Network security

1. Cryptography

The word **cryptography** has come from a Greek word, which means *secret writing*. For private communication through public network, cryptography plays a very crucial role. The role of cryptography can be illustrated with the help a simple model of cryptography as shown in Fig. 1.1.1. The message to be sent through an unreliable medium is known as **plaintext**, which is encrypted before sending over the medium. The encrypted message is known as **ciphertext**, which is received at the other end of the medium and decrypted to get back the original plaintext message.

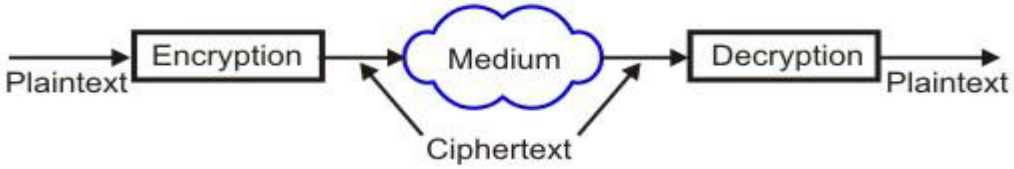


Fig. 1.1.1 A simple cryptography model

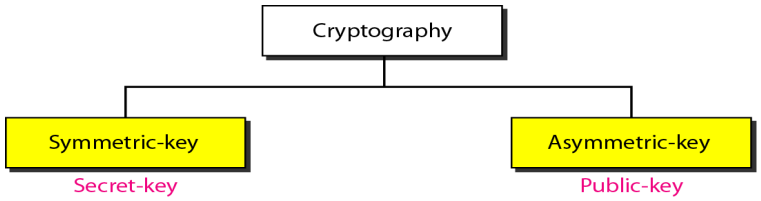
Cipher

We refer to encryption and decryption algorithms as ciphers. The term *cipher* is also used to refer to different categories of algorithms in cryptography.

Key

A key is a number (or a set of numbers) that the cipher, as an algorithm, operates on. To encrypt a message, we need an encryption algorithm, an encryption key, and the plaintext. These create the ciphertext. To decrypt a message, we need a decryption algorithm, a decryption key, and the ciphertext. These reveal the original plaintext.

Cryptography algorithms (ciphers) are divided into two groups: symmetric-key (also called secret-key or private key) cryptography algorithms and asymmetric (also called public-key) cryptography algorithms.



1.1 Symmetric Key Cryptography

In symmetric-key cryptography, the same key is used by both parties. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and the corresponding decryption algorithm to decrypt the data (as shown in Fig 1.1.2). The algorithm used to decrypt is just the inverse of the algorithm used for encryption. For example, if addition and division is used for encryption, multiplication and subtraction are to be used for decryption.

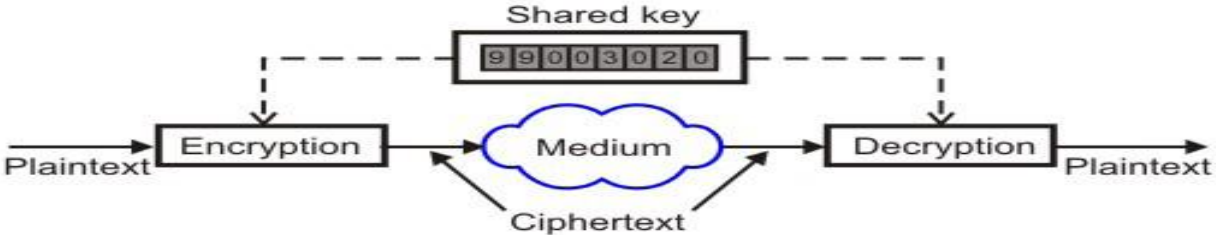


Fig 1.1.2 Symmetric-key cryptography

Symmetric key cryptography algorithms are simple requiring lesser execution time. As a consequence, these are commonly used for long messages. However, these algorithms suffer from the following limitations:

- Requirement of large number of unique keys. For example for n users the number of keys required is n(n-1)/2.
- Distribution of keys among the users in a secured manner is difficult.

1.1.1 Monoalphabetic Substitution

One simple example of symmetric key cryptography is the *Monoalphabetic substitution*. In this case, the relationship between a character in the plaintext and a character in the ciphertext is always one-to-one. An example Monoalphabetic substitution is the Caesar cipher. As shown in Fig. 1.1.3, in this approach a character in the ciphertext is substituted by another character shifted by three places, e.g. A is substituted by D. Key feature of this approach is that it is very simple but the code can be attacked very easily.

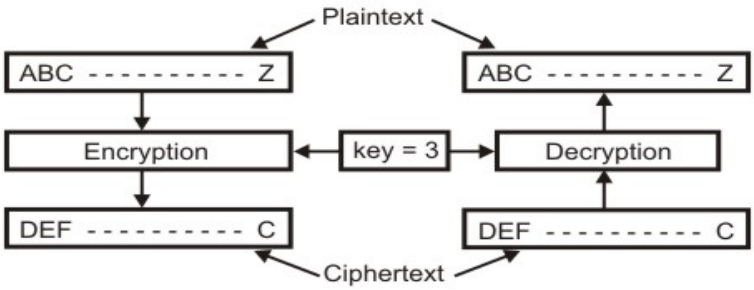


Fig. 1.1.3 The Caesar cipher

1.1.2 Polyalphabetic Substitution

This is an improvement over the Caesar cipher. Here the relationship between a character in the plaintext and a character in the ciphertext is always one-to-many. Key feature of this approach is that it is more complex and the code is harder to attack successfully. Example of polyalphabetic substitution is the Vigenere cipher. In this case, a particular character is substituted by different characters in the ciphertext depending on its position in the plaintext.

1.1.3 Transpositional Cipher

The transpositional cipher, the characters remain unchanged but their positions are changed to create the ciphertext. Figure 1.1.4 illustrates how five lines of a text get modified using transpositional cipher. The characters are arranged in two-dimensional matrix and columns are interchanged according to a key is shown in the middle portion of the diagram. The key defines which columns are to be swapped. As per the key shown in the figure, character of column is to be swapped to column 3, character of column 2 is to be swapped to column 6, and so on. Decryption can be done by swapping in the reverse order using the same key.

Transpositional cipher is also not a very secure approach. The attacker can find the plaintext by trial and error utilizing the idea of the frequency of occurrence of characters.

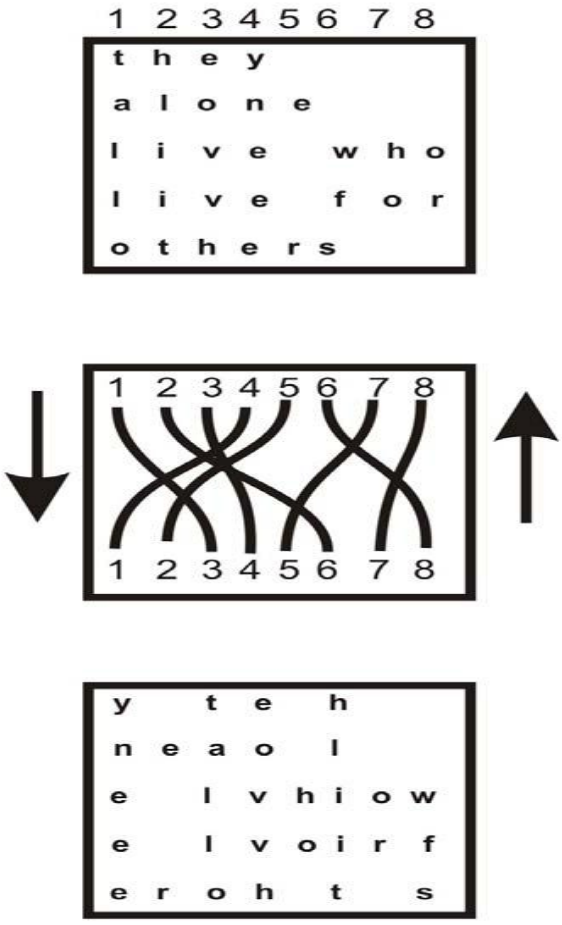


Figure 1.1.4 Operation of a transpositional cipher

1.1.4 Block Ciphers

Block ciphers use a block of bits as the unit of encryption and decryption. To encrypt a 64-bit block, one has to take each of the 264 input values and map it to one of the 264 output values. The mapping should be one-to-one. Encryption and decryption operations of a block cipher are shown in Fig. 1.1.5. Some operations, such as permutation and substitution, are performed on the block of bits based on a key (a secret number) to produce another block of bits. The permutation and substitution operations are shown in Figs 8.1.7 and 8.1.8, respectively. In the decryption process, operations are performed in the reverse order based on the same key to get back the original block of bits.

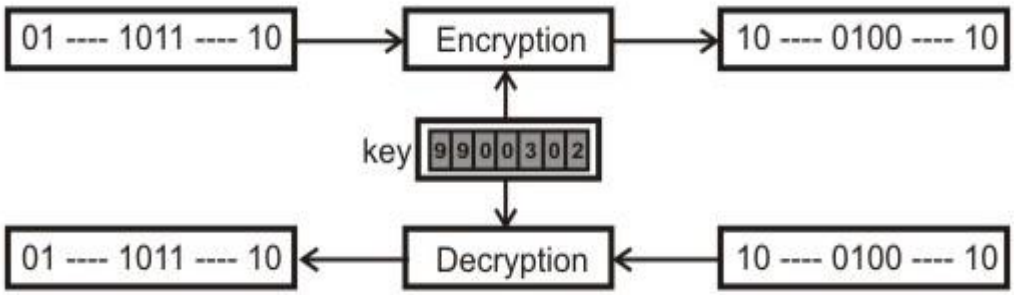


Fig. 1.1.5 Transformations in Block Ciphers

Permutation: As shown in Fig. 1.1.6, the permutation is performed by a permutation box at the bit-level, which keeps the number of 0s and 1s same at the input and output. Although it can be implemented either by a hardware or a software, the hardware implementation is faster.

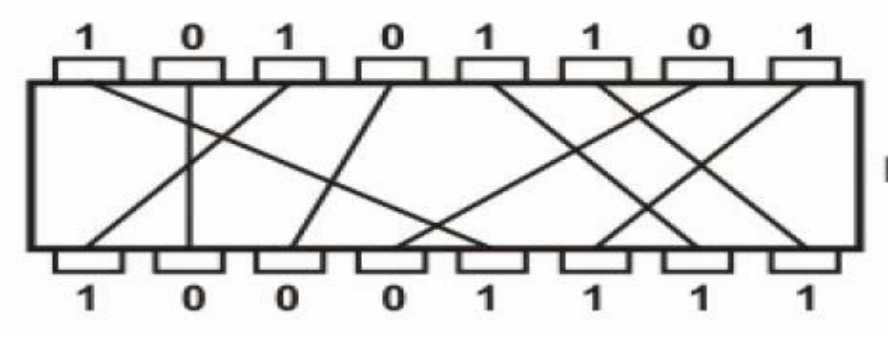


Fig. 1.1.6 Permutation operation used in Block Ciphers

Substitution: As shown in Fig. 1.1.7, the substitution is implemented with the help of three building blocks – a decoder, one p-box and an encoder. For an n-bit input, the decoder produces an 2n bit output having only one 1, which is applied to the P-box. The P-box permutes the output of the decoder and it is applied to the encoder. The encoder, in turn, produces an n-bit output. For example, if the input to the decoder is 011, the output of the decoder is 00001000. Let the permuted output is 01000000, the output of the encoder is 011.

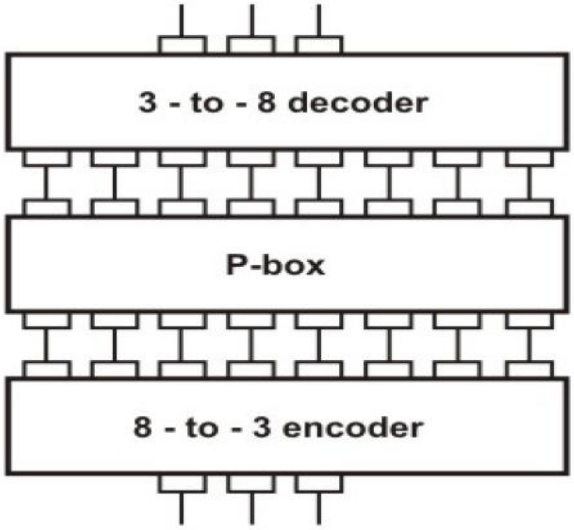


Fig. 1.1.7 Substitution operation used in Block Cipher

1.1.5 Data Encryption Standard (DES)

One example of a complex block cipher is the Data Encryption Standard (DES). DES was designed by IBM and adopted by the U.S. government as the standard encryption method for nonmilitary and nonclassified use. The algorithm encrypts a 64-bit plaintext block using a 64-bit key, as shown in Figure 1.1.8

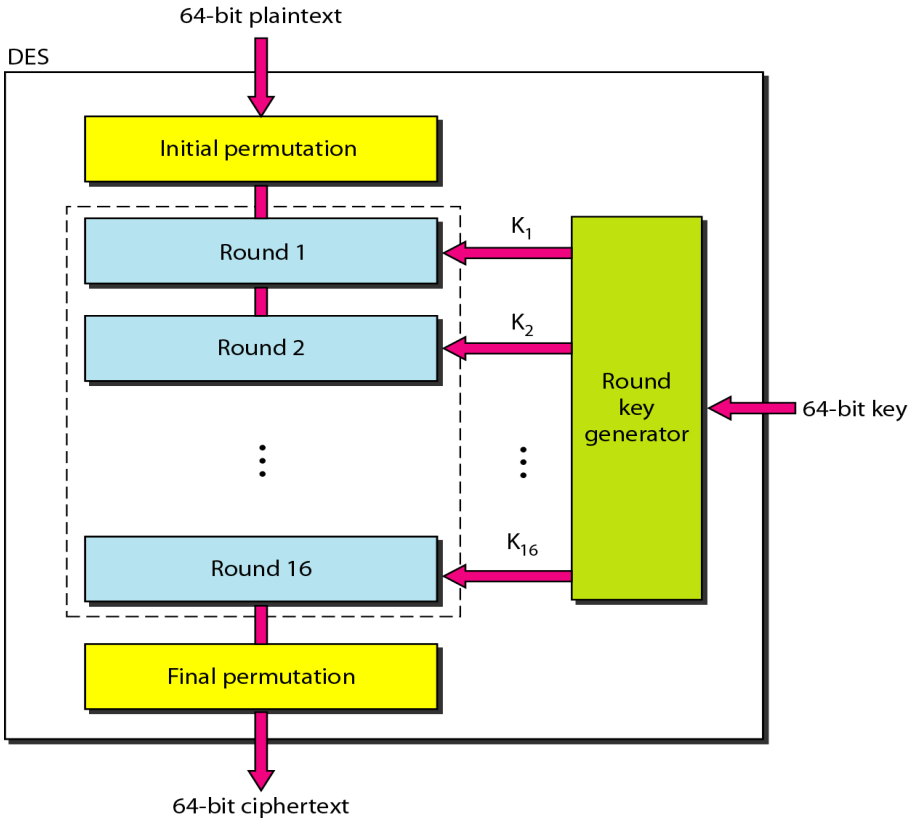


Figure 1.1.8(a) DES

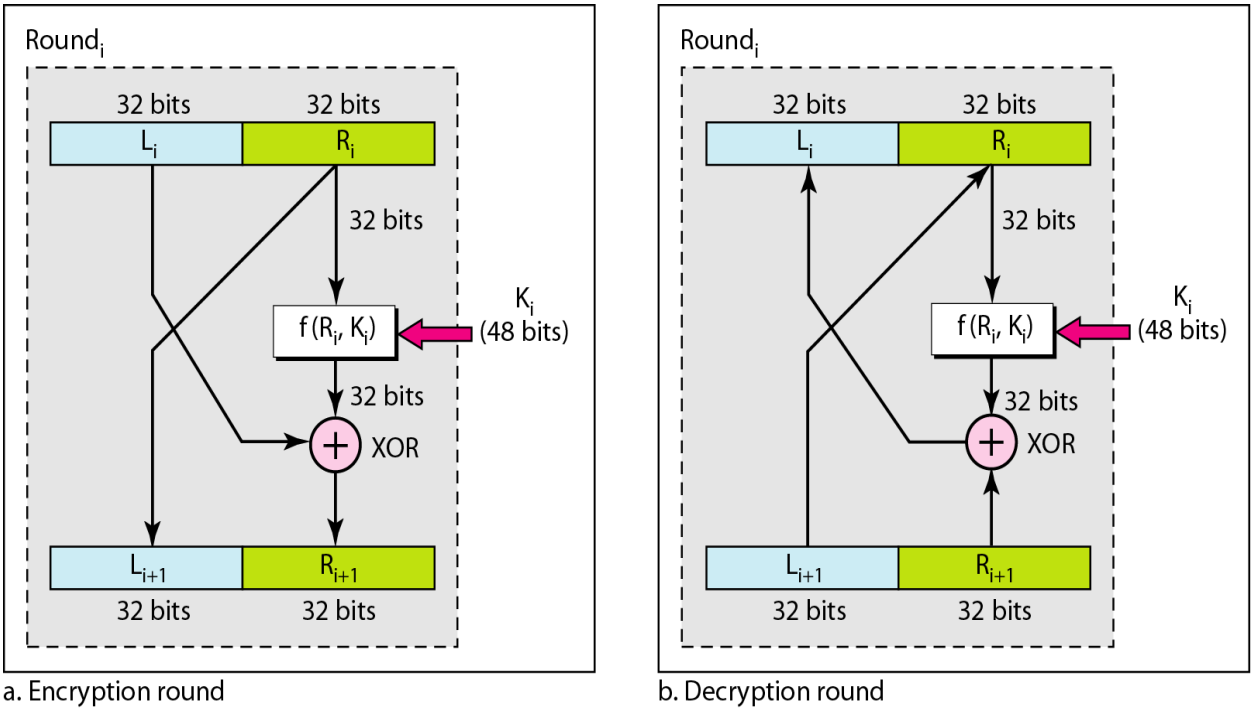


Figure 1.1.8(b) One round in DES ciphers

DES has two transposition blocks (P-boxes) and 16 complex round ciphers (they are repeated). Although the 16 iteration round ciphers are conceptually the same, each uses a different key derived from the original key. The initial and final permutations are keyless straight permutations that are the inverse of each other. The permutation takes a 64-bit input and permutes them according to predefined values. Each round of DES is a complex round cipher, as shown in Figure 1.1.8(b). Note that the structure of the encryption round ciphers is different from that of the decryption one.

DES Function: The heart of DES is the **DES function.** The DES function applies a 48-bit key to the rightmost 32 bits *Ri* to produce a 32-bit output. This function is made up of four operations: an XOR, an expansion permutation, a group of S-boxes, and a straight permutation.

1.2 Public key Cryptography

In public key cryptography, there are two keys: a private key and a public key. The public key is announced to the public, where as the private key is kept by the receiver. The sender uses the public key of the receiver for encryption and the receiver uses his private key for decryption as shown in Fig. 1.2.1

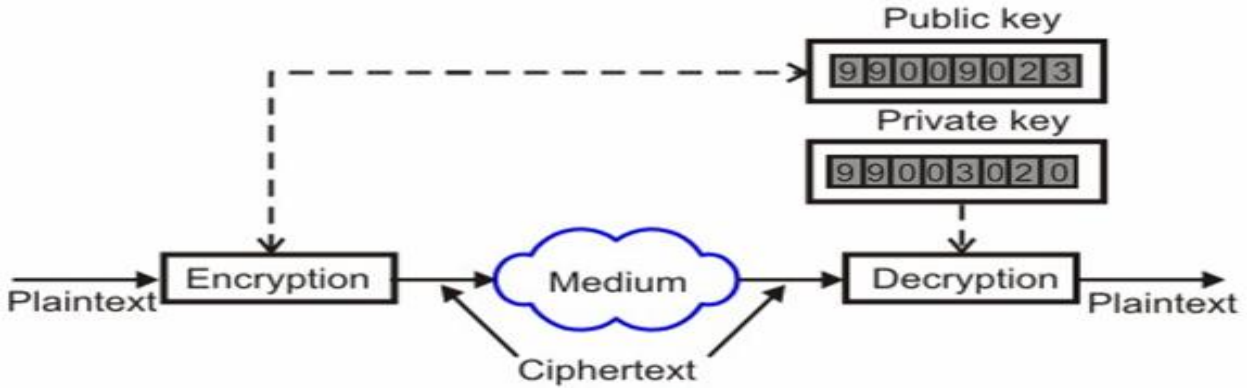


Fig. 1.2.1 Public key Cryptography

Advantages:

- The pair of keys can be used with any other entity
- The number of keys required is small

Disadvantages:

- It is not efficient for long messages
- Association between an entity and its public key must be verified

1.2.1 RSA

The most common public key algorithm is RSA, named for its inventors Rivest, Shamir, and Adleman (RSA). It uses two numbers, e and d, as the public and private keys, as shown in Figure 1.2.2.

Selecting Keys

1. Bob use the following steps to select the private and public keys:
2. Bob chooses two very large prime numbers p and q. Remember that a prime number is one that can be divided evenly only by 1 and itself.
3. Bob multiplies the above two primes to find n, the modulus for encryption and decryption. In other words, n= p X q.
4. Bob calculates another number α= (p -1) X (q - 1).
5. Bob chooses a random integer e. He then calculates d so that d x e= 1 mod α
6. Bob announces e and n to the public; he keeps α and d secret.
7. In RSA, e and n are announced to the public; d and <I> are kept secret.

Encryption

Anyone who needs to send a message to Bob can use nand e. For example, if Alice needs to send a message to Bob, she can change the message, usually a short one, to an integer. This is the plaintext. She then calculates the ciphertext, using e and n.

$$C=P^e \pmod{n}$$

Alice sends C, the ciphertext, to Bob.

Decryption

Bob keeps α and d private. When he receives the ciphertext, he uses his private key d to decrypt the message:

$$P= C^d \pmod{n}$$

Restriction

For RSA to work, the value of P must be less than the value of n. If P is a large number, the plaintext needs to be divided into blocks to make P less than n.

1.3 Digital Signature

There are two alternatives for Digital Signature:

- Signing the entire document
- Signing the digest

In the first case the entire document is encrypted using private key of the sender and at the receiving end it is decrypted using the public key of the sender as shown in Fig. 1.3.1. For a large message this approach is very inefficient. In the second case a miniature version of the message, known as *digest*, is encrypted using the private key of the sender and then the signed digest along with the message is sent to the receiver as shown in Fig. 1.3.2. The receiver decrypts the signed digest using the public key of the sender and the digest created using the received message is compared with the decrypted digest shown in Fig. 1.3.3. If the two are identical, it is assumed that the sender is authenticated. This is somewhat similar to error detection using parity bit.

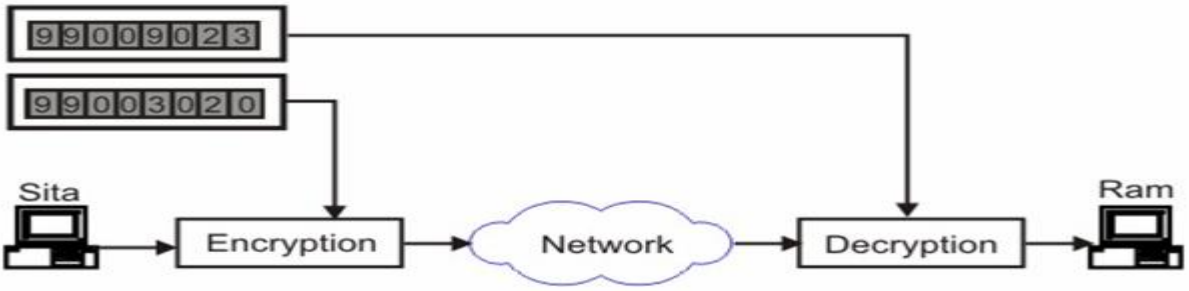


Fig. 1.3.1 Authentication by signing he whole document

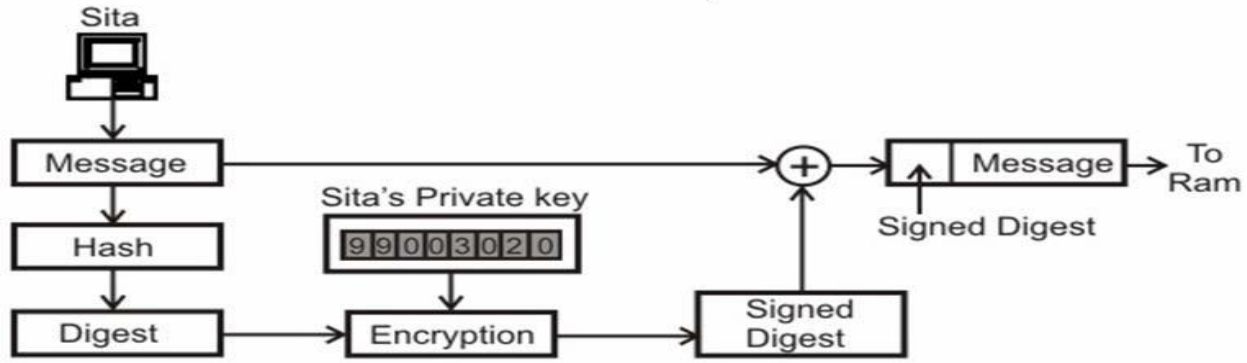


Fig. 1.3.2 Sender site for authentication by signed digest

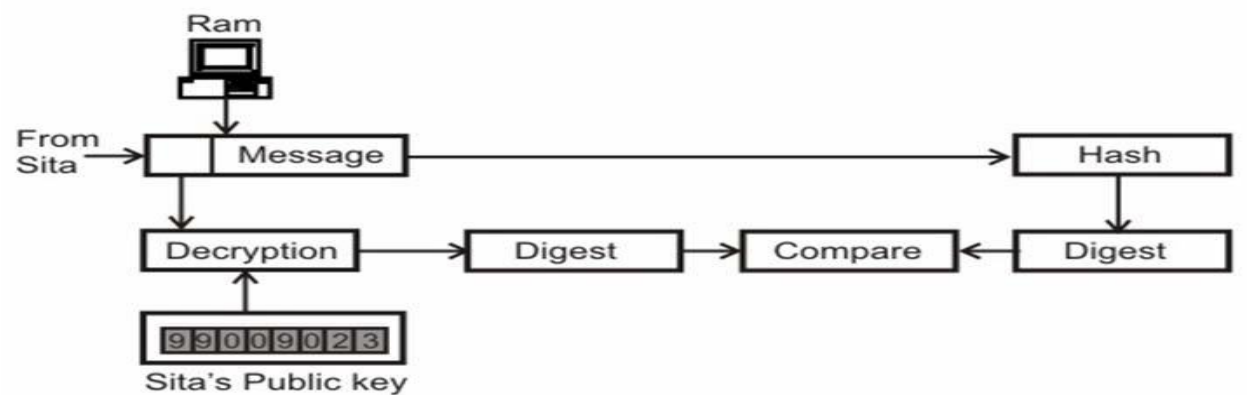


Fig. 1.3.3 Receiver site for authentication by signed digest

Some key features of this approach are mentioned below:

- Digital signature does not provide privacy
- Hash function is used to create a message digest
- It creates a fixed-length digest from a variable-length message
- Most common Hash functions:
  MD5 (Message Digest 5): 120-bit
  SHA-1 (Secure Hash algorithm 1): 160-bit
- Important properties:
  One-to-One
  One-way

1.4 Firewall

To control access to a system, we need firewalls. A **firewall** is a device (usually a router or a computer) installed between the internal network of an organization and the rest of the Internet. It is designed to forward some packets and filter (not forward) others. Figure 1.4.1 shows a firewall. For example, a firewall may filter all incoming packets destined for a specific host or a specific server such as HTTP. A firewall can be used to deny access to a specific host or a specific service in the organization.

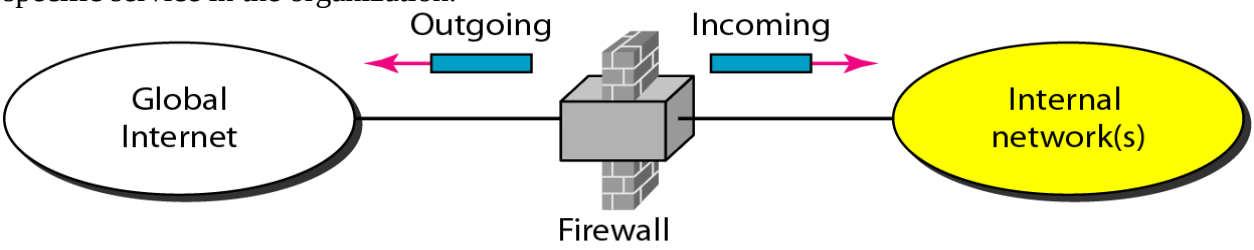
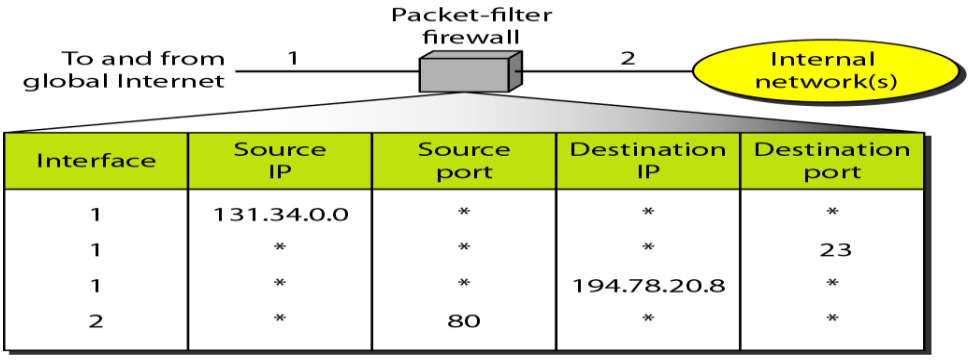


Figure 1.4.1 Firewall

A firewall is usually classified as a packet-filter firewall or a proxy-based firewall.

1.4.1 Packet-Filter Firewall

A firewall can be used as a packet filter. It can forward or block packets based on the information in the network layer and transport layer headers: source and destination IP addresses, source and destination port addresses, and type of protocol (TCP or UDP). A packet-filter firewall is a router that uses a filtering table to decide which packets must be discarded (not forwarded). Figure 1.4.2 shows an example of a filtering table for this kind of a firewall.



Interface	Source IP	Source port	Destination IP	Destination port
1	131.34.0.0	*	*	*
1	*	*	*	23
1	*	*	194.78.20.8	*
2	*	80	*	*

Figure 1.4.2 Packet-filter firewall

According to Figure 32.23, the following packets are filtered:

1. Incoming packets from network 131.34.0.0 are blocked (security precaution). Note that the * (asterisk) means "any."
2. Incoming packets destined for any internal TELNET server (port 23) are blocked.
3. Incoming packets destined for internal host 194.78.20.8 are blocked. The organization wants this host for internal use only.
4. Outgoing packets destined for an HTTP server (port 80) are blocked. The organization does not want employees to browse the Internet.

1.4.2 Proxy Firewall

The packet-filter firewall is based on the information available in the network layer and transport layer headers (IP and TCP/UDP). However, sometimes we need to filter a message based on the information available in the message itself (at the application layer). As an example, assume that an organization wants to implement the following policies regarding its Web pages: Only those Internet users who have previously established business relations with the company can have access; access to other users must be blocked. In this case, a packet-filter firewall is not feasible because it cannot distinguish between different packets arriving at TCP port 80 (HTTP). Testing must be done at the application level (using URLs).

One solution is to install a proxy computer (sometimes called an application gateway), which stands between the customer (user client) computer and the corporation computer shown in Figure 1.4.3.

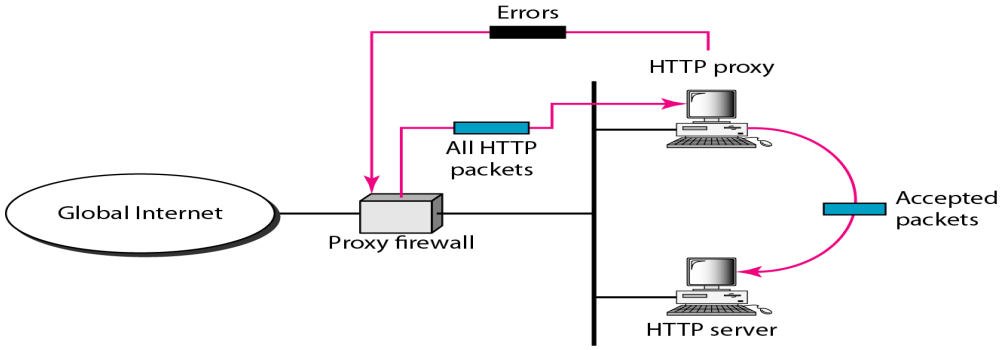


Figure 1.4.3. Proxy firewall

When the user client process sends a message, the proxy firewall runs a server process to receive the request. The server opens the packet at the application level and finds out if the request is legitimate. If it is, the server acts as a client process and sends the message to the real server in the corporation. If it is not, the message is dropped and an error message is sent to the external user. In this way, the requests of the external users are filtered based on the contents at the application layer.