributed so that the other users can use these keys for communication to a particular user. The sender encrypts the message using the public key of the recipient. This encrypted message is decrypted using the private key of the recipient's private key. The private key is a secret key and known to the owner only. The private key is used for encrypting the digital signature of the user. In this, the recipient uses sender's public key for decryption. For encrypting e-mails, PGP is useful. Any individual can use PGP to protect his/her message. PGP helps in providing privacy to the people.

11.2.1 Need of PGP

Privacy is the most important factor, which is needed by every user. PGP empowers users to maintain their privacy by their own. Some important conditions are given below in which encryption is required to protect the data:

1. Suppose two friends are exchanging encrypted messages via e-mail. For these friends, the messages may seem only passionate. If the messages are not encrypted, other people may read them, and for those people, they may be erotic messages. In such case, if the messages are encrypted, other people may not be able to read them.
2. The development team of a company's new product definitely wants to keep all the communication between the members of the team to be secret. For this, the e-mail regarding the product shared among them should be encrypted. This communication should be kept secret not only till the product becomes patented but also till the launching of the product in the market. In this case, not only the e-mails but also the data files need to be encrypted.
3. A wife and husband share a laptop/notepad and also their personal e-mail accounts. They also share e-mail passwords with each other. Now, they can access each other's mail accounts. Now, suppose as the wife can access husband's e-mail account, her husband keeps his passphrase secret. He wants to make her happy by giving a surprise birthday party. But he is anxious that she might know about the party. So, he uses encrypted e-mails for planning the party.
4. On the computer system at our office, if we store some important personal data, credit card numbers, personal scanned documents, family photos, etc., then this data should be kept in encrypted form.

PGP should be used in the following cases:

1. Financial data which we want to send through e-mail
2. Some data related to trade
3. Personal information
4. Information related to some product before its commercial launching

These are very limited examples. The list includes any information which causes us any type of loss. We use locker to protect our valuables; in the same way, PGP

provides security to the documents electronically. Nothing is wrong if someone wants to secure his personal information. Taking proper care, PGP is stronger than your locker.

11.2.2 Working of PGP

PGP provides some important services such as:

1. Authentication
2. Confidentiality
3. Confidentiality and authentication
4. Compression
5. E-mail compatibility
6. Segmentation and reassembly

Now, we will see in brief how these services work.

1. Authentication: When the sender sends message, the receiver wants to authenticate the message. So, the sender uses digital signature for authentication. The authentication carries out as given below:

- (a) The sender creates the message.
- (b) The message digest or hash value of the message is generated by the sender.
- (c) Message digest is encrypted using sender's private key. The encrypted message digest is the digital signature. RSA algorithm is used for this encryption. The message is attached to the digital signature.
- (d) The receiver decrypts the message using sender's public key.
- (e) The message digest or hash value of the received message is generated by the receiver.
- (f) If the two message digests match, then the received message is accepted as authentic.

2. Confidentiality: PGP is also used to provide confidentiality. For providing confidentiality to the message, any encryption algorithm can be used. PGP suggests CAST-128 or IDEA or 3DES for encryption. These algorithms work on the block of fixed size. But the length of the message is an arbitrary and not the multiple of the size of the block. So there may be less number of bits in the last block, the cipher feedback mode is used. The PGP uses a symmetric key encryption. The session key is bound to the message and transmitted with it. The confidentiality is provided as given below:

- (a) The sender creates the message and a 128-bit session key.
- (b) Using CAST-128 encryption algorithm and a session key, the message is encrypted.
- (c) The session key is encrypted using receiver's public key. For this, RSA algorithm is used. The encrypted session key is prepended to the message.
- (d) The receiver recovers the session key by decrypting the message using his private key. For this, he uses RSA algorithm.
- (e) Using session key, the message is decrypted.

PGP provides Diffie–Hellman algorithm as an option to RSA algorithm for encryption.

3. Confidentiality and authentication: To provide more security, we can combine authentication and confidentiality.

- (a) The sender creates the message and a 128-bit session key.
- (b) The message digest or hash value of the message is generated using SHA-1 by the sender.
- (c) Message digest is encrypted using sender's private key. The encrypted message digest is the digital signature. RSA algorithm is used for this encryption. The message is attached to the digital signature.
- (d) The session key is encrypted using receiver's public key. For this, RSA algorithm is used. The encrypted session key is prepended to the message.
- (e) The receiver decrypts the message using his private key and recovers the session key. For this, he uses RSA algorithm.
- (f) The receiver decrypts the message using session key.
- (g) The receiver decrypts the message using sender's public key.
- (h) The message digest or hash value of the received message is generated by the receiver.
- (i) If the two message digests match, then the received message is accepted as authentic.

4. Compression: Compression means compact or reduce the size of the message. To reduce the size of the final message, PGP uses compression technique. Compression of the message is done after the generation of digital signature, but before encryption of the message. Compression provides advantages like saving of space for transmission and storage. The digital signature is generated before compression so that one can store only the uncompressed message, together with the digital signature for future verification. Encryption of the message is done after compression. This provides strong security because it has less redundancy than the original message. So, the cryptanalysis of the message is more difficult.

5. E-mail compatibility: When the PGP is used, at least the part of the block that needs encryption should be encrypted. For example, if authentication is needed, then the message digest is encrypted. If confidentiality is needed, the message and digital signature both are encrypted. Thus, the resulting block consists of a stream of arbitrary 8-bit octets. For the e-mail, where the block must have ASCII characters, PGP provides the service of converting the raw 8-bit binary stream into ASCII. This conversion is done using radix-64. Due to this conversion, the size of a message is increased by 33%. Generally, the sizes of digital signature and message digest are very small and PGP uses compression technique which reduces the size of the message. This compensates the above problem. Also, radix-64 converts the input, irrespective of the contents.

6. Segmentation and reassembly: We cannot send large messages through e-mail. If the message is very large, we have to divide the message into smaller parts and then e-mail the smaller segments separately. PGP automatically subdivides the larger message into smaller parts so that it can be sent through e-mail. When all the processes i.e., digital signature, message digest or compression on the original message are completed, then before sending it through e-mail segmentation is done on the message, so that the message digest or the key is in only one segment, other segments do not contain it. At the receiving end, PGP collects all the e-mail headers and reassembles

the original segments or parts of the message before decryption, compression, etc. The transmission diagram for PGP message is shown in Figure 11.1. The reception diagram for PGP message is shown in Figure 11.2.

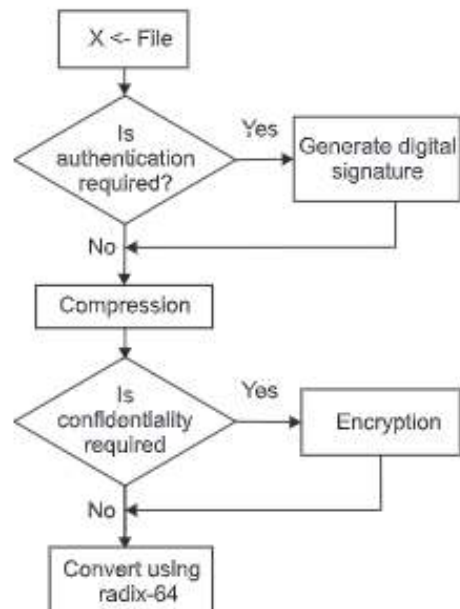


Figure 11.1 Generic transmission diagram of PGP messages.

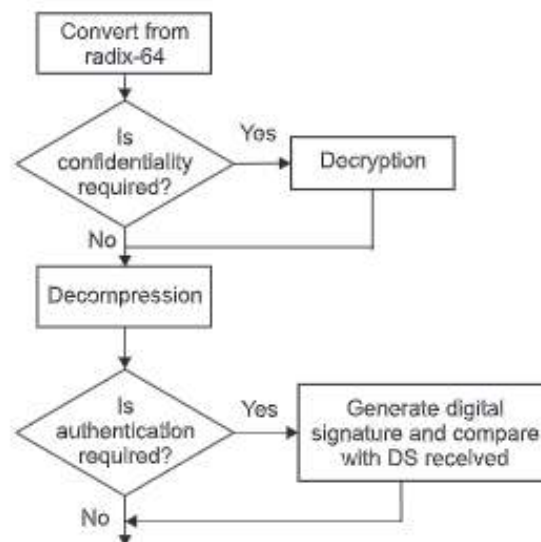


Figure 11.2 Generic reception diagram of PGP messages.

Radix-64 Conversion

We have to follow the following steps for radix conversion:

Step 1 Read the binary number. Then, split it into blocks of size 24-bits (3 bytes).

Step 2 Each block of 24-bit is then divided into four groups of 6-bit each.

Step 3 Each group will have a value between 0 and 2^6-1 (=63).

Step 4 This value is encoded into a printable character.

Table 11.1 gives the radix-64 encoding

Table 11.1 Radix-64 encoding

6-bit value	Character encoding	6-bit value	Character encoding	6-bit value	Character encoding	6-bit value	Character encoding
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

Processing the message includes various steps. First, the message is encrypted. Then, ZIP algorithm is used to compress the message. The compressed message is converted to radix-64 format. Converting ASCII to radix-64 format is done by substituting 6-bit values for 8-bit ASCII characters. At a time, three bytes are converted into four characters. If the last group size is less than three bytes, append zeroes at the end of the 6-bit value. This is called *padding*. "=" sign is used if the padding bits are equal to one character. Following examples explain this process:

1. If there are 3 bytes in the last group, no processing is required.
2. If there are 2 bytes in the last group, the first two groups are processed in the normal way. The two zero-value bits are added to the third (incomplete) group and are converted. A pad character (=) is added to the output.
3. If the last data group has 1 byte, then the first 6-bit group is processed in the normal way. The four zero-value bits are added to second (incomplete) data group and are converted. Two pad characters (=) are added to the output. Finally, a 24-bit checksum is computed to detect errors after conversion. The checksum is lead by the equals sign (=).

Format of PGP message is shown in Figure 11.3.

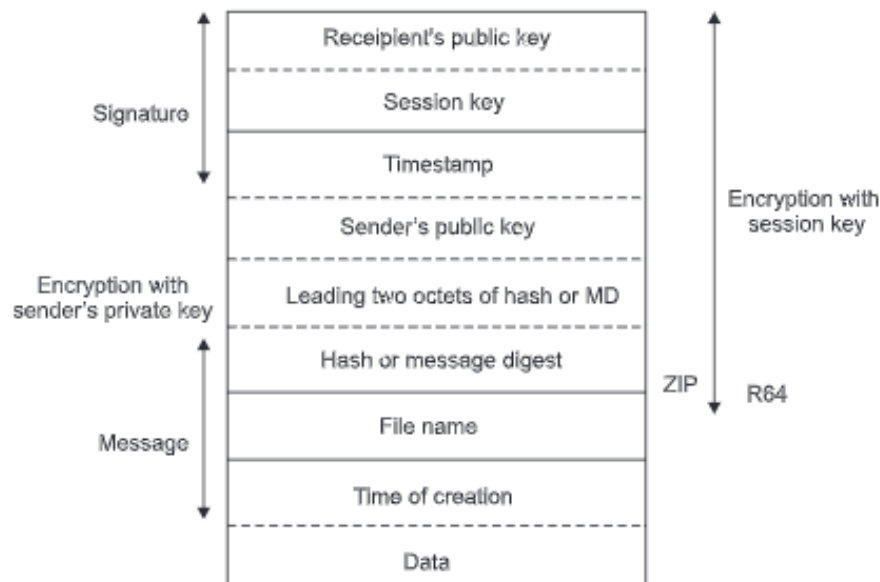


Figure 11.3 Format of PGP message

11.2.3 PGP Encryption Applications

The messages sent by e-mail are encrypted using PGP. Since 2002, the applications of PGP have been diversified. Digital signatures, laptop hard disk encryption, file and folder security, protection, batch file transfer encryption, and protection for files and folders stored on network servers can be encrypted using PGP.

11.2.4 PGP: Backdoors and Key Escrow

Backdoors

PGP has many features, which may affect its performance. One of such features is backdoor. It allows an outside user to decrypt the encrypted message. How does the sender know whether someone has intentionally planted his own security hole in PGP? Suppose that the government induces PGP Corporation to insert a backdoor feature, which allows the police department to decrypt public messages and files easily.

Now, when somebody purchases the PGP product for which the source code is unavailable, then the outside inspection of the product will be impossible for the user. Due to this, old users of PGP would lose confidence in newer versions of the product. So, to restore confidence in the user, the PGP Corporation makes an attempt. The source code for various versions of PGP is available publically. Cryptanalyst can attack suddenly on each new version and examine the source code carefully. After finding out that the source code contains no backdoors, he can compile that source code. He

compares the result with the executable program obtained from the PGP Corporation. Alarm has not been raised for a single time, which indicates that intentionally, no weakness has been inserted into PGP by the PGP Corporation. The same is true for other versions of the PGP.

Every time, source code examination does not prove fruitful. It has its limitation. This is because each newer version of PGP has much larger source code than its predecessors. This makes code examination more difficult. To overcome this drawback, digital signature is used by PGP with its own key.

Key Escrow

It is a key exchange process. In this, an encryption key is stored by a third party. This system provides the backup for the keys. A key, which is lost by its original user, may be used to decrypt the ciphertext. This allows the original message be retained in its original form. But key escrow system is sometimes risky, as third party is involved for storing the keys.

PGP makes use of four types of keys:

1. One-time session symmetric keys
2. Public keys
3. Private keys
4. Passphrase-based symmetric keys

With respect to these keys, three separate requirements can be identified.

1. Session key should be generated randomly and difficult to guess.
2. User can use multiple pairs of public-key/private-key. As a result, there is not one-to-one mapping between the users and their public keys. Thus, it is difficult to identify particular keys.
3. Each PGP entity should maintain a directory of its own public/private key pairs as well as a directory of public keys of other users.

Problems

Backdoors and key escrow have certain problems, which are discussed below:

1. The PGP design and algorithms are available to all. If the commercial sources have backdoors, then a cryptanalyst can simply implement his own PGP. If the personal PGP is made illegal, it does not mean that the cryptanalyst will bother.
2. Non-backdoor versions of PGP will remain available at the border of the country.
3. There is no guarantee that the government keeps our data secret after using backdoors and key escrow.

Suppose the government implements a backdoor. It makes mandatory to use the additional decryption key (ADK), which belongs to the government for all PGP users. If ADK is compromised or leaked, it affects all the confidential data of individuals. If the private key of the individual user is compromised or leaked, it will affect the

security of that individual, but if the ADK is compromised, it will affect the security of everyone.

11.2.5 PGP Security Quality

PGP is more secure and is not possible for the cryptanalyst to break it by any mean. SSL secures our data in transit only, whereas PGP protects the data while in storage.

The security of PGP depends on the assumption that the PGP algorithms used are difficult for cryptanalysis using current hardware and techniques. In the original version of PGP, the session key is encrypted using RSA algorithm. Cryptanalyst is able to break the RSA algorithm, then he captures the session key. In the same way IDEA is used for symmetric encryption in PGP. After some time, if the security flaws are detected with IDEA algorithm one cannot use IDEA algorithm for encryption in PGP. So, to add more security, some additional encryption algorithm can be used to make it difficult for cryptanalyst to break it and provide more security.

11.3 MIME

MIME is defined by an Internet standard document, called RFC1521. *MIME* stands for Multipurpose Internet Mail Extensions. *MIME* adds some additional fields to the old standard. It provides new standard for e-mail. This includes some new fields for headers of the message. Initially, e-mail systems are used to send only text messages. However, nowadays, use of e-mail is done not only to send the text messages but also the audio, video and multimedia files. Also, e-mail is used to send images, documents in various formats, etc. Large message can split into different parts while sending by e-mail. Each part has its header, which gives information about that part.

Using *MIME*, messages can be sent through e-mail. The messages may contain:

1. Text with additional fields such as special headers and formatted sections
2. A number of parts
3. Any size
4. Other than English characters
5. Text having different fonts
6. Binary files
7. Images, audio, video and multimedia files
8. Application specific files

The improvement in *MIME* is done to provide more security to the message. This new version of *MIME* is known as *Secure MIME* or *SMIME*.

11.3.1 MIME Headers

There are different headers provided by the e-mail system. These headers include From, To, Date, Subject, etc. Some new header fields of *MIME* are as given below:

1. ***MIME-Version:*** Time to time, standards are updated and new standards are introduced. These new standards should support the old standards. The same is

happened in case of MIME. MIME takes care of the old standard by making itself compatible with the older Internet mail standards. It has added some new fields. One such field is a version number. It is used to declare that a message matches with the MIME standard. For a single message, only one version field is used and it lies at the top of a message. If the message has more than one part, then in that case also, header has only one version field.

2. Content-Type: Another new field has been added, called *Content-Type*. The message contains data having different types and subtypes. This new field, i.e., Content-Type specifies the type and subtype of data. It also provides information about the encoding of data.

Content-Type header field is written as

```
Content-Type: = type "/" subtype [ ";" parameter]
```

Subtypes of Content-Type header are as follows:

- *Value text:* Textual information in the message is represented using value text field. Types of value text are plaintext, and enriched text. Plaintext allows a string of ASCII characters and enriched text allows greater formatting flexibility.
- *Value multipart:* If the message has more than one part, then the field value multipart indicates the information that there are multiple parts. It can be used to combine several parts of different types of data into a single message.
- *Value application:* It is used for transmission of binary or application data.
- *Value video:* This is used for transmitting video data.
- *Value image:* It is used for transmitting still images.
- *Value audio:* It is used for transmitting audio or sound data.
- *Value message:* This is used for encapsulating a message.

3. Transfer encoding: It specifies the encoding mechanism for the message.

4. ID: This refers to unique identification of entities with reference to multiple contexts.

5. Description: It gives more information about the data (video data) in a message.

In the MIME standard, the content-type values, subtypes and parameter names are case-insensitive; however, many parameter values are case-sensitive.

MIME standard is extensible such that new header fields and parameters can be added with time. Other MIME fields are likely to have new values defined over time. All such enhancements in MIME should be introduced in a proper manner. MIME has a registration process that uses the Internet Assigned Numbers Authority (IANA) as a central authority for registration.

The MIME has defined following content types:

1. **Audio:** This content type indicates that the data is audio data.
 - (a) *Audio/Basic:* The audio is encoded using 8-bit ISDN u-law. The requirement for this type is a single channel and a sample rate of 8000 Hz.

2. **Image:** This content type indicates that the data is image data. The requirement for this type is a display device. The image may be in *jpeg* or *gif* format.

3. **Message:** This content type indicates the complete message.

- (a) *Message/RFC822:* It indicates message with an RFC822 syntax. It is used for transmitting the message digest of a message.
- (b) *Message/Partial:* It indicates a segment of message. It allows to transmit the message in parts or segments.

There are three parameters in this content type—*ID*, *Number* and *Total*.

- (i) *ID* indicates the unique number for each message. For a message, there is only one ID. All the segments of a message have the same ID.
 - (ii) When the message is split into number of parts/segments, each part/segment has a *number*, which indicates the sequence of that part/segment. It is an integer value.
 - (iii) *Total* indicates the total number of parts or segments in a message. It is also an integer value. It is compulsory for the final part/segment, whereas for other parts it is optional.
- (c) *Message/External-body:* This content type is used to show the reference and not the actual message. It indicates the parameters for accessing the external data.

For a message, a complete message or part/segment of message or external body consists of a header, a blank line and the message header for complete message. To end the message header, another blank line should be added.

Consider the following message:

```
Content-Type: message/external-body;
Access-Type=local;
Name="/c/abc/photo.gif"
```

```
Content-Type: image/gif (gif shows the file is *.gif format)
Content-ID: 12345
Content-Transfer-Encoding: binary
```

Access-Type is mandatory to mention; other fields are optional or mandatory depending on the value of *Access-Type*.

In addition to *Access-Type*, some parameters are optional, which are given below:

- (i) *Expiration date* gives the last date for valid data. After this date, there is no guarantee about the existence of data.
- (ii) The *size* of the data is specified.

4. **Multipart:** If the message is split into number of parts and each part is having its own data type, then this parameter is used. It gives the information about the start and end of each part using an *encapsulation boundary*; the end of last part is followed by a closing boundary.

A multipart Content-Type header field syntax is as follows:

```
Content-Type: multipart/mixed; boundary= abc09qw56as
```

The above syntax shows that a message has several parts. Each part has identical structure like to an RFC822 message. The header may be empty and each part is preceded by—

```
abc09qw56as
```

The closing boundary is shown as—

```
abc09qw56as
```

There are spaces to add more additional information before the first encapsulation boundary and following the final boundary. These spaces are often blank. Anything appearing before the first or after the last boundary is ignored.

Consider a multipart message, which has two parts. Both the parts are text, one is explicitly typed and the other is implicitly typed. The message syntax is shown below:

```
From: Ram Vyasana
To: Shyam Dhur
Subject: Hellow Message
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="simple boundary"
```

This is called the *preamble* of a message and may be ignored.

```
simple boundary
This is implicitly typed ASCII text.
simple boundary
Content-Type: text/plain; charset= ASCII
to end this part use
-simple boundary-
```

One can use a Content-Type of multipart in a body part within another multipart entity. Care should be taken to use a different boundary delimiter. In some cases, multipart Content-Type with only a single body part may be useful.

5. Multipart/Mixed: It is used to view multiple parts of a message sequentially.

Multipart/Alternative: There is an “alternative” version of the information of each part of the message.

```
From: Ram Vyasana
To: Shyam Dhur
Subject: Hello Text Message in formatted form
MIME-Version: 1.0
Content-Type: multipart/alternative;
  boundary=boundary25
```

```
boundary25
Content-Type: text/plain; charset=ASCII
```

```
... plain text version...
```

```
boundary25
Content-Type: text/richtext

... richtext version ...
-boundary25
Content-Type: text/x

... formatted version...
-boundary25-
```

In the above example, users whose e-mail system support for text/x format would be able to read the formatted version. Other users would see only the richtext or plain text version.

6. Multipart/Parallel: Its syntax is similar to Multipart/Mixed format. Here, all the parts of a message are presented simultaneously for processing. It is basically used to send multimedia messages.

7. Multipart/Digest: It shows that each part is an RFC822 mail message.

8. Text: If the message is in text form, then text Content-type is used to send the message. The syntax for text Content-type used for Internet mail is

```
Content-type: text/plain; Charset=US-ASCII
```

9. Text/Plain: It shows that the message is in the plain text form.

10. Text/Richtext: It shows that the message is in the word processing format, as per MIME standard.

11. Video: It shows that the message contains image or picture.

12. Video/MPEG: It shows that the message is video encoded as per MPEG standard.

13. X-TypeName: This is any type name that begins with X-.

The mechanisms for defining new Content-Type subtypes are as follows:

(a) Private values (starting with X-) are defined between cooperating mail composing and reading programs. No outside registration is required. This is useful to the reader to correctly identify the type of the content.

(b) New standard values must be registered with IANA.

Syntax for user agents is

```
Content-Type: Text/plain; Charset = US-ASCII
```

14. Application: It shows data which is not represented in any category.

15. Application/Octet-Stream: This shows uninterpreted binary data. The parameters used are

(a) *Name:* This shows name of binary data

(b) *Type:* This shows type of binary data.

(c) *Padding:* It shows the number of bits used for padding.

16. Application/PostScript: This shows a message having a postscript document.

11.3.2 MIME Transfer-Encoding Header Field

Many Content-Types are usefully represented as 8-bit character or binary data. Some transport protocols such as SMTP are not supported for such data transmission. SMTP supports 7-bit ASCII data and message up to 1000 characters per line. MIME converts the data into 7-bit format and the mechanism used for conversion is shown in Content-Transfer-Encoding header field.

The Content-Transfer-Encoding field are as follows:

1. BASE64
2. QUOTED-PRINTABLE
3. 8BIT
4. 7BIT
5. BINARY
6. x-EncodingName

Message should be in a 7-bit mail-ready format. The syntax is

```
Content-Transfer-Encoding: 7bit
```

There are maximum 76 ASCII characters per line for BASE64 and QUOTED-PRINTABLE fields. For the data that consists of printable ASCII characters, QUOTED-PRINTABLE is the most appropriate encoding method. The equal's sign (=) is used for any character, which is not a printable ASCII character. To show extended line greater than 76 characters, equal sign is used.

The advantages of the QUOTED-PRINTABLE format are mentioned below:

1. Additional characters required are less.
2. The message is in readable form.

Non-standard values are represented by starting with x. Its syntax is

```
Content-Transfer-Encoding: x-new message.
```

If the header of a message contains a Content-Transfer-Encoding field, then it applies to the complete message. If it is included in the header of a part, then it applies to that part of a message. If a message Multipart or Message type, then the Content-Transfer-Encoding must be 7-bit, 8-bit or binary.

The syntax for header of a message or part of a message is

```
Content-Type: text/plain, charset=ISO-8859-1
Content-transfer-encoding: base64
```

This means that the part of a data, which is a Base64 ASCII, was originally in ISO-8859-1.

Optional Content-ID Header Field

It allows one part to reference another part of a message.

Optional Content-Description Header Field

The descriptive information is indicated using this header.

11.4 S/MIME

S/MIME stands for Secure/Multipurpose Internet Mail Extensions. It is a standard for public key encryption and digital signing of e-mail encapsulated in MIME. Apart from text data, there are some other data, which are also important. MIME specifies to encrypt those data which look like a text data to the third party in communication.

The standards for Internet e-mail were established in 1982. These standards put some restrictions on the e-mail messages. Some of the restrictions are as follows:

1. The message contains only ASCII characters.
2. The message contains maximum 1000 characters in one line.
3. The message does not exceed a certain length.

Since EDI messages can violate all of these restrictions, the 1982 standards, do not allow EDI to be reliably transmitted through Internet e-mail. There are some new standards which support to send different types of messages and services. A new Internet mail standard was approved in June, 1992. The new standard is called *MIME*. Before S/MIME, e-mail administrators used a widely accepted e-mail protocol, called *Simple Mail Transfer Protocol (SMTP)*. It was inherently not secure. S/MIME provides a solution for e-mail administrator that is more secure and also widely accepted. S/MIME is as important as SMTP because it brings SMTP to the next level. It allows widespread e-mail connectivity, with strong security.

11.4.1 History of S/MIME

The first version of S/MIME was proposed by RSA data security, Inc. in 1995. At that time, for security of message, no recognised standard existed, but many contending standards existed.

In 1998, the second version of S/MIME was submitted to IETF to be considered as Internet standard. In 1999, to improve the performance of S/MIME, IETF proposed the third version of S/MIME. RFC 2632 built on the work of RFC 2311 in specifying the standards for S/MIME messages, and RFC 2633 enhanced RFC 2312 specification of certificate handling. RFC 2634 added some additional services to S/MIME, which include labels, receipts and triple-wrapping.

Following two security services are provided by S/MIME:

1. Digital signatures
2. Encryption

S/MIME improves the security of e-mail by providing digital analogs. It also provides the integrity of a message. We have already discussed more about integrity in message digest and hashing algorithms. Privacy is another factor in e-mail communication. S/MIME makes messaging cheap, the software is user-friendly and reduces the cost. Due to all these factors, the commercial community attracts towards S/MIME. The first example of this community is the Electronic Data Interchange (EDI). It has been made over value-added networks (VANs). VANs provide trustworthy and secure service, but these are costly. S/MIME helps in reducing the cost of this service, and at the same time, improves the security, makes user-friendly connectivity and reduces the response time.

11.4.2 Working of S/MIME

S/MIME uses symmetric encryption algorithms, public key cryptography, message digest algorithms, and certificates for authentication and message integrity. This improves the efficiency of S/MIME. S/MIME uses RC2 and Triple-DES as symmetric encryption algorithms, RSA as key generation algorithm and SHA-1 or MD5 as message digest algorithm. We will see how different services such as secrecy and authentication are provided in S/MIME.

Secrecy

Suppose Ram wants to send a secret message to Shyam so that only Shyam can read it. Ram and Shyam has to take some steps to provide security to the message as below:

- (a) A random key, called *session*, is generated. It is used to encrypt for only one session of the e-mail. Then, the e-mail program uses symmetric encryption algorithm and the session key to encrypt the message.
- (b) The session key is also encrypted using Shyam's public key.
- (c) The e-mail program creates folder, which contains encrypted message, encrypted session key, Ram's x.509 certificate, and information of the encryption algorithms used.
- (d) The folder is transmitted to Shyam. This is an S/MIME e-mail message. This encrypted message is called *digital envelope*.
- (e) When Shyam receives the message, his private key is used to decrypt the session key. He also gets the information about the encryption algorithm used for the encryption of message.
- (f) Using this session key, Shyam's e-mail program decrypts the message.

Authentication

Suppose Ram wants to send a message to Shyam. He wants to prove (i.e., authenticate) that the mail is really sent by him and not by someone else. Ram and Shyam have to take some steps to provide authentication of the sender as below:

- (a) A message digest algorithm is used to create a message digest of the message.
- (b) The e-mail program encrypts the message digest using Ram's private key.
- (c) The e-mail program creates folder that contains the original message, encrypted message digest, Ram's x.509 certificate, and information of the encryption algorithms used.
- (d) The folder is transmitted to Shyam. This is an S/MIME e-mail message. This encrypted message is called *digital envelope*.
- (e) When Shyam receives the message, first, his e-mail program verifies whether x.509 certificate is valid or not. If it is valid, then he retrieves Ram's public key from the certificate.
- (f) Ram's public key is used to decrypt the message digest. He also gets the information about the message digest algorithm used by Ram.

- (g) Shyam's e-mail program computes the message digest of the message received.
- (h) Two message digest values are compared. If the values match, then Shyam authenticates Ram as the originator of the e-mail received.

Secrecy and Authentication

To provide secrecy and authentication to a message, we should use the combination of the above two approaches. For this, one has to follow the following steps:

Use authentication technique to the message such as calculate message digest.

The authenticated folder is encrypted using receiver's public key.

Transmit the folder to the recipient.

The recipient decrypts the folder by using his/her private key. He gets the session key.

Using session key, he decrypts message. The result is a signed S/MIME message.

11.4.3 Applications of S/MIME

S/MIME is useful for transmitting the data securely through e-mail. So, many companies and organisations use S/MIME to securely exchange their data. In software companies, part of code can be securely transmitted using S/MIME. Government organisations and stock market data can be securely transmitted using S/MIME. Many hospitals use to send patient's information confidentially to the authorised persons only.

11.5 COMPARISON OF PGP AND S/MIME

A comparison of PGP and S/MIME is given in Table 11.2.

Table 11.2 Comparison of PGP and S/MIME

Class	PGP	S/MIME
Authentication	Distributed authentication	Hierarchical authentication
Key storage	Key ring	Key certificate
Standard	–	IETF
Commercialisation	No compatibility test, small products	Compatibility test, many commercial products
Applications	Personal	Company, enterprise
Summary		

SUMMARY

PGP was developed in late 1980s and early 1990s by Phil Zimmermann. PGP is used for encrypting and digitally signing the data. It is used for authentication of the sender and also for encryption applications such as e-mail and attachments, digital signatures, laptop hard disk encryption, file and folder security, protection. The security of PGP depends on the assumption that the PGP algorithms used are difficult for cryptanalysis using current hardware and

CHAPTER 14

Intrusion

14.1 INTRODUCTION

Network security is becoming increasingly important in the modern systems, where Internet is an essential part of human life. The evolution of the Internet along with its use in day-to-day life has increased the need for network security systems. With the development of networking and interoperation on public networks, the number and the severity of security threats have increased significantly. Internet has changed the life of human being completely. Applications of computer using Internet are unlimited. Unfortunately, due to large scale use of Internet, the risks and chances of attacks are also increased. So, it is essential to protect our system from different attacks. The system which is used to protect our system from different attacks is called *intrusion detection systems (IDS)*. The process of identifying the attacks in a system or network is called *intrusion detection*. An *intrusion* is a deliberate or unauthorised activity or action that attempts to access or manipulate the information or compromise the security of the systems to make them unreliable or unusable. An *intruder* is a person who is responsible for intrusions. He may be a person from inside the network, i.e., legitimate user of the network or from outside the network.

It is a well-known thought that prevention is better than cure. Same is applicable to computer systems and networks. Generally, firewall is used to prevent attacks on the network. Firewall is having a set of rules and it protects those attacks, which are defined in advance as a rule. So, firewall cannot protect the new attack, as the rule is not defined. In this case, IDS is useful to detect new attacks. But it is unrealistic to prevent all the attacks. An IDS collects the information from inside as well as from outside the network and analyses this information to identify whether there is intrusion or not.

Intrusion detection is different from intrusion prevention. *Intrusion detection* means the process of observing the incoming and outgoing traffic and it collects the data. Then, it analyses the data for possible attacks. *Intrusion prevention* is the process

of identifying the attacks and it endeavours to block the detected possible incidents. *Intruders* are the person, who have unauthorised access the network.

There are different types of intruders.

1. Masquerader: This refers to the unauthorised user of the computer system, who penetrates the security of computer system by using legitimate user account. Generally, masquerader is an outsider.

2. Misfeasor: It refers to a legitimate user who accesses the resources that he is not authorized to access, or the user who is an authorised user of a system, but misuses his privileges. Generally, misfeasor is an insider.

3. Clandestine user: It refers to a user who gains supervisory control over the system. Clandestine may be either an insider or outsider.

Anderson, first time in 1980, introduced the concept of intrusion detection. He defined an *intrusion* is an attempt or a threat to be the potential possibility of a deliberate unauthorised attempt to

1. Access information
2. Manipulate information, or
3. Render a system unreliable or unusable

After 1980, many techniques for intrusion detections have been studied.

14.2 INTRUSION DETECTION

Intrusion detection means to detect the vulnerabilities exploited against the computer system or against any application. Intrusion detection system helps in providing the information about such vulnerabilities to the network administrator and helps him in preparing some system to protect such attacks or deal with such vulnerabilities. It includes collection of information by monitoring the network traffic and the suspicious activities in the network. It also collects the information about these vulnerabilities from different sources and analyses the same. Many people think that firewall is sufficient to protect their network and can recognise the attacks on the network and block the intrusions. But the fact is that the firewall works just like a fence to our home. It restricts the access only to the designated points on the network, but the whole network cannot be secured using firewall. Firewall cannot detect the new attacks on the network. This detection of new attacks is done by IDS.

Intrusion detection provides the following functions:

1. Monitoring and analysing both the user and the system activities
2. Analysing system configurations and vulnerabilities
3. Assessing the integrity of system and files
4. Analysing the traffic pattern based on knowing attack patterns
5. Analysing abnormal activity patterns
6. Tracking the policy violations by the user
7. Doing audit of the operating system

Today, it is more difficult to provide 100% security to any network. This is because the technologies for attacks are very user-friendly and many free tools are

easily available to perform such attack. No prior technical knowledge is required to attack any system. This helps a novice attacker in making the attack with more ease.

The intrusion detection system can be divided into following two types depending on the architecture:

1. Network intrusion detection system (NIDS): It works on the network and performs an analysis of all the traffic passing on the entire subnet. Every packet is monitored and if the attack is identified or some abnormal behaviour is observed, then the alert can be sent to the administrator.

2. Host intrusion detection system (HIDS): It works off the host, monitors the system events and audits the event logs. It then takes a snap shot of the existing system files and compares it with the previous snap shot available. If any of these files are found modified or deleted, then the alert is sent to the administrator.

14.3 INTRUSION DETECTION SYSTEM

An *intrusion detection system (IDS)* is a security system that monitors the network traffic and analyses the data for possible attacks from outside the network or from inside the network.

IDS can be categorised depending on the method of detection attacks. Following are the categories of IDS:

1. Misuse-based detection versus anomaly-based detection: The misuse-based intrusion detection systems (IDS) uses a database of previous attack patterns and known vulnerabilities as a reference. Each intrusion have some specific pattern. This pattern is called *signature*. This pattern or signature is used to identify the attacks on the computer system or on the network. So, this system is also called *signature-based IDS*. The drawback of misuse-based IDS is that there is a need of frequently updation of the database. If there are some unique attacks, this IDS may fail to identify such attacks.

In anomaly-based intrusion detection systems (IDS), a baseline or learned pattern of normal system activity is used as references to identify intrusion. Using this information, an alarm is to be triggered. The drawback of this method is that it has higher false alarm rate.

2. Network-based system versus host-based systems: Network-based intrusion detection system monitor the packets that flow over the network. These packets are compared with the reference data present and then analysed. Then, it is verified whether the said packet is malicious or benign. It is responsible to control the vulnerabilities in the networks, so, it is distributed IDS. Network-based IDS uses packet-sniffing technique to collect the packets along the network. The architecture for Network-based IDS is shown in Figure 14.1.

In a host-based system, the activity on each individual computer or host is examined by the IDS. It is installed on an individual computer to detect the attack on that computer. There are two drawbacks of this method. The first drawback is that the system compromises with security; therefore, the log files of the system are corrupt or inaccurate. The same corrupted or inaccurate files are reported by the IDS, which makes it unreliable. The second drawback is that in host-based IDS, the IDS has to be deployed on each computer, which increases the administrative overload.

3. Passive system versus active system: A passive IDS is configured to only monitor and analyse the network traffic and if vulnerability or attack is found, it sends an alert to the network administrator. It is not able to protect or correct any actions by itself. An active IDS is used to block the suspected attacks automatically. There is no intervention required from the network administrator. It is also known as *intrusion detection and prevention system*. The advantage of this method is that it takes real time corrective action.

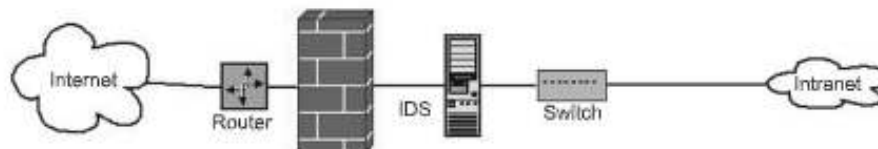


Figure 14.1 Network-based IDS.

14.3.1 Need for Intrusion Detection Systems

Out of the total security attacks that occur on a network, up to 85% attacks come from the users inside the network. These users may be authorised users of the system. The remaining attacks come from outside the network. It consists of mainly denial of service attacks or attacks to penetrate the infrastructure of the network. To protect the network from all these types of attacks, IDS is an integral part of the network or information security. It is helpful for complete supervision of the network. IDS is used to

1. Prevent problem like behaviours of the system
2. Detect various attacks and vulnerabilities in the network
3. Detect new attacks and identify its signature
4. Protect the network from internal as well as external users

Nowadays, due to the availability of tools for making attacks, it is very easy to make attack on any computer system or network. There are different methods to protect the system or network from these attacks. Firstly, develop a fully secure computer system or network. For this, the system is accessible only to the authenticated and authorised users. Secondly, use of cryptographic methods to protect the data applies tight access control. But in real life, all these solutions are feasible due to the following reasons:

1. In actual practice, to develop a completely secure system is not possible. Designing and implementing a totally secure system is an extremely difficult task.
2. Use of cryptographic methods to protect the information has its own limitations. The security of these methods depends on the secret key. If the attacker is able to capture this secret key, then he can read, change or modify the data and the entire system can be broken.
3. Many times, the protective measures are applied to prevent the external attacks. But as discussed above, approximately 85% of the total attacks are

from internal users. This happens because the internal legitimate users misuse their privileges and create the attacks internally.

4. If we tight the access control, the efficiency of the system reduces.

When an attack is detected, the IDS first alarms the network administrator. IDS works as a reactive system, instead of preventative one. It works as an informative system.

14.3.2 Intrusion Detection Method

Intrusion detection can be done using the following strategies:

1. Define the rules for the normal behaviour of the computer system or network and then search for the traffic, which is responsible for the change in behaviour of the system or computer.
2. Define the patterns of the attack and then search for the occurrence of an attack.

The first strategy is called *anomaly based IDS* and the second strategy is called *misuse-based IDS*. We will now discuss about these methods in the subsequent sections.

Anomaly-based Detection

Anomaly-based detection techniques are based on the assumption that all intrusive activities are malicious. Therefore, we have to build a system, with a normal activity profile of the computer system and then wait for the anomalous activities to happen. That is, we identify the system states which have different behaviour from the normal established profile. Such activities are identified as intrusive activities and flagged as intrusion. However, if we assume that the rules for intrusive activities and the rules of anomalous activities are not exactly the same, but there are some matches among them, then there are chances like

1. Some activities, which are anomalous, but not intrusive activities, are also flagged as intrusion. This results in false positives.
2. Some activities are intrusive activities, but not anomalous. Such activities are not flagged and treated as normal activities. This results in false negatives.

False negative is a serious problem, as malicious packets are allowed to enter in the network or system as a normal packet. It may harm the system. This reduces the accuracy of the IDS and deteriorates the performance of the IDS. To reduce the false negatives in the system, generally the threshold is used. In anomaly-based IDS, there is a need to keep track of system profile and also updating of system profile is required. Therefore, this method is computationally expensive. The advantage of this method is that it is able to detect the new or unknown attacks.

A block diagram of anomaly-based detection system is shown in Figure 14.2.



Figure 14.2 Anomaly-based detection system

The main advantages of anomaly-based detection system are as follows:

1. It is possible to detect the new or unknown attacks.
2. Accuracy is more.
3. Internal attacks can be detected easily.

Disadvantages of anomaly-based detection system are given below:

1. False negatives are more.
2. It is expensive.
3. Accuracy is less.

Misuse-based Detection

In misuse-based detection method, the patterns or signatures are used to detect the attacks. If there are variations in the same attack, then it is possible to detect those attacks by using pattern or signature. Misuse-based detection systems use a database of information that has a number of patterns. The system collects audit data and compares this data with the stored patterns in its database. If any match is found, then the alarm is generated. If the match is not found, then it is considered as legitimate activity. The success of misuse-based detection method depends on the database of signatures or patterns. The database should include all possible patterns with the variations for different attacks and also for normal activities. How to generate these patterns or signatures is the main issue of this approach.

A block diagram of a misuse-based detection system is shown in Figure 14.3.

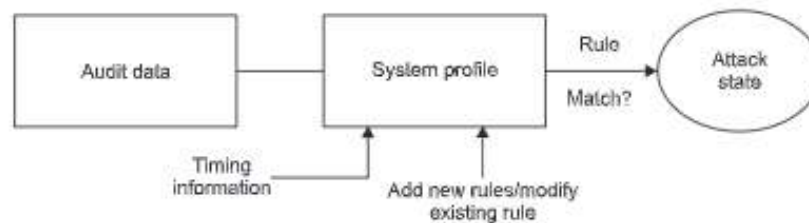


Figure 14.3 Misuse-based detection system.

The only advantages of misuse-based detection system is that it generally produces very few false positives.

Disadvantages of misuse-based detection system are as under:

1. A lot of effort is required for the generation of pattern or signature database.
2. It cannot detect new attacks.

14.4 ANOMALY-BASED INTRUSION DETECTION SYSTEMS

Anomaly-based intrusion detection systems have few major approaches to detect the intrusions. Some of these approaches are described here.

14.4.1 Statistical Approach

Statistical approach is the earliest method used for intrusion detection. It is assumed that the normal behaviour and malicious behaviour are different. So, statistical approach can be used to differentiate the normal user from the intruder. In this approach, the behaviour profiles are generated. Its variances are generated from the present profile. This approach is adaptive; so, the performance of this approach is good. The drawback of this method is that the intruders are trained gradually so the intrusions are treated as normal. The performance of this approach depend on the threshold. If the threshold is set too low or too high, then it affects false negatives and false positives.

Denning suggested a number of statistical parameters for IDS. Some of them are as follows:

- 1. Threshold:** It is the heuristic limit to occur an event or count of events occurs within a specific interval. For example, logging into user account is not allowed after a specific number attempts of failed log in.
- 2. Mean and standard deviation:** The confidence interval for the abnormality is computed using the comparison of event measures and the mean and standard of a profile deviation.
- 3. Multivariate model:** This model considers computing the correlation between different event measures with respect to the profile expectations.
- 4. Markov process model:** This model considers the types of events with respect to the state variables in a state transition matrix.

Limitations

1. The performance of statistics-based IDS depends on the data representation. If it contains some irrelevant data, then IDS may fail to identify an unknown attack.
2. The performance of this approach depends on the threshold. If the threshold is set too low or too high, then it affects false negatives and false positives.

14.4.2 Immune System Approach

This approach provides a model of normal behaviour in the form of application code paths. For a variety of different conditions, the applications are modelled in terms of sequences of system calls. These conditions include normal behaviour, error conditions and attempted exploits. These models are compared with the event observed and then classify them as normal or attack. This approach detects a number of typical attacks, but cannot detect the attacks which are based on race conditions, policy violations or masquerading.

14.5 MISUSE-BASED INTRUSION DETECTION SYSTEMS

Misuse-based intrusion detection systems have few major approaches to detect the

intrusions. Some of these approaches are described below. In misuse-based detection, it is checked whether a pattern being executed violates the security policy of the system. If this happens, then set the alarm for an intrusion.

14.5.1 Expression Matching

It is the simplest form of misuse-based IDS. It uses techniques like matching of expression, searches for various event patterns. For this model, we can define the signatures/patterns easily.

14.5.2 State Transition Analysis

This model uses matching events to find out the attacks. Every observed event is compared with the finite state machine patterns and causes the transitions. If the machine reaches the final state, that means it is an attack. Complex intrusions can be detected using this model. It is used to detect distributed attacks.

14.5.3 Genetic Algorithm

Genetic algorithm can be used to identify the known attacks. In this, the patterns of the observed event are compared with the available patterns and the best match is found out. Then, a hypothesis vector is evaluated depending on the risk associated with the attacks involved. If a mismatch occurs, a quadratic penalty function is used to give the details. In each turn, the best hypotheses of the current set are mutated and retested again. This reduces the false positives and false negatives to zero. The performance of this approach is good.

14.6 DISTRIBUTED INTRUSION DETECTION SYSTEM

With the widespread growth of internet, there is an increase in malicious activity against the network. Current IDS technology is sufficient to protect the global network infrastructure from attacks. So, there is a need of an IDS, which can support global network. Distributed IDS (dIDS) increases the identifying power and scope of single IDS by using an attack correlation with database obtained from geographically different networks. A distributed IDS consists of multiple intrusion detection systems over a large network. All these IDSs are communicated with each other, or with a central server. This allows them to monitor different advanced networks, help for analysis and find out the attack data. There are different agents which coordinate with each other and are distributed across a network. This gives the detailed pictures of different events that take place in the network to the network administrators. It allows maintaining the records related to attacks at the central place so that it is easily available to the analyst easily. There is a centralised analysis engine and on each system, there is an agent which monitors the network traffic. Current dIDS has the following limitations:

1. Observing a single site is not sufficient to detect the existence of attacks by the single attacker.
-

2. When a single attack is generated by a group of attackers, then to protect such attack, the global scope for assessment is required.
3. If there is a change in the system and attack behaviour, then false negatives are generated.
4. For prevention of attacks, the intention behind these attacks is required.
5. Due to advances in technology the attacks are automated. Also, these attacks are autonomous. To prevent these attacks, quick analysis and mitigation of these attacks are required.
6. The total volume of notifications about the attacks received by internet service providers (ISPs) and host systems is becoming overwhelming. If collective details about the attacks are provided to the responsible party, then the possibility of a positive response increases.

Thus, in nutshell, distributed intrusion detection system has a number of IDSs, which are spread over a global network (Internet). All these IDSs are interconnected and communicate with each other or with a central server. This allows it to monitor the network, help for analysis of the incident and find out the data related to attacks.

14.6.1 Overview

There are various components of dIDS. These components are discussed below:

Central Analysis Server

The core part of dIDS is central analysis server. It consists of a database, reporting engine and the web server. It collects the responses from the agents to permit attack correlation. It allows analysis to perform aggregation of attack, collect statistical information, recognise the pattern of attack and report the incident details.

Cooperative Agent Network

The most important components of the dIDS is the cooperative agent network. An *agent* is a software or a device which is responsible for collecting and inspecting the data. Agent might be a firewall, or host-based IDS or network-based IDS. Agent reports the information related to the attack on the central analysis server. There are multiple agents across a single network. It gives a broad view of the network than a single IDS.

Generally, these agents are located on separate network segments. They are also located on different geographical locations, as shown in Figure 14.4. The agents can also be distributed across multiple physical locations. Due to this, a single incident analysis team can view attack data across multiple corporate locations.

Attack Aggregation

Another important part of dIDS is attack aggregation. It is located in the central server. It is the method in which the information is gathered from the agent network. An example is collecting the information according to IP address of the attacker, putting together all attacks from an IP address, together with the other attacks from the same IP address. Another example is to collect the data of attacks according to destination port or put them by date and time.

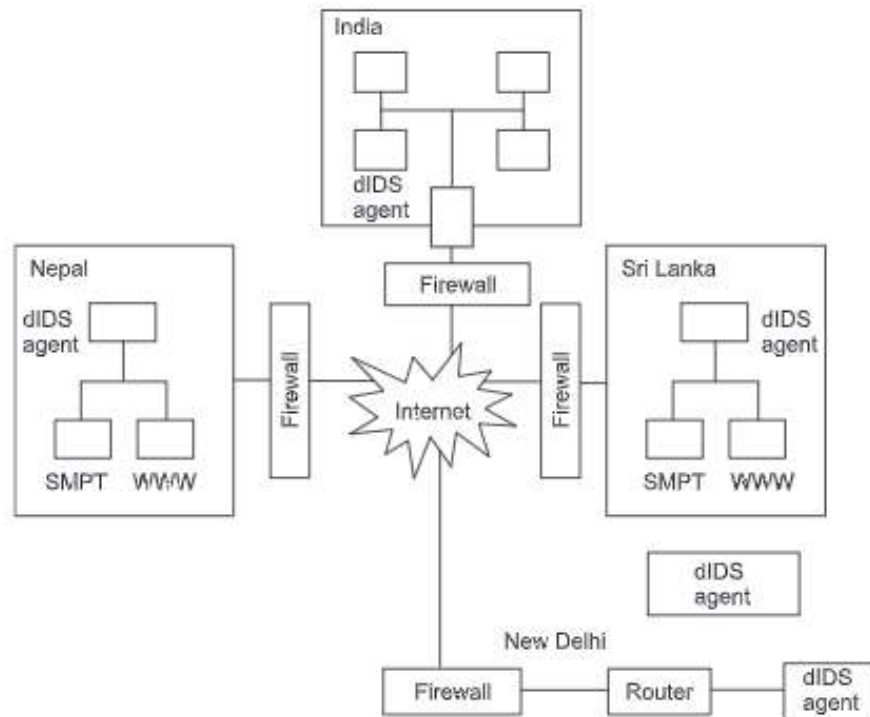


Figure 14.4 Agent network.

Data Sources

The agent reports about the internal network traffic to the perimeter or an organisation or both. This depends on where the agent resides. For dIDS, both the locations are useful because some malicious software behaves differently for inside the network than for outside the network. To get useful information about the internal attacks, the agent located inside the organisation is useful.

14.6.2 Advantages of a dIDS

The dIDS has many advantages over a single IDS. It can detect attack patterns across the entire network of the organisation, with physical locations having different time zones or even in different continents. Therefore, the attacks can be detected early and this protects the system from damage. The offending IPs are prohibited to access. Another advantage of dIDS is to allow early detection of Internet worm through the global network. The major advantage of dIDS is that a single team can analyse the attack information of places located at different places, whereas a single IDS system requires several analysis teams for the same. So, it reduces the cost of analysis. It also reduces the time required to collect the information from different places. Many times, the attacks are generated by people within the network. With the help of analysis server, the incident analyst is able to keep track of such attacks within the

organisation and provide evidence against such internal people. This helps in protecting the system or network internally.

14.6.3 Incident Analysis with dIDS

The important part of dIDS is incident analysis. The main command, strength and flexibility of the dIDS lie with incident analysis. The analysis of incidence of attack helps in protecting the system. So, while designing dIDS, the designer mainly focusses on this part.

14.6.4 Analysis Using Aggregation

Aggregation is helpful for the analysis of attack incidence across the multiple segments of a network. By aggregating the similar or related data from multiple physical locations using multiple agents, the analyst, using dIDS, is able to examine how an attack has progressed across multiple network segments through different stages. Incident analyst also gets the information about time frame in which the attacker is working. He also correlates other attack attempts against the networks and decides about the attacker's working, i.e., whether he is working in multiple cooperative groups. The most common methods of aggregation are source IP, destination port, agent ID, date, time, protocol, or attack type. These are discussed below:

1. Aggregating by source IP: It allows an incident analyst to follow an attacker's attempt from start to finish across the multiple segments of a network.

2. Aggregating by destination port: It permits an incident analyst to look new trends in attack types and methods. Using this method, dIDS is able to identify several new attacks.

3. Aggregating by agent ID: It allows an analyst to see the variety of attacks and different attempts made by the attackers on a specific segment network, where the agent is located. The analyst can find out whether the attackers are working in group in cooperation with each other or they target some specific network segments for attacks. This analysis is helpful to the security team to concentrate on providing the security.

4. Aggregating by date and time: It allows the analyst to look the pattern of new attack. So, new worms or viruses can be identified easily. These worms or viruses are only triggered at certain times.

5. Aggregating by protocol: It works in a statistical manner. The analyst can identify new attacks in particular protocols or identify protocols on a network segment using aggregating by protocol.

6. Aggregating by attack type: It is useful for matching the attack pattern and also correlates the coordinated attacks against multiple network segments.

Using the above aggregation methods, an analyst can get a variety of data to correlate against other attacks. Therefore, a variety of data is available to the analyst. He can identify the attacks by correlating this data against other attacks. It is possible to detect coordinated distributed attacks, attacks within the network and new exploits and vulnerabilities. Use of dIDS gives a guarantee of minimum false positives and false negatives.

14.7 BASE RATE FALLACY

The main aim of analyst is to make the detection rate of intrusion detection system high, with minimum false negatives. This can be achieved using the base rate fallacy phenomenon. For assessment, two types of information are available with the analyst. One is the specific information about the individual case at hand, and second is in terms of numerical data that summarises information about many similar cases. This type of numerical information is called *base rate or prior probability*. For IDS, we can apply base rate fallacy by first finding the different probabilities or if such probabilities cannot be found, then necessary assumptions are made regarding these probabilities.

14.7.1 Basic Frequency Assumptions

We have to install the network consists of few workstations and some servers. This installation produces 10,00,000 audit records per day used to test data for intrusion detection. Suppose there are very few actual intrusion attempts per day. But it is difficult to know the exact number of real attack attempts per day. If we assume that only one person can analyse the data that react to a relatively low number of alarms, especially when the false alarm rate is high, then in this case, there is a possibility that an intrusion could affect only one audit record; it is likely, on an average, that it will affect a few more than that.

14.7.2 Honeypots

A *honeypot* is an information system on the Internet. It is a virtual machine used mainly to attract and trap people, who make an attempt to penetrate other users of the computer systems. A honeypot makes fool of attackers by enabling them to believe that it is a legitimate system. The attacker attempts the attack on the honeypot assuming that it is a system. These attempts are observed, but the attacker does not know about it. At the same time, the information related to attack is collected by the system, which includes the IP address of the attacker. This information is useful to trace back the attacker. When many honeypots are connected to a system on a subnet, it is called a *honeynet*. Honeypots can be used for multiple applications such as for detection, prevention or information gathering.

The objectives of honeypot designs are given below:

1. It diverts the attacker from accessing some specific systems.
2. It is useful to collect the information regarding the attacks.
3. It motivates the attacker to use the system for more time so that the administrator gets more time to collect the information about the intruder.

Types of Honeypots

Honeypots are classified into two types.

1. Production honeypots: These honeypots are easy to use. It captures only limited information and uses it for risk mitigation. Most of these honeypots are emulations of specific operating systems. The attacks generated by automated tools are random and can be identified by these honeypots. Some of these honeypots can shut down

attacks altogether. They send an acknowledgement of zero size to them. The attack waits to increase the size of window so that it could send the data when the window size increases.

2. Research honeypots: These honeypots are real operating systems and services. The attackers interact with these operating systems (honeypots); therefore, there is high risk with these honeypots. These honeypots are used to collect detailed information regarding new methods and techniques used for attacks. All these information provide an accurate picture of the types of attacks being penetrated. They also provide more information from the log files. The information provided by honeypots is used for intrusion detection and prevention.

Advantages

1. Honeypots attract and trap attackers and can be diverted them to target that they cannot damage.
2. Administrators have sufficient time to respond to an attacker.
3. The activities of the attackers' can be monitored through it. It captures valuable information about the attacks and uses this information for analysing attacking techniques and improves the threat models.
4. It can capture internal attacks efficiently and provides information about internal attackers.
5. It does not add extra burden on the existing bandwidth of the network.

Disadvantages

1. Honeypots cannot detect attacks against the other systems in the network because they only track and capture activity that directly interacts with them
2. If honeypots are deployed without planning and thought, then they may introduce risks to the existing network.

14.8 PASSWORD MANAGEMENT PRACTICES

One of the important security services is authentication. Password is the most common mechanism for authentication. So, there should be a password management policy. The policy contains the procedure for selecting the passwords, validity of the password, distribution rules. There are different types of password management practices. Some of them are discussed here.

User Authentication and Passwords

It describes the objectives of authentication of the user and alternative mechanism for authentication. It also describes the importance of the passwords for the authentication of the user. In many applications, identification of user depends on authentication. For authentication, different methods are used. These methods are classified below:

1. Secrets (e.g. a password or PIN)
2. Tokens (e.g. a token card or smart card)
3. Biometrics (e.g. fingerprint, iris, voice or face)

The cost of secrets using password is less as compared to the other methods. So, password is the most popular method for authentication.

Security Threats

Using passwords or PIN for authentication is the most knowledge-based authentication method. Today, the use of computer for day-to-day life is increasing. This results in increasing the number of user IDs and passwords. For any individual, there are many user IDs and passwords. To remember all these user IDs and passwords is difficult for any user without keeping any record of it. If any individual user uses the same user ID and password for all his accounts and if the password is compromised, then huge damage may happen to the user. Because an intruder may gain access to all the accounts of that individual user using a single password. If the user selects different passwords and user IDs for different systems, then there is a tendency to select the passwords which he can easily remember. Secondly, there is a chance that the user may forget the passwords. This increases overload on the system to provide new passwords.

The most convenient and easy method of authentication is password method. It can be easily implemented. At the same time, there are a number of risk factors associated with this method. Some common risk factors are given below:

1. As an individual can have a number of passwords, there is a tendency to write down or share the password. Therefore, the password does not remain secret.
2. When the individual uses his password, the evesdropper may be able to know the password. This can be done by observing the typing of the keys or from the camera.
3. Attackers may guess the password using brute force method. If the length of the password is small, it is very easy for the attacker to apply brute force method and find out the password.
4. Password can be captured using sniffing attack. When the passwords are transmitted over a network either in plaintext or in poor encrypted form, then sniffing tool is used to get the password.
5. Attacker creates a fake log in screen looking similar to the original screen. When the user logs in to the fake screen, immediately his password is captured and informed to the attacker.

Using one of the above techniques, an attacker is able to get the password of the users. There is no method which ensures whether the holder of the password is a valid user or somebody else.

Human Factors

In a large organisation, each user may have different passwords for accessing different computer systems. To provide security, some rules are framed for selecting the password. But in general, it is difficult for any individual to remember all these passwords and a particular password for a particular computer system. For the users, it is difficult to remember

1. Complicated passwords
 2. Number of passwords
 3. Frequently changing passwords
 4. Rarely used passwords
-

For any one of the above case, it is troublesome for the user to remember the passwords. So, they select one of the following options.

1. Users may note down their passwords on a paper. This reduces the security of the password and also the security of the computer system.
2. Many people forget their passwords and request the network administrator for a new password. This increases the overhead on the network administrator.
3. Many users use a single password for all the computer systems. If the password is compromised, then the attacker may access all the computer systems.
4. Many times, users select very simple and easy-to-remember passwords. But these passwords can easily be compromised.
5. Sometimes, users select old password and reuse it again and again.

There should be some strong password management system which helps in avoiding the above practices of users.

Composition Rules

The main weakness of the password authentication method is that the password can be easily guessed. Using software like Crack and L0phtCrack attacker can guess millions of passwords combinations easily. So, it is necessary to prevent password guessing attack by selecting a strong password. Password strength is decided by the length and complexity of a password. Complexity depends on the unpredictability of the characters used in the password. For strong password, select a long password. The strong password can be generated using the following guidelines:

1. Password should be a passphrase having multiple words or acronyms.
2. It should be a combination of both upper and lowercase characters (e.g., a-z, A-Z).
3. It should have some digits and punctuation characters as well as letters (e.g., 0-9, !@#\$\$%^&*()_+|-=\`{}[]:;'\<>?.,.).
4. The length of the password should be at least 6 alphanumeric characters long.
5. The password should not be a word in any language, slang, dialect, jargon, etc.
6. It should not be based on some personal information such as names or family or pets name, etc.

The strength of password for some likely combinations using the above rules are shown in Table 14.1.

Table 14.1 Password strength

Legal characters	5	6	7	8	9	10
0 to 9	$1.00e^{04}$	$1.00e^{05}$	$1.00e^{07}$	$1.00e^{08}$	$1.00e^{09}$	$1.00e^{10}$
a to z	$1.00e^{06}$	$3.09e^{08}$	$8.03e^{08}$	$2.09e^{11}$	$5.43e^{12}$	$1.41e^{14}$
a to z and 0 to 9	$6.05e^{07}$	$2.18e^{09}$	$7.84e^{10}$	$2.82e^{12}$	$1.02e^{14}$	$3.66e^{15}$
a to z, 0 to 9, 3 punct	$9.02e^{07}$	$3.52e^{09}$	$1.37e^{11}$	$5.35e^{12}$	$2.09e^{14}$	$8.14e^{15}$
a to z, A to Z	$3.80e^{06}$	$1.98e^{10}$	$1.03e^{13}$	$5.35e^{13}$	$2.78e^{15}$	$1.45e^{17}$
a to z, A to Z, 0 to 9	$9.16e^{08}$	$5.68e^{10}$	$3.52e^{13}$	$2.18e^{14}$	$1.35e^{16}$	$8.39e^{17}$
a to z, A to Z, 0 to 9, 32 punct	$7.34e^{09}$	$6.90e^{11}$	$6.48e^{19}$	$6.10e^{15}$	$5.73e^{17}$	$5.39e^{19}$

As per the system restrictions, user has to select his passwords. For example, some systems are not case-sensitive and treat uppercase and lowercase letters as same. So, there is a need to design the password policies which ensure that the search space for all possible combinations of password is reasonably large. Such passwords are difficult to guess.

Changing and Reusing Passwords

For better security, the passwords should be changed periodically. But many users do not follow this rule and use the same password for long time. Such passwords can be compromised. Some of the ways in which password may be compromised are given below:

1. Passwords shared with the other users
2. Passwords noted down on some paper
3. Password transmitted via insecure manner
4. Use of social engineering by the intruders to capture the password

To avoid the compromises of the password, the best solution is to change the password regularly. Many systems force the users to change their passwords after some fixed time period. Also, another precaution the user should take is that avoid the use of old password again. To avoid the reuse of passwords, many systems keep track of passwords previously used by the particular users. This may keep record of some limited attempt of changing passwords. So, some users who do not wish to change their passwords may take advantage of these criteria. Some users change their passwords to the required times in a day, and then, again use their current password. To prevent such users from doing this, all the previous passwords of the users should be stored so that in any case, user's old password is not permitted.

Secrecy

To identify a user uniquely, a password is used because it is secret to each user. Many times, users are not serious for keeping their passwords secret. As a result, other person may know the password. This is happened as the users:

1. Select passwords which are easy to guess
2. Share the passwords with the other persons
3. Note down passwords on paper or store in the computer file

Intruder Detection

If the user tries to log in using wrong password, in many systems, there is a provision to detect such attempts. If there are too many such attempts within a short period of time, then it is assumed that somebody tries to log in some other user's account by guessing the passwords. To protect the system from such an attempt, many systems allow the user some fixed attempt to feed the password and after that the user account is locked. For example, if the user tries to log in the SBI account, after attempting three times with wrong password, the system locks that account for a day and not allowed to use that account for that day for a valid password also. This method is

effective to protect the attack against a single user account. This method cannot be useful to prevent the attacker from guessing the passwords of multiple user accounts simultaneously. In order to guess the password, attackers makes some number of attempts to login through a particular account which will cause locking of the account for some pre-decided period. This may lead to denial of service attack for the genuine user of that account.

To avoid this attack, the best practices are as follows:

1. Apply the wrong log in attempt limit only to the users account. There should not be any limit for wrong log in attempt to the administrator log in.
2. Apply some high threshold value for intruder detection, e.g., 10 wrong log in attempts within 10 minutes.
3. Automatically allow the user to log in after some time, e.g., after 30 minutes.

Encryption

Generally, user passwords are stored on the workstations or the server and during his first log in, the password is transmitted from the workstation to the server in encrypted form. But the best method is to use the hashing algorithms like MD5 or SHA algorithms to check the integrity of the password with the encryption of the password. This provides more security to the password. The password transmission from workstation to server is based on the server's protocol used. Mainframe or Unix servers have no default encryption method used for the transmission of passwords. Some computer systems may cache password and when the user wants to log in, the passwords automatically transmitted to the server. This transmission of password also requires strong protection. So, if security is not guaranteed, then avoid such methods.

Synchronisation

Generally, users have different user accounts on different systems for which they have different passwords. So, there should be some mechanism to help the users in keeping the passwords same that they can use for their different accounts on different systems. This process is called *password synchronisation*, which helps the users to log in different user accounts with same passwords. It is questionable whether such technique is really secure or not. There is some debate as to whether password synchronisation makes systems more secure, or less. The questions may be as follows:

1. Compromise the security: A user has same password for a number of accounts on different systems in the same network. If any system compromises the security, then the attacker/intruder is able to log in all the accounts of the same user. To avoid such attack, the user should use different passwords for different accounts in the same network.

2. Synchronisation improves security: As discussed earlier, a single user may have different passwords for different systems. But there is a problem of remembering all these passwords. Many users write their passwords on paper or store on their notepads, laptops, etc. This reduces the security or there is almost no security. So, synchronised password avoids this need of remembering the password and improves the security.

Some guidelines for using synchronised passwords are mentioned below:

1. Only strong security systems should be involved in this synchronisation system.
2. Users should change their passwords periodically.
3. The passwords should be strong enough so that it cannot be compromised easily.

It is better to use a single, but strong password rather than using multiple passwords. The password should be changed periodically.

User Support

Nowadays, there are a number of user accounts for a single user on different systems. So, the user may forget his password for a particular system. In this situation, he tries to log in with a wrong password and this leads to lock his account. If this happens, the help desk provides help to solve the problem of log in. The process is given below:

1. There is a log in problem to the user.
2. The user requests the help desk for necessary action.
3. The member of help desk team asks the username and details about the problem faced.
4. To authenticate, the member of help desk asks the user's information.
5. For authentication, he compares the information given by the user with the information available with him.
6. Once authentication is completed, the member of help desk either resets the password or forwards the user request to the respective member of help desk, who has the tools and privileges to reset the user's password.
7. Once the password is assigned to the user, he tries to log in.
8. The user should change the password and enter his new password immediately.

But the above process has a number of security loop holes. Some of these loop holes are as under:

1. The authentication of the user is not carried out.
2. For any particular user, the information for authentication may not be available.
3. The information for authentication may be compromised easily.
4. From the members of help desk, anybody may change the password and there is no record of this. The member, who changes the password, knows the user's password. So, the password is no longer truly secret.

The above problem can be solved by ensuring that there should be authentication information available for all the users. The member of help desk logs in the server and then changes the password. Some systems allow the users to reset their own forgotten passwords after reasonable authentication. For this, they have to select a secret question. If the user forgets his password, he has to give the answer of the question selected by him and then the system allows the user to reset his password.

14.9 LIMITATIONS OF INTRUSION DETECTION SYSTEMS

1. Similar IDSs have identical vulnerabilities. So, they cannot detect similar attacks. Thus, the selection of proper IDS model is a serious issue.

2. No IDS has 100% accuracy. It is difficult to measure the sensitivity of IDS. So, making the balance between sensitivity and accuracy of an IDS is critical.
3. For working of IDS, human interference and monitoring are required.
4. Many times, IDS compromises with its security policies.
5. IDS is unable to instantaneous detect, report and respond to an attack.

14.10 CHALLENGES OF INTRUSION DETECTION

Theoretically, IDS looks like a defense tool used to secure our network from the attackers. Speed, accuracy and adaptability are the basic requirements of every IDS. If the attack is not identified in specific time, then the damage caused by the intrusion is large. So, speed is the important requirement of any IDS. IDS should correctly identify the attack. For this, more analysis is required and this increases the computational overhead. If the false alarm rate is more, then many normal attacks are treated as attacks and the alarm is triggered. This blocks normal traffic to enter into the network. The last requirement is adaptability, which means that the IDS should be able to adapt to the new environment and to detect the new attacks. However, IDS has some challenges. The most relevant challenges in the area of intrusion detection that motivate the research work in this area can be summarised as follows:

1. False alarms: One of the most important challenges in intrusion detection is detection accuracy. The high rate of false positives and false negatives intrusion detection tools generated does not help system administrators in simplifying their daily tasks. On the contrary, they may eventually opt for disabling these tools rather than interpreting the alarms reported by the security system that may, in fact, not be reflecting the truth.

2. Performance: Data filtering and screening is a computationally demanding task that is central to intrusion detection. Providing a system with real-time monitoring capabilities implies good search and pattern matching algorithms as well as very fast processing of network traffic, audit trails, etc. that guarantees no data element goes through without being scrutinised. Given the detailed checking needed for intrusion detection, a lot of resources are consumed and the performance of the hosting system ends up being severely affected. As a consequence, the detection task is moved to run offline where its benefits are considerably reduced.

3. Amount of data: The amount of data available within a single host may render the intrusion detection problem intractable. For network-based intrusion detection systems, the amount of traffic going through a gateway, for instance, can be immensely large, making it practically impossible for a security scanner to check every single packet. Similarly, audit trail facilities can generate huge log files, which also limit the possibility of real-time detection. Intrusion detection is a needle-in-the-haystack problem, as the amount of normal activity information may be huge as compared to the pieces of irregular activity that need to be found.

4. Switched networks: Modern networks utilise more intelligent devices to route packets. Many intrusion detection prototypes are based on the ability to configure a

network device in promiscuous mode in order to observe a large number of packets. With the modern network technology, this monitoring approach may no longer be successful. A network switch only forwards a packet to its intended recipient to optimise the bandwidth use. Network-based intrusion detection systems need to be strategically placed in order to protect as many hosts as possible inside a LAN. The filtering functions of a switch, which represent performance advantage, are, in fact, a challenge to the security of networked systems.

5. Encryption: Encrypted data cannot be easily interpreted unless the decryption key is available. The evolution of the Internet has witnessed the development of cryptographic protocols and tools that protect the privacy of data. Encrypted file transfers, encrypted interactive sessions, and encrypted e-mail—all eliminate the possibility of meticulous security analysis. As more companies have become security-conscious, the use of encryption is increasing and will certainly increase even more. Intrusion detection tools are currently unable to deal with this problem, as they are blind to the encrypted information.

6. Security of intrusion detection technologies: By definition, a security tool must be secure; otherwise, it cannot be trusted and its results are worthless. As software products, intrusion detection systems face the same security challenges that the other applications do (e.g., poor implementation and a weak design). In fact, not much attention has been put on the secure design of this type of software. Only a few systems can be considered moderately secure, but the vast majority is prone to many types of security attacks.

7. Reaction to incidents: Although it is not a part of the traditional definition of intrusion detection, response to incidents is a very desirable feature that a few intrusion detection systems have actually tried to implement. Taking action based on incorrect information can have serious consequences and the problem of automated response has remained unexplored due to the low levels of detection accuracy which intrusion detection tools currently display.

Many current approaches have been suggested by the researchers for intrusion detection. They have many advantages. But at the same time, there are some problems with these approaches. Some of the problems with these approaches are given below:

1. These new approaches have high detection rate. But at the same time, they have relatively high false alarm rates. Therefore, most of the normal packets are treated as attacks and alarms are generated for the same. Checking all these normal packets for attacks consumes the resources.
2. The computational complexities of these new approaches are very high. Therefore, practical implementations of these approaches are difficult.
3. Updating the signature database is very important for signature-based intrusion detection systems. If a new attack occurs, the pattern of that attack is not present in the signature database. So, in this case, detection of this new attack is difficult.

CHAPTER 16

Firewall

16.1 INTRODUCTION

Today, the use of Internet is increasing rapidly. Internet is the main medium of attack to the computer system or network. To protect the computer or the network is the main intention of the network administrator. Firewall plays an important and major role in protecting the system. It prevents the unauthorised access to and from the network. Firewall is an effective tool used to protect the network from the attackers. It also allows the internal users to access the outside network via Internet and WAN. The walls of our home restrict the access to any body to enter in our home. In the same way, firewall protects our network. It is a first line of defense. Firewalls are of two types—software and hardware or combinations of both. Firewall observes each and every packet coming inside and going outside the intranet and allows only authenticated packets to do the same. If the packets are not authenticated, then the firewall blocks such packets to cross the firewall. In short, firewall isolates one network from the other network.

To protect the message in communication, different measures are applied. These include encryption of the message, password protection. These measures are not sufficient to prevent the attackers to send different malwares to the computer systems and steal data from the computers. Firewall makes the security strong enough by performing centralised access control and protects the network as well as computers in the network. Routers are used to filter the packets using the IP address of the packets, whereas a firewall stops all packets and filters up through the application layer. Firewalls are generally installed between the network and the Internet. It is also installed in an Intranet to protect one internal network such as one LAN from other LAN.

Firewall performs different security functions as follows:

1. It blocks unauthorised traffic.
2. It forwards the incoming traffic to more reliable internal computer systems.

3. It hides internal computers or networks, which are vulnerable.
4. It hides the information about internal network such as names of the computer system, network topology used, types of network device, etc.
5. It provides strong user authentication.
6. It can serve as a platform for IPSec.

16.2 CHARACTERISTICS OF A FIREWALL

Firewall has the following characteristic:

1. Firewall must act as a gateway for all traffic between the two networks.
2. It allows to pass only authorised traffic that is permitted by local security policy.
3. The firewall itself is protected from any type of penetration.
4. It cannot give assurance about protection from attacks coming from outside network.
5. Risk analysis helps in defining the level of protection for firewall implementation.
6. Firewall implements different local security policy. It
 - (a) defines what type of protection is to be expected from the firewall.
 - (b) specifies the cases when the exceptions are considered.
 - (c) defines the rule for determining authorised and unauthorised traffic.

Firewalls works on simple rules given below:

1. All traffic is denied except that which is specifically authorised.
2. All traffic is allowed except that which is specifically denied.

Firewalls use four techniques to control access and enforce the security policies. These techniques are service control, direction control, user control and behaviour control.

1. **Service control:** The access to any specific types of Internet Services is controlled by this technique. It filters the traffic on the basis of port number, protocol or IP address.
2. **Direction control:** It decides from where the particular service requests should be initiated. It decides whether to allow the request to flow through the firewall or not.
3. **User control:** Depending on the user access, it controls the access to a service.
4. **Behaviour control:** It controls the behaviour of a particular service.

16.3 TYPES OF FIREWALL

Firewalls are categorised into three types—packet filtering firewall, application level gateway and circuit level (proxies) gateways.

16.3.1 Packet Filtering Firewall

The first generation of firewalls is the packet filtering firewalls. It works on the rule and allows the IP packets for incoming and outgoing. Using this rule, the packets

are forwarded or discarded. This firewall works at OSI layers 3 and 4. It is generally designed to filter packets going in both the directions. It tracks the source address and destination address of the packets and TCP/UDP port numbers. The contents of a packet are not analysed by this firewall. The detailed architecture of packet filtering firewall is shown in Figure 16.1.

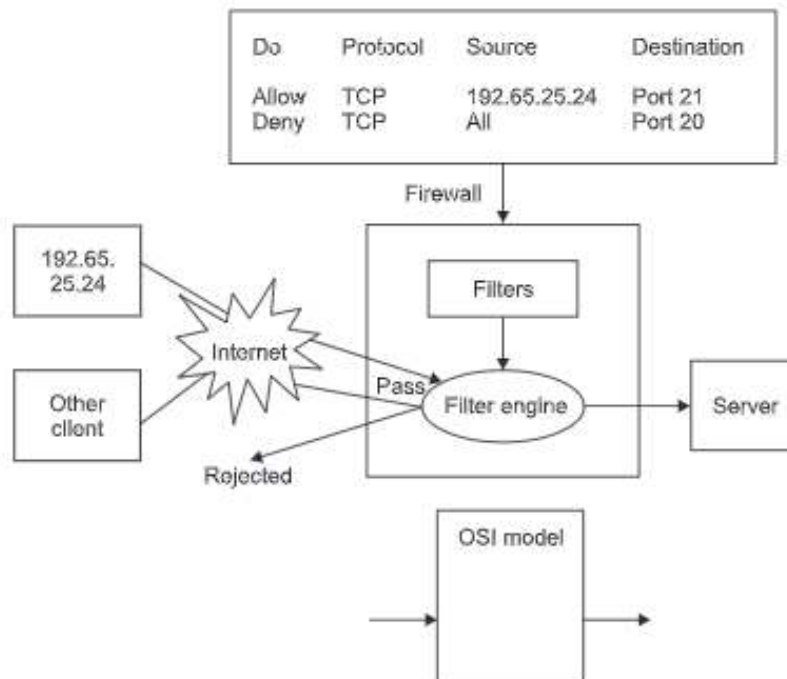


Figure 16.1 Packet filters.

There are various strategies for implementing packet filters. These strategies are based on information contained in a network packet. Some of them are as follows:

1. Source IP address
2. Destination IP address
3. Source and destination port number
4. IP protocol field
5. Interface

Advantages: Its advantages are as follows:

1. Its performance is good.
2. It is very fast.
3. Packet filters are relatively inexpensive.
4. It is transparent to users.
5. The traffic management is good.
6. It is simple.

Disadvantages: Its disadvantages are as follows:

1. Direct connections are allowed between untrusted and trusted hosts.
2. It is vulnerable to spoofing attacks.
3. It has poor scalability.
4. Large port ranges may be opened.
5. Most of these firewalls do not support advanced user authentication schemes.
6. Due to improper configuration, it is susceptible to security breaches.

Some of the attacks against packet filtering firewalls are discussed below:

IP Address Spoofing

The attacker sends the packets from outside the firewall. The packets have different fields. One of the fields is IP address of the source. It gives the address information of the source of the IP packet. But the attacker puts the internal address as a source address. He believes that due to this spoof address, the firewall assumes that this packet is from trusted internal host and allows to pass that packet.

Countermeasure: If the packet coming from outside and has the IP address of the internal host, then discard such packet. This is implemented at the router, which is external to the firewall.

Source Routing Attacks

The attacker assumes that the source routing information is not analysed. So, the route of a packet across the Internet is specified by the source.

Countermeasure: Discard the packets which use this option

Tiny Fragment Attacks

The attacker creates small fragments using the IP fragmentation option. The separate fragment is used for the TCP header information. This design helps in avoiding the filtering rules based on TCP header information. The filtering decision is taken from the first fragment, of a packet and on the basis of this first fragment subsequent fragments, of that packet are allowed or discarded. The attacker takes the advantage of this strategy that only the first fragment of the packet is examined for forwarding the complete packet.

Countermeasure: The preventive measure for this attack is to enforce a rule that the first fragment must hold a predefined minimum amount of the transport header. The filter should remember the packet if the first fragment of the packet is rejected. Then, discard the remaining fragments of the packet.

16.3.2 Application Level Gateways

Packet filtering firewalls is based on address information, so it examines the lower layers of the OSI model. To provide more security, all layers of the OSI model should be examined simultaneously. Application level gateway firewall is useful which provides

this security. It uses server-based programs, known as *proxy server* or *bastion host*. It forwards or rejects the packets by ensuring that the protocol specification is correct.

Proxies accept requests from the external side, examine the request, and then forward the legitimate and trusted requests to the destination host on the other side. This type of firewall makes decisions at all the seven layers of the OSI model. It acts as a mediator for different applications such as e-mail, FTP, etc. It does not permit the client to directly connect to the destination node.

Advantages: Its advantages are as follows:

1. It is configured so that firewall is the only host address that is visible to an outside network.
2. For separate services, separate proxy servers are used.
3. Application gateways support strong user authentication.
4. It provides strong security at the application level.
5. At the application level, it is easy to log and audit all the incoming traffic.
6. It provides strong access control.
7. It is more secure than packet filtering firewall.

Disadvantages: Its disadvantages are as follows:

1. For each application, special proxy is required.
2. Performance is slow.
3. On each connection, there is an additional processing overhead.
4. Sometimes, it is inconvenient for the users.
5. There is a lack of transparency.

Demilitarised Zone (DMZ)

There are two types of firewall based on its locations in the network—Internal firewall and external firewall. An *internal firewall* protects the entire network of an organisation. An *external firewall* is installed at the boundary of local or organisation network. It is located inside the boundary router. The region between the two firewalls is called *demilitarised zone* or *DMZ*. Many network devices are located between these two firewalls. It includes devices which are allowed to access from external networks. These devices can be accessed externally, but protected from vulnerability. The systems which are located in DMZ require connectivity with the external network such as domain name server (DNS), an e-mail server or web sites. The protection and access control to these systems are provided by the external firewall. The external firewall provides basic security to the entire network. The internal firewall has strong filter capabilities as compared to external firewall. This provides strong security to the servers and workstations from the external attacks. The internal firewall also provides two-way protection. Initially, it protects the entire internal network from attacks launched from DMZ systems that originate from worms, bots or other malicious software. Then, the internal firewall protects system located at DMZ area from internal attack. To protect different portions of the large network from each other, multiple internal firewalls are used.

16.3.3 Circuit Level Gateways

Circuit level gateway is a type of firewall and it is a stand-alone system. It validates connections and then allows the exchange of data. It also works as per the defined rules similar to packet filter. Circuit level gateway cannot route the packets. The connections are allowed or discarded based on these rules. So, circuit level gateway establishes the connections between the source and the destinations. It focuses on the TCP/IP layer. It is more secure than packet filters. This firewall is installed between the router and the external network such as Internet. The actual address is hidden from the external users because only the address of the proxy is transmitted. Circuit level gateway provides services for many different protocols. It can be adapted as per the requirement to provide greater variety of communications.

Advantages: Its advantages are as follows:

1. It is transparent to the users.
2. This is excellent for relaying outbound traffic.

Disadvantages: Its disadvantages are as follows:

1. It is slower than packet filtering firewall.
2. Inbound traffic is risky.

16.4 BENEFITS OF A FIREWALL

1. Increased ability to enforce network security standards/policies (e.g. id. undocumented systems)
2. Centralisation of internetwork audit capability (audit of in/out-bound traffic)

16.5 LIMITATIONS OF A FIREWALL

1. It cannot protect those attacks that bypass the firewall.
2. It cannot protect the network against the internal attacks.
3. An internal firewall that separates the different parts of a network cannot protect against wireless communications among local computer systems on different sides of the internal firewall.
4. Different devices such as laptop or portable storage device may be used and infected outside the network, and then used internally.

16.6 FIREWALL ARCHITECTURES

Generally, firewall is implemented on a single machine. The stronger firewalls have multiple components. In this section, different firewall architectures are discussed.

16.6.1 Dual-Homed Host Architecture

The simple firewall architecture uses dual-homed host architecture. It is a computer system, which has separate network interfaces for minimum two networks. This host

computer can act as a router between the networks. This routing function should be disabled when it is used in firewall architecture. Therefore, the host computer isolates the networks from each other, but it can see traffic on all the networks. Therefore, IP packets from one network, for example, external network or Internet are not directly routed to the other network, for example, internal network. Systems inside the firewall, i.e., internal network can communicate with the dual-homed host via one interface and systems outside the firewall, i.e., on the Internet can communicate with the dual-homed host via another interface. However, these systems cannot communicate directly with each other. The traffic between these two networks is completely blocked.

The dual-homed host firewall architecture is simple. The architecture for dual-homed host is shown in Figure 16.2.

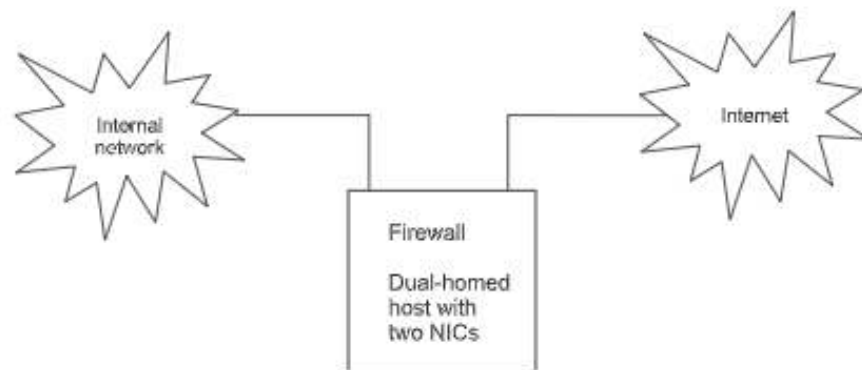


Figure 16.2 Dual-homed host architecture.

Dual-homed hosts can provide a very tight control on the traffic flowing on the network. Not a single packet is allowed without the consent of dual-homed hosts. If the rule is designed that does not allow the packets to flow between the internal and external network then all the packets are blocked by the dual-homed hosts. In this architecture, dual-homed host itself is critical for the network security. It provides services by only proxying them. Proxying is much less challenging, but it may not be available for all services the users are interested in.

16.6.2 Screened Host Architecture

The screened host architecture provides services from a host that is attached to the internal network only. For this, a separate router is used. In this architecture, packet filter is used to provide the main security. The screened host architecture is shown in Figure 16.3. The required applications are provided by the bastion host that sits on the internal network. The packet filtering rules are configured in such a way that the bastion host is the only host on the internal network that is accessible from the Internet. Even then, only certain types of connections are allowed. If any external computer wants access to the internal computer or wants to use some services, it has to first connect to this host. The bastion host is responsible to maintain a high level of security for the host.

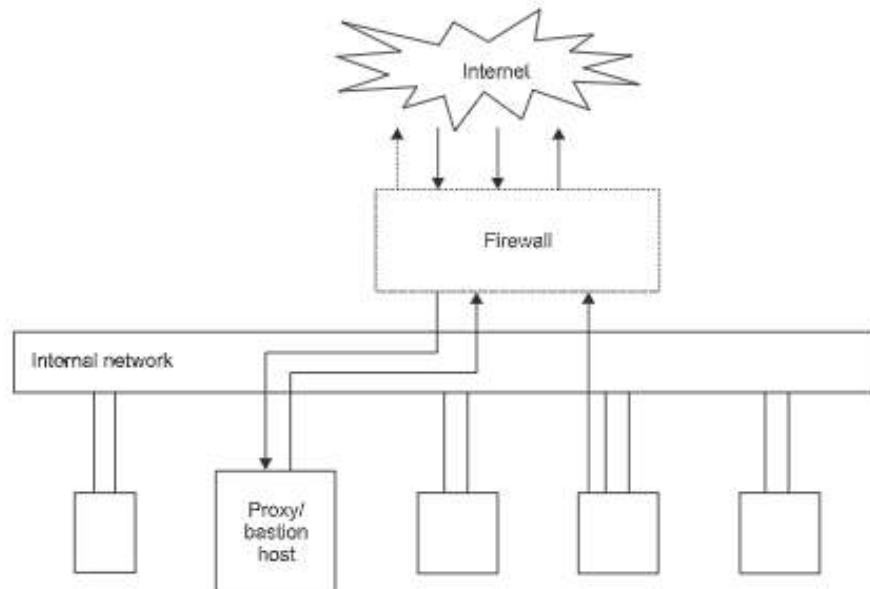


Figure 16.3 Screened host architecture.

The major disadvantage of the screened host architecture is that if an attacker becomes successful to attack into the bastion host then the network security between the bastion host and the rest of the internal hosts is completely collapsed. In screened host architecture, the router still works as the first line of defense. Filtering and access control for all the packets is performed at the router. The router allows entering only that traffic that the rules explicitly identify. It restricts other incoming connections to the host. Therefore, the router may also be a single point of failure. If the security of the router is compromised, the complete network is available to an attacker.

Nevertheless, screened host architecture is popular due to the following reasons:

1. It allows companies to easily enforce various security policies in different directions.
2. It is relatively easy to implement.

16.6.3 Screened Subnet Architecture

The screened subnet architecture is functionally similar to the screened host architecture. But it provides some extra security. It gives strength to the security of the firewall by adding a perimeter network, which helps in further isolating the internal network from the Internet. The most vulnerable computer in the network is bastion host. In screened host architecture, the internal network is wide open to attack through the bastion host. In this case, bastion host is a very attractive target for the attacker. If the attacker is able to successfully break the security of bastion host, then the complete internal network is open for the attacker. To reduce the impact of attack, the solution is to

isolate the bastion host on perimeter network. In this case, the attacker may be able to get partial access, but the complete internal network is not available to the attacker.

In screened subnet architecture, two screening routers are used. Each of these routers are connected to the perimeter net. One router is installing between the internal network and the perimeter net, while the other router is to install between the external network and the perimeter net. So, there are two routers are used to protect the network. To break into the internal network, the attacker has to pass both the routers. If the attacker is able to break the bastion host (external router), then also due to the internal router, the internal network is not accessible to the attacker. And to compromise the internal network, there is no single point that will be vulnerable.

To provide more security, a number of perimeter nets can be used between the Internet or outside world and the interior network. The less trusted services are kept in the outer perimeter net so that if the attacker is able to break the outer systems, he has to work hard to break the internal systems. This can be achieved by using different configurations for each perimeter net. If all the systems have same rules, then this additional perimeter has no use to provide the additional security. The screened subnet architecture is shown in Figure 16.4.

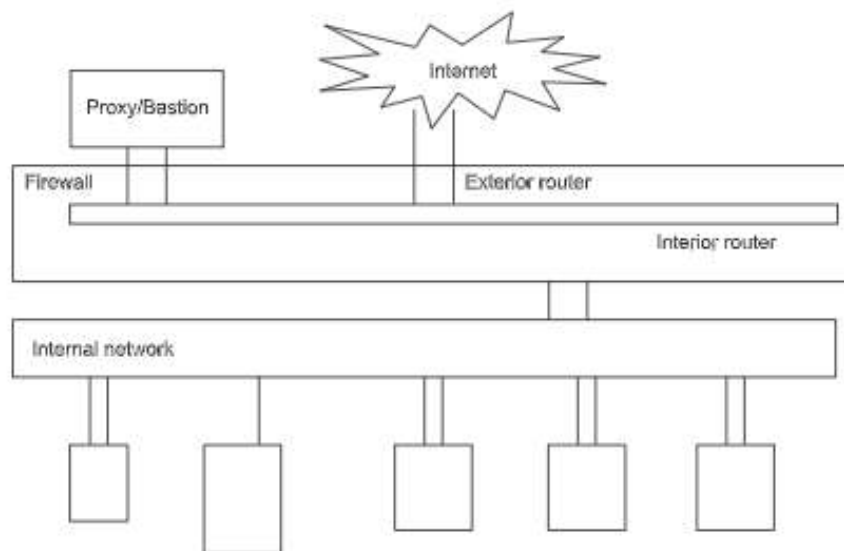


Figure 16.4 Screened subnet architecture (using two routers).

Perimeter Network

A *perimeter network* is a firewall, which is installed between a private network and Internet. It controls all the traffic between the private network and the Internet. The perimeter network is an additional network between the external network and the internal network. It provides additional security to the internal network. If an attacker is able to break the outer security of the firewall, then the perimeter net protects the internal network by providing an extra layer of security between the attacker and the internal network.

If the attacker breaks a bastion host on the perimeter net, then the use of perimeter net allows him to snoop only the traffic on that specific network. There are two types of traffic on the perimeter net.

1. To or from the bastion host
2. To or from the Internet

As the traffic from two internal hosts does not pass from the perimeter net, it is safe from the attacker, though the security of bastion is broken. But the traffic to and from the bastion host or the traffic to and from the external world remains visible to the attacker. So, care should be taken during the configuration of the firewall because this traffic is not itself confidential enough. So, if the attacker is able to capture it, it does not make any harm to the network.

Bastion Host/Proxy

In the screened subnet architecture, a host is attached to the perimeter net. This host is called *bastion host*. Any connection from the external network is first contacted to this host. For example, for e-mail session coming from external network sessions to deliver electronic mail to the site

Outbound services, i.e., services from users on the internal network to any servers on the external network are handled in the following ways:

1. Install the packet filtering firewalls on both routers. This allows internal clients to establish direct connections with the external servers.
2. Install proxy servers to run on the bastion host. This allows internal clients to establish indirect connections with the external servers. Packet filtering firewall can be used to establish the connections between the internal clients with the proxy servers on the bastion host and vice versa. This prohibits direct communication between the internal clients and external network.

In any of the above case, the packet filtering firewall is used, which allows the bastion host to connect to the Internet. It also allows the bastion host to accept the connections from the hosts on outside world.

Interior Router

The interior router is also called *choke router*. It protects the internal network from the outside world or external network as well as from the perimeter net. It blocks most of the packets using packet filtering for the firewall. It allows selected services on the internal network to any servers on the external network. Selected services reduce the number of computers, which can be compromised from the bastion host. These services are provided using packet filtering rather than proxies. These services might include FTP, Telnet and others, as per needs. It is not necessary that the services are allowed by the interior router between the bastion host and the internal network, and the services between the internal network and the Internet are same.

Exterior Router

Theoretically, exterior router is also called *access router* in firewalls. It protects the perimeter net as well as the internal net from the outside world such as Internet.

Practically, exterior routers allow almost all packets outbound from the perimeter net. Exterior routers perform packet filtering on a small scale. There would be same rules for interior and exterior router. If any of the rule has an error, it allows the attacker to get access in the network. This error is present on both the routers.

The exterior router is provided by an Internet provider and it limits the access of the internal users. An Internet provider puts few general packet filtering rules. One of the special rules on the exterior router is related to protect the computers on the perimeter network. A user should not trust on this exterior router as one trusts on interior router. Many rules on the interior are put on the exterior router, which prevent insecure traffic between the internal host and the Internet. The interior router allows the internal hosts to send some protocols to talk to the bastion host. This is used to support the proxy services. The exterior router allows those protocols through as long as they come from the bastion host. It is expected that these rules on the exterior router provide an extra security, but theoretically, it does not happen. This is because these packets are already blocked by the interior router. If such packets do exist, that means the interior router has either failed or an unexpected host is connected to the perimeter network. The main task of the exterior router is not to allow the packets coming from the internet which have forged source addresses, i.e., the source address is shown as the address of the system within the internal network. For interior router, it is possible to do this, but cannot tell that the packets are forged.

A comparison between packet filtering firewalls and application level gateways is shown below:

<i>Packet Filtering Firewalls</i>	<i>Application Level Gateways</i>
It is the simplest firewalls.	It is complex firewalls.
No change in the software is required for performing its job.	No change in the software is required for performing its job, but it is visible to the user.
It can see only addresses and type of service protocol.	It can see full data of packet.
Auditing of this filter is difficult.	Auditing of this filter is simple.
Configuration is difficult due to complex rules.	Proxies can be used to substitute complex addressing rules.
Screen is based on connection rules.	Screen is based on behaviour of proxies.
It is less powerful.	It is more powerful.

16.7 TRUSTED SYSTEM

A system that you have no choice but to trust is known as *trusted system*. The security of the system depends on the success of the system. If the trusted system fails, then it will compromise the security of the entire system. Therefore, there should be minimum number of trusted components in a system. Trusted system should provide security, integrity, reliability and privacy.

16.7.1 Trusted Systems in Policy Analysis

In trusted systems, some conditional prediction about the behaviour of users or elements

within the system has been determined prior to authorising access to resources within the system. Many systems are trusted systems if they have the following options:

1. The probability of threat or risk analysis is calculated, which is used to access trust for taking the decision before authorisation.
2. To insure the behaviour within the system, the deviation analysis is used.

16.8 ACCESS CONTROL

One of the security services is access control. In this service, users are identified and given certain privileges to computer systems or resources. The access to the data or computer is controlled. If the user misuses the privileges given to him, then the access to such user's is denied. For example, suppose you give access to your friend to your own computer by giving him separate user account. But you find out that your friend misuses your system by some other activities. Then, the administrator of the computer can delete that user account so that he cannot access your computer any more. Protecting the private and confidential information from the unauthorised users on the computer, which is connected to the network, particularly to the Internet, is most important. Access control mechanism allows to protect this private and confidential information from unauthorised users. In access control mechanism, identification of the users is carried out first using authentication algorithm. It includes keeping the records and timestamp of all the requests, which is helpful during the audit of the system.

16.8.1 Objectives of Access Control

Access control has different objectives, which include preserve and protect the confidentiality, availability and integrity of data or information, systems and resources.

Confidentiality is the process of protection of data or information from unauthorised disclosure. It ensures that information is accessible only to the authorised person. In cryptography and network security data, confidentiality is done by using encryption techniques.

Integrity means assurance that data received is same as it is sent by an authorised sender, i.e., in transmission, there is no change in the data. This modification or change includes deletion, modification and creation of new information in the data. We can use hashing algorithms like MD5, SHA to check the integrity of the message. Integrity protects the data from unauthorised modification.

Availability is the measure to which a system or information is accessible and usable upon request by an authorised user at any particular time. Availability means a functioning condition of a system at any particular instance. For example, access to a system or information should not be prevented to the legitimate users.

16.8.2 Types of Access Control

Access control can be categorised into four classes as follows:

1. Discretionary access control systems: In discretionary access control system, the owner of the information has the right to decide about who can read, write and execute the information. Using discretionary access control, users can create and modify files in their own directories.

2. Mandatory access control systems: In mandatory access control systems, the creator of the information has no right to decide about who can access or modify it. The access to any information is decided by the administrators and authorities. These systems are used in military applications, financial application, etc.

3. Role-based access control systems: In role-based access control systems, the users are permitted to access systems and information based on their role within the organisation. This type of access can be allowed to individuals or groups of people.

4. Rule-based access control systems: In rule-based access control systems, some rules are configured and predefined. The decisions about whom to allow the access to the systems and information are based on these rules. Rules are designed such that all the users from a particular network, host, domain or IP address are allowed to use the information or system or resources.

SUMMARY

In this chapter, we have discussed about the firewall and types of firewall. Firewall is an effective tool used to protect the network from the attackers. Dual-homed host architecture is a computer system, which has separate network interfaces for minimum two networks. This host computer can act as a router between the networks. Dual-homed hosts can provide a very tight control on the traffic flowing on the network. Not a single packet is allowed without the consent of dual homed hosts. The screened host architecture provides services from a host that is attached to the internal network only. For this, a separate router is used. The screened subnet architecture gives strength to the security of the firewall by adding a perimeter network which helps in further isolating the internal network from the Internet.

EXERCISES

- 16.1 What is a firewall?
 - 16.2 What is the main function of a firewall?
 - 16.3 List the protections associated with firewall systems.
 - 16.4 Discuss the characteristics of a firewall.
 - 16.5 What are the three main types of firewall?
 - 16.6 Explain packet filtering firewalls.
 - 16.7 Explain application level gateways.
 - 16.8 Explain circuit level gateways.
 - 16.9 List the advantages and disadvantages of application level gateways.
 - 16.10 What are the benefits of firewalls?
-