UNIT 5

Transport Layer - Services - Connection Management - Addressing, Establishing and Releasing a Connection - Simple Transport Protocol - Internet Transport Protocols (ITP) - Network Security: Cryptography.

Introduction:

The network layer provides end-to-end packet delivery using data-grams or virtual circuits. The transport layer builds on the network layer to provide data transport from a process on a source machine to a process on a destination machine with a desired level of reliability that is independent of the physical networks currently in use. It provides the abstractions that applications need to use the network.

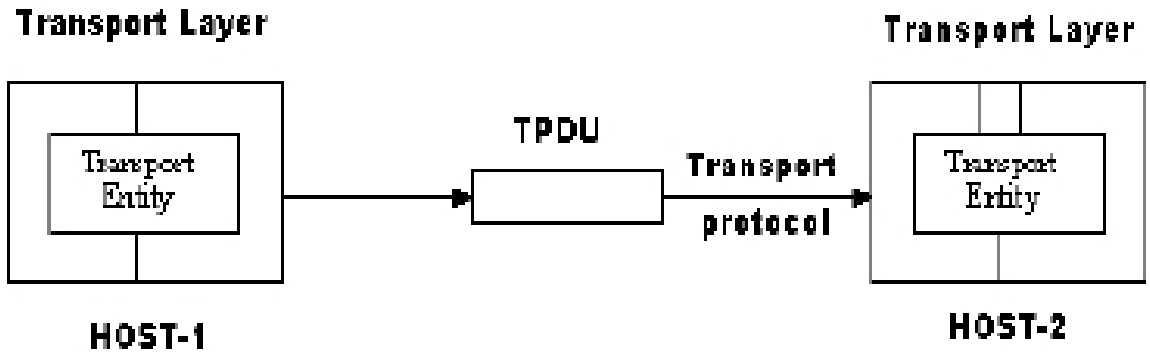
Transport Entity: The hardware and/or software which make use of services provided by the network layer, (within the transport layer) is called transport entity.

Transport Service Provider: Layers 1 to 4 are called Transport Service Provider.

Transport Service User: The upper layers i.e., layers 5 to 7 are called Transport Service User.

Transport Service Primitives: Which allow transport users (application programs) to access the transport service.

TPDU (Transport Protocol Data Unit): Transmissions of message between 2 transport entities are carried out by TPDU. The transport entity carries out the transport service primitives by blocking the caller and sending a packet the service. Encapsulated in the payload of this packet is a transport layer message for the server's transport entity. The task of the transport layer is to provide reliable, cost-effective data transport from the source machine to the destination machine, independent of physical network or networks currently in use.



TRANSPORT SERVICE

1. Services Provided to the Upper Layers

The ultimate goal of the transport layer is to provide efficient, **reliable, and cost-effective data transmission** service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the **services pro-vided by the network layer.** The software and/or hardware within the transport layer that does the work is called the **transport entity.** The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card.

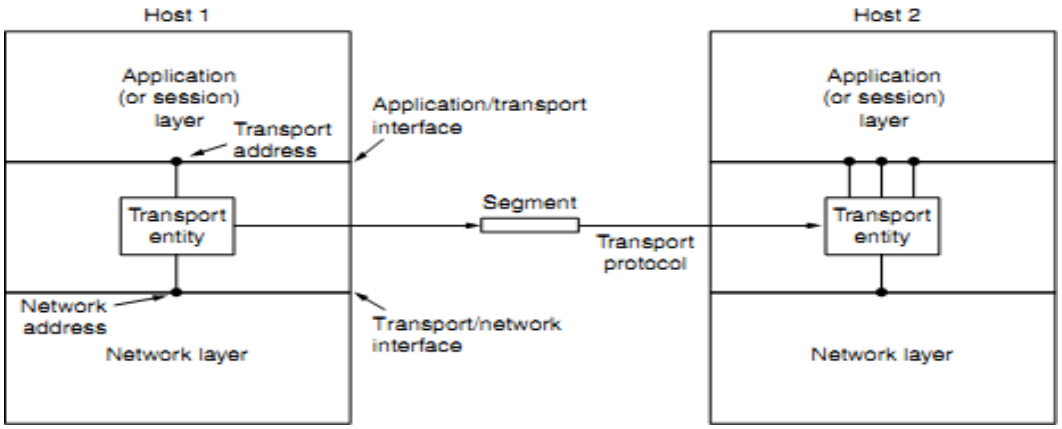


Fig 4.1: The network, Application and transport layer

There are two types of network service

- Connection-oriented
- Connectionless

Similarly, there are also two types of transport service. The connection-oriented transport service is similar to the connection-oriented network service in many ways.

In both cases, connections have three phases:

- Establishment
- Data transfer
- Release.

- Addressing and flow control are also similar in both layers. Furthermore, the connectionless transport service is also very similar to the connectionless network service.
- The bottom four layers can be seen as the transport service provider, whereas the upper layer(s) are the transport service user.

2. Transport Service Primitives

- To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface.

- The transport service is similar to the network service, but there are also some important differences.
- The **main difference** is that the network service is intended to model the service offered by real networks. Real networks can lose packets, so the network service is generally **unreliable.**
- The (connection-oriented) transport service, in contrast, is **reliable**

As an example, consider two processes connected by pipes in UNIX. They assume the connection between them is perfect. They do not want to know about acknowledgements, lost packets, congestion, or anything like that. What they want is a 100 percent reliable connection. Process A puts data into one end of the pipe, and process B takes it out of the other.

A **second difference** between the network service and transport service is **whom the services are intended for**. The network service is used only by the transport entities. Consequently, the transport service must be convenient and easy to use.

Table:4.1 - The primitives for a simple transport service.

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Eg: Consider an application with a server and a number of remote clients.

1. The server executes a "LISTEN" primitive by calling a library procedure that makes a System call to block the server until a client turns up.
2. When a client wants to talk to the server, it executes a "CONNECT" primitive, with "CONNECTION REQUEST" TPDU sent to the server.
3. When it arrives, the TE unblocks the server and sends a "CONNECTION ACCEPTED" TPDU back to the client.
4. When it arrives, the client is unblocked and the connection is established. Data can now be exchanged using "SEND" and "RECEIVE" primitives.
5. When a connection is no longer needed, it must be released to free up table space within the 2 transport entries, which is done with "DISCONNECT" primitive by sending "DISCONNECTION REQUEST"

   TPDU. This disconnection can b done either by asymmetric variant (connection is released, depending on other one) or by symmetric variant (connection is released, independent of other one).

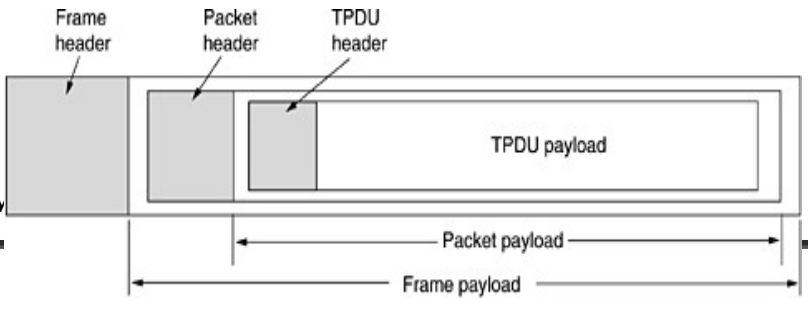


Figure 4.2 - Nesting of TPDUs, packets, and frames

- The term segment for messages sent from transport entity to transport entity.
- TCP, UDP and other Internet protocols use this term. Segments (exchanged by the transport layer) are contained in packets (exchanged by the network layer).
- These packets are contained in frames(exchanged by the data link layer).When a frame arrives, the data link layer processes the frame header and, if the destination address matches for local delivery, passes the contents of the frame payload field up to the network entity.
- The network entity similarly processes the packet header and then passes the contents of the packet payload up to the transport entity. This nesting is illustrated in Fig. 4.2.

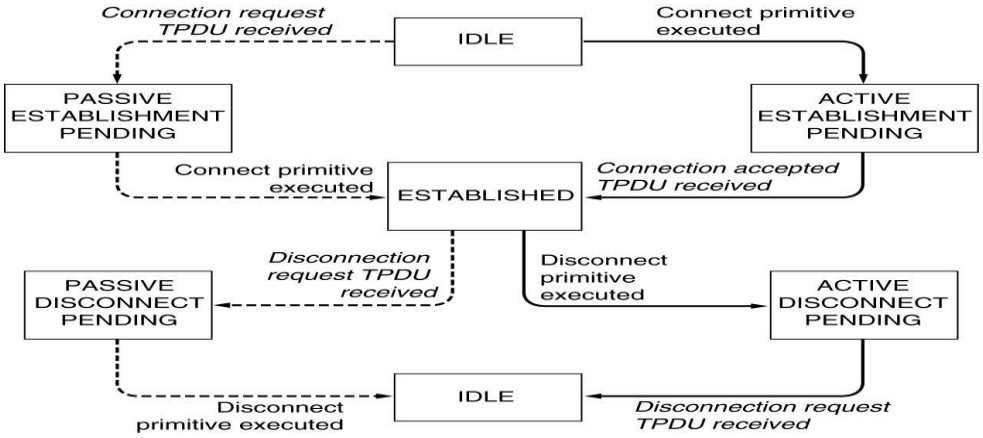


Figure 4.3 - A state diagram for a simple connection management scheme. Transitions labelled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

ELEMENTS OF TRANSPORT PROTOCOLS

The transport service is implemented by a transport protocol used between the two transport entities. The transport protocols resemble the data link protocols. Both have to deal with error control, sequencing, and flow control, among other issues. The difference transport protocol and data link protocol depends upon the environment in which they are operated.

These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig.

At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network. This difference has many important implications for the protocols.

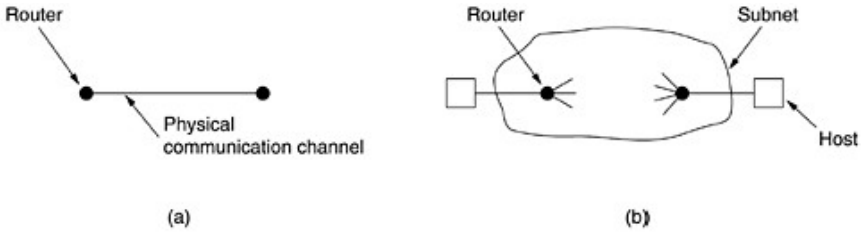


Figure (a) Environment of the data link layer. (b) Environment of the transport layer.

In the data link layer, it is not necessary for a router to specify which router it wants to talk to. In the transport layer, explicit addressing of destinations is required.

In the transport layer, initial connection establishment is more complicated, as we will see. Difference between the data link layer and the transport layer is the potential existence of storage capacity in the subnet

Buffering and flow control are needed in both layers, but the presence of a large and dynamically varying number of connections in the transport layer may require a different approach than we used in the data link layer.

The transport service is implemented by a transport protocol between the 2 transport entities.

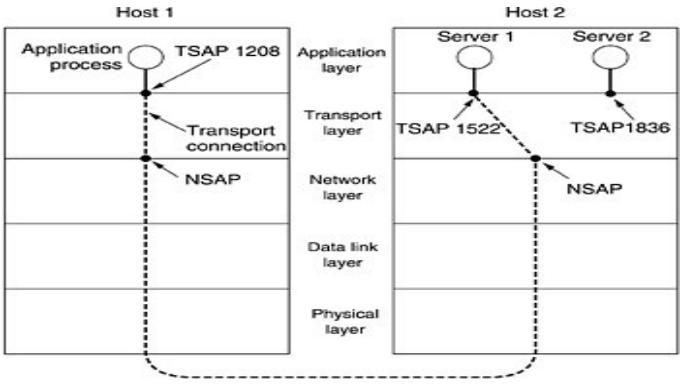


Figure 4.5 illustrates the relationship between the NSAP, TSAP and transport connection. Application processes, both clients and servers, can attach themselves to a TSAP to establish a connection to a remote TSAP.

These connections run through NSAPs on each host, as shown. The purpose of having TSAPs is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport end points that share that NSAP.

The elements of transport protocols are:

1. ADDRESSING
2. Connection Establishment.
3. Connection Release.
4. Error control and flow control
5. Multiplexing.

1. ADDRESSING

When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these endpoints are called **ports**.

There are two types of access points.

TSAP (Transport Service Access Point) to mean a specific endpoint in the transport layer.

The analogous endpoints in the network layer (i.e., network layer addresses) are not surprisingly

called

NSAPs (Network Service Access Points). IP addresses are examples of NSAPs.

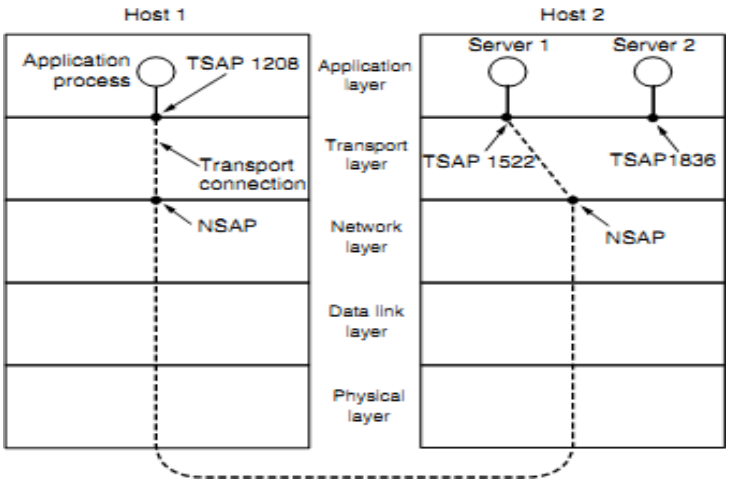


Fig 4.5: TSAP and NSAP network connections

Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP. These connections run through NSAPs on each host. The purpose of having TSAPs is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport endpoints that share that NSAP.

A possible scenario for a transport connection is as follows:

1. A mail server process attaches itself to TSAP 1522 on host 2 to wait for an incoming call. How a process attaches itself to a TSAP is outside the networking model and depends entirely on the local operating system. A call such as our LISTEN might be used, for example.
2. An application process on host 1 wants to send an email message, so it attaches itself to TSAP 1208 and issues a CONNECT request. The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.
3. The application process sends over the mail message.
4. The mail server responds to say that it will deliver the message.
5. The transport connection is released.

2. CONNECTION ESTABLISHMENT:

With packet lifetimes bounded, it is possible to devise a fool proof way to establish connections safely.

Packet lifetime can be bounded to a known maximum using one of the following techniques:

- Restricted subnet design
- Putting a hop counter in each packet

- Time stamping in each packet

Using a 3-way hand shake, a connection can be established. This establishment protocol doesn't require both sides to begin sending with the same sequence number.

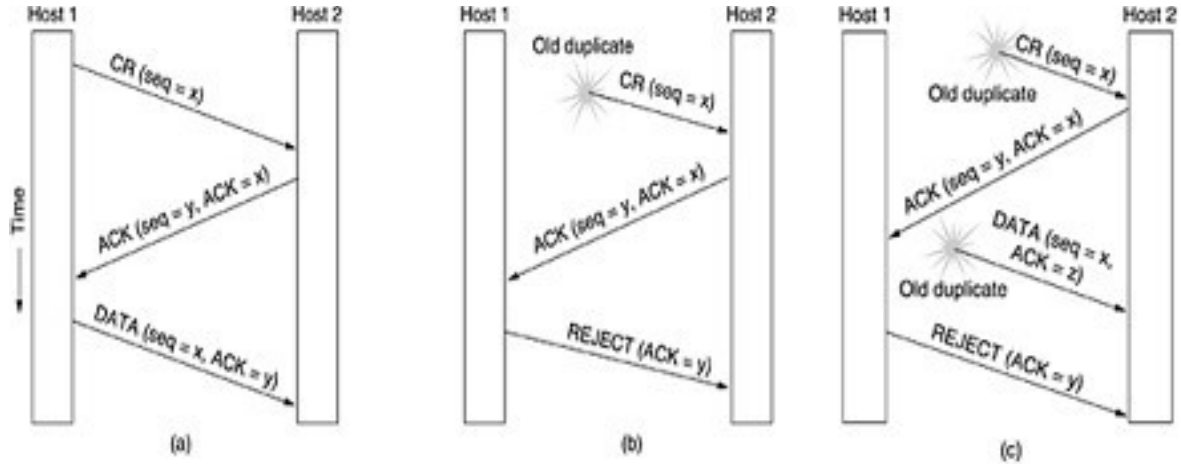


Fig 4.6: Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNEC TION REQUEST (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK .

- The **first technique** includes any method that prevents packets from looping, combined with some way of bounding delay including congestion over the longest possible path. It is difficult, given that internets may range from a single city to international in scope.
- The **second method** consists of having the hop count initialized to some appropriate value and decremented each time the packet is forwarded. The network protocol simply discards any packet whose hop counter becomes zero.
- The **third method** requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreed-upon time.

In **fig (A)** Tomlinson (1975) introduced the **three-way handshake**.

- This establishment protocol involves one peer checking with the other that the connection request is indeed current. Host 1 chooses a sequence number, x , and sends a CONNECTION REQUEST segment containing it to host 2. Host 2replies with an ACK segment acknowledging x and announcing its own initial sequence number, y.

- Finally, host 1 acknowledges host 2's choice of an initial sequence number in the first data segment that it sends

In **fig (B)** the first segment is a delayed duplicate CONNECTION REQUEST from an old connection.

- This segment arrives at host 2 without host 1's knowledge. Host 2 reacts to this segment by sending host1an ACK segment, in effect asking for verification that host 1 was indeed trying to set up a new connection.
- When host 1 rejects host 2's attempt to establish a connection, host 2 realizes that it was tricked by a delayed duplicate and abandons the connection. In this way, a delayed duplicate does no damage.

- The worst case is when both a delayed CONNECTION REQUEST and an ACK are floating around in the subnet.

In **fig (C)** previous example, host 2 gets a delayed CONNECTION REQUEST and replies to it.

- At this point, it is crucial to realize that host 2 has proposed using y as the initial sequence number for host 2 to host 1 traffic, knowing full well that no segments containing sequence number y or acknowledgements to y are still in existence.
- When the second delayed segment arrives at host 2, the fact that z has been acknowledged rather than y tells host 2 that this, too, is an old duplicate.
- The important thing to realize here is that there is no combination of old segments that can cause the protocol to fail and have a connection set up by accident when no one wants it.

3.CONNECTION RELEASE:

A connection is released using either asymmetric or symmetric variant. But, the improved protocol for releasing a connection is a 3-way handshake protocol.

There are two styles of terminating a connection:

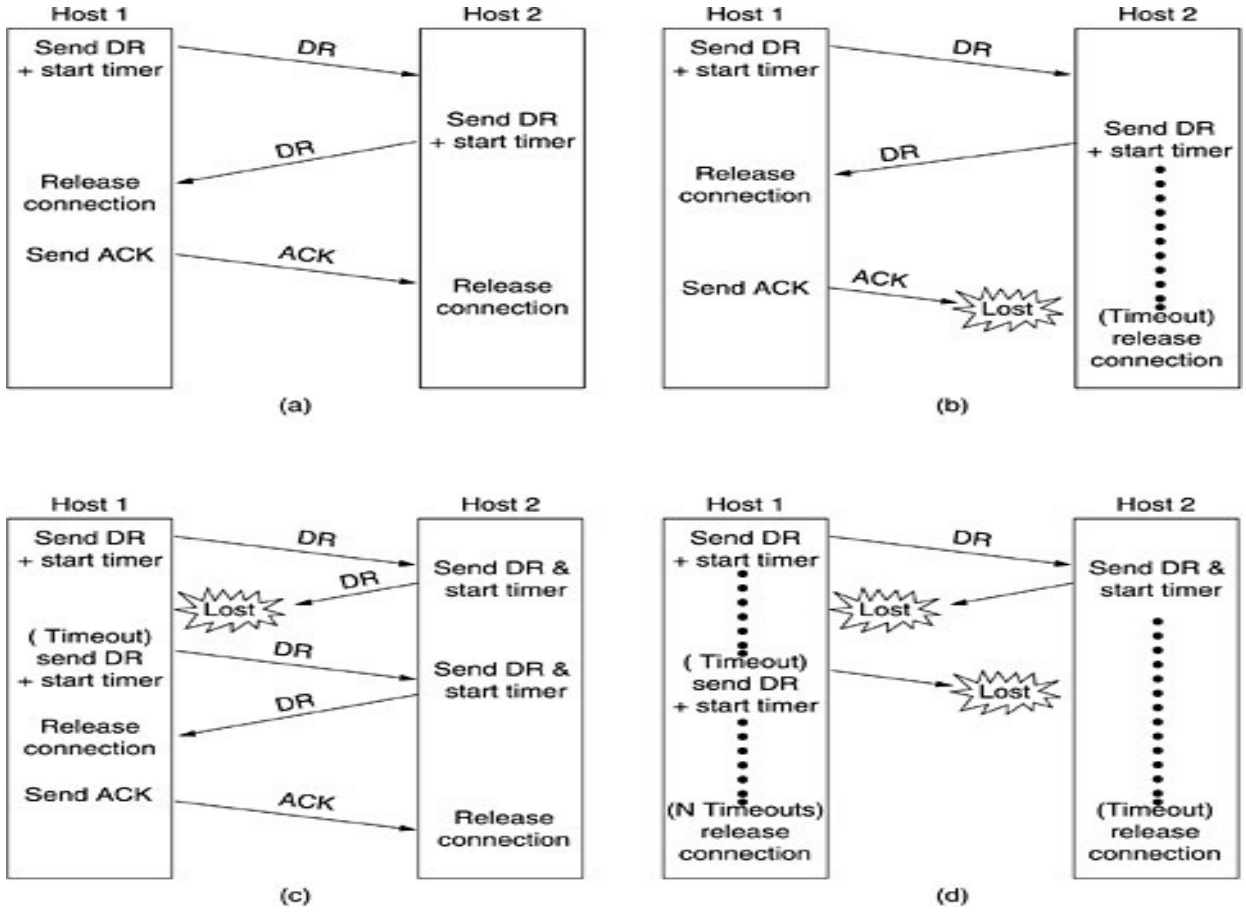
1) Asymmetric release and
2) Symmetric release.

Asymmetric release is the way the telephone system works: when one party hangs up, the connection is broken.

Symmetric release treats the connection as two separate unidirectional connections and requires each one to be released separately.

Fig-(a)	Fig-(b)	Fig-(c)	Fig-(d)

One of the user sends a DISCONNECTION REQUEST TPDU in order to initiate connection release. When it arrives, the recipient sends back a DR-TPDU, too, and starts a timer. When this DR arrives, the original sender sends back an ACK-TPDU and releases the connection. Finally, when the ACK-TPDU arrives, the receiver also releases the connection.	Initial process is done in the same way as in fig-(a). If the final ACK-TPDU is lost, the situation is saved by the timer. When the timer is expired, the connection is released.	If the second DR is lost, the user initiating the disconnection will not receive the expected response, and will timeout and starts all over again.	Same as in fig-(c) except that all repeated attempts to retransmit the DR is assumed to be failed due to lost TPDUs. After 'N' entries, the sender just gives up and releases the connection.



TRANSPORT PROTOCOLS - UDP

The Internet has two main protocols in the transport layer, **a connectionless protocol** and a **connection- oriented** one. The protocols complement each other. The connectionless protocol is **UDP.** It does almost nothing beyond sending packets between applications, letting applications build their own protocols on top as needed.

The connection-oriented protocol is **TCP.** It does almost everything. It makes connections and adds reliability with retransmissions, along with flow control and congestion control, all on behalf of the applications that use it. Since UDP is a transport layer protocol that typically runs in the operating system and protocols that use UDP typically run in user s pace, these uses might be considered applications.

INTROUCTION TO UDP

- The Internet protocol suite supports a connectionless transport protocol called UDP (User Datagram Protocol). UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection.
- UDP transmits segments consisting of an 8-byte header followed by the pay-load. The two ports serve to identify the end-points within the source and destination machines.
- When a UDP packet arrives, its payload is handed to the process attached to the destination port. This attachment occurs when the BIND primitive. Without the port fields, the transport layer would not know what to do with each incoming packet. With them, it delivers the embedded segment to the correct application.

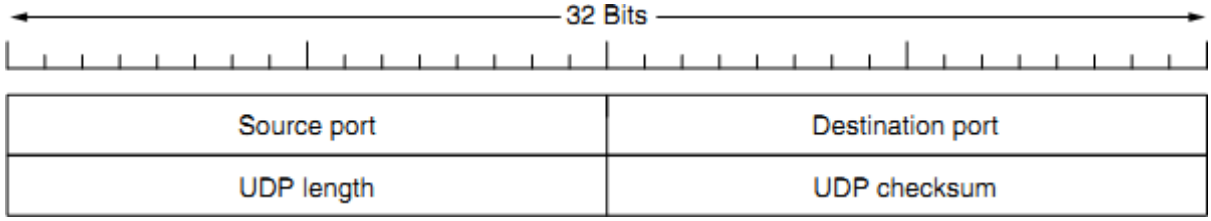


Fig 4.9: The UDP header

Source port, destination port: Identifies the end points within the source and destination machines.

UDP length: Includes 8-byte header and the data

UDP checksum: Includes the UDP header, the UDP data padded out to an even number of bytes if need be. It is an optional field

Disadvantages of UDP protocol

- UDP provides basic functions needed for the end-to-end delivery of a transmission.
- It does not provide any sequencing or reordering functions and does not specify the damaged packet when reporting an error.
- UDP can discover that an error has occurred, but it does not specify which packet has been lost as it does not contain an ID or sequencing number of a particular data segment.

TCP (TRANSMISSION CONTROL PROTOCOL)

It was specifically designed to provide a reliable end-to end byte stream over an unreliable network. It was designed to adapt dynamically to properties of the inter network and to be robust in the face of many kinds of failures.

Each machine supporting TCP has a TCP transport entity, which accepts user data streams from local processes, breaks them up into pieces not exceeding 64kbytes and sends each piece as a separate IP datagram. When these datagrams arrive at a machine, they are given to TCP entity, which reconstructs the original byte streams. It is up to TCP to time out and retransmits them as needed, also to reassemble datagrams into messages in proper sequence.

Features Of TCP protocol

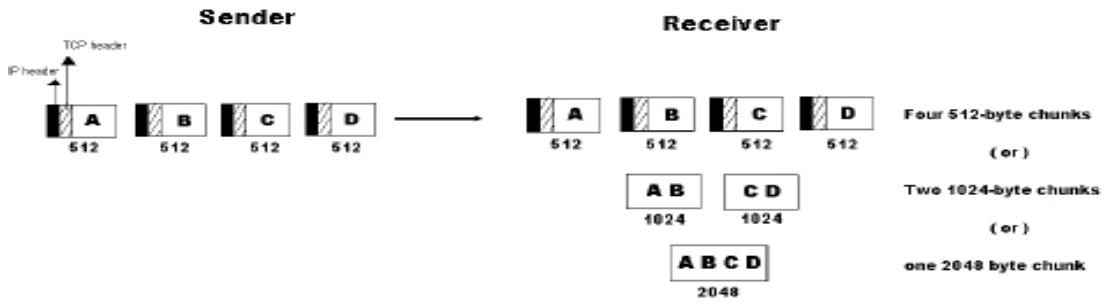
- **Stream data transfer:** TCP protocol transfers the data in the form of contiguous stream of bytes. TCP group the bytes in the form of TCP segments and then passed it to the IP layer for transmission to the destination. TCP itself segments the data and forward to the IP.
- **Reliability:** TCP assigns a sequence number to each byte transmitted and expects a positive acknowledgement from the receiving TCP.
- If ACK is not received within a timeout interval, then the data is retransmitted to the destination. The receiving TCP uses the sequence number to reassemble the segments if they arrive out of order or to eliminate the duplicate segments.
- **Flow Control:** When receiving TCP sends an acknowledgement back to the sender indicating the number the bytes it can receive without overflowing its internal buffer. The number of bytes is sent in ACK in the form of the highest sequence number that it can receive without any problem. This mechanism is also referred to as a window mechanism.
- **Multiplexing:** Multiplexing is a process of accepting the data from different applications and forwarding to the different applications on different computers. At the receiving end, the data is forwarded to the correct application. This process is known as demultiplexing. TCP transmits the packet to the correct application by using the logical channels known as ports.
- **Logical Connections:** The combination of sockets, sequence numbers, and window sizes, is called a logical connection. Each connection is identified by the pair of sockets used by sending and receiving processes.
- **Full Duplex:** TCP provides Full Duplex service, i.e., the data flow in both the directions at the same time. To achieve Full Duplex service, each TCP should have sending and receiving buffers so that the segments can flow in both the directions. TCP is a connection-oriented protocol. Suppose the process A wants to send and receive the data from process B. The following steps occur:
    - Establish a connection between two TCPs.
    - Data is exchanged in both the directions.
    - The Connection is terminated.

The TCP Service Model

- TCP service is obtained by having both the sender and receiver create end points called **SOCKETS**
- Each socket has a socket number(address)consisting of the IP address of the host, called a **"PORT"** (= TSAP)

- To obtain TCP service a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine
- All TCP connections are full duplex and point to point i.e., multicasting or broadcasting is not supported.
- A TCP connection is a byte stream, not a message stream i.e., the data is delivered as chunks

E.g.: 4 * 512 bytes of data is to be transmitted.



Sockets:

A socket may be used for multiple connections at the same time. In other words, 2 or more connections may terminate at same socket. Connections are identified by socket identifiers at same socket. Connections are identified by socket identifiers at both ends. Some of the sockets are listed below:

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

Ports: Port numbers below 256 are called Well- known ports and are reserved for standard services.

***Eg*:**

PORT-21	To establish a connection to a host to transfer a file using FTP
PORT-23	To establish a remote login session using TELNET

The TCP Protocol

- A key feature of TCP, and one which dominates the protocol design, is that every byte on a TCP connection has its own 32-bit sequence number.
- When the Internet began, the lines between routers were mostly 56-kbps leased lines, so a host

blasting away at full speed took over 1 week to cycle through the sequence numbers.

- The basic protocol used by TCP entities is the **sliding window protocol**.
- When a sender transmits a segment, it also starts a timer.
- When the segment arrives at the destination, the receiving TCP entity sends back a segment (with data if any exist, otherwise without data) bearing an acknowledgement number equal to the next sequence number it expects to receive.
- If the sender's timer goes off before the acknowledgement is received, the sender transmits the segment again.

The TCP Segment Header

Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options. After the options, if any, up to 65,535 - 20 - 20 = 65,495 data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header. Segments without any data are legal and are commonly used for acknowledgements and control messages.

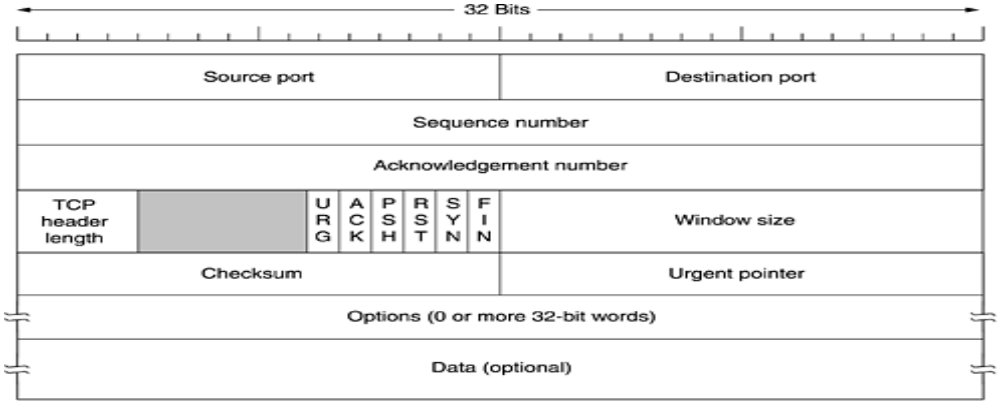


Fig 4.11: The TCP Header

Source Port, Destination Port : Identify local end points of the connections

Sequence number: Specifies the sequence number of the segment

Acknowledgement Number: Specifies the next byte expected.

TCP header length: Tells how many 32-bit words are contained in TCP header

URG: It is set to 1 if URGENT pointer is in use, which indicates start of urgent data.

ACK: It is set to 1 to indicate that the acknowledgement number is valid.

PSH: Indicates pushed data

RST: It is used to reset a connection that has become confused due to reject an invalid segment or refuse an attempt to open a connection.

FIN: Used to release a connection.

SYN: Used to establish connections.

TCP Connection Establishment

To establish a connection, one side, say, the server, passively waits for an incoming connection by executing the LISTEN and ACCEPT primitives, either specifying a specific source or nobody in particular.

The other side, say, the client, executes a CONNECT primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data (e.g., a password).

The CONNECT primitive sends a TCP segment with the *SYN* bit on and *ACK* bit off and waits for a response.

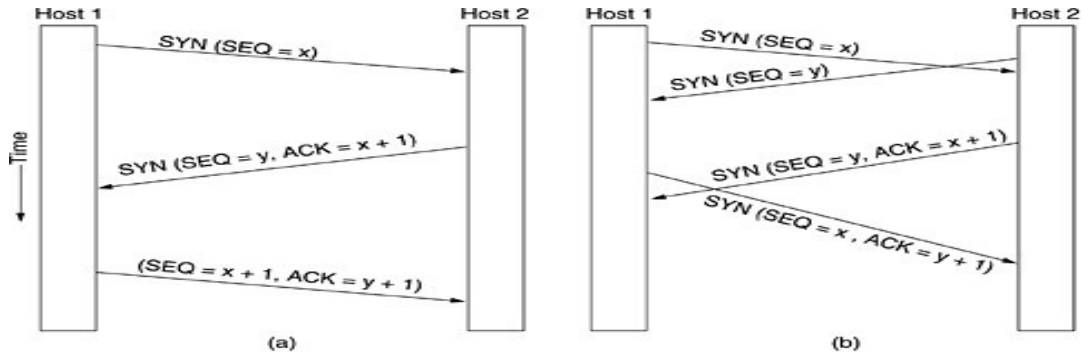


Fig 4.12: a) TCP Connection establishment in the normal case b) Call Collision

Differences b/w TCP & UDP

Basis for Comparison	TCP	UDP
Definition	TCP establishes a virtual circuit before transmitting the data.	UDP transmits the data directly to the destination computer without verifying whether the receiver is ready to receive or not.
Connection Type	It is a Connection-Oriented protocol	It is a Connectionless protocol
Speed	Slow	high
Reliability	It is a reliable protocol.	It is an unreliable protocol.
Header size	20 bytes	8 bytes
Acknowledgement	It waits for the acknowledgement of data and has the ability to resend the lost packets.	It neither takes the acknowledgement, nor it retransmits the damaged frame.

Network Security:

Computer security requirements and Attacks:

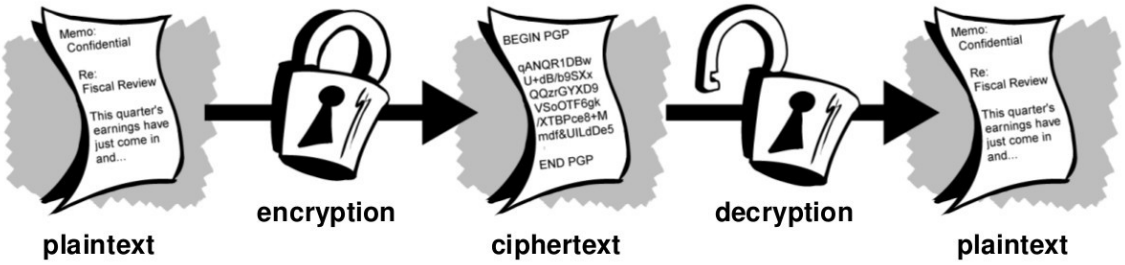
Computer and network security address four requirements:
1. Confidentiality: Requires that data only be accessible by authorized parties. This types of access includes printing, displaying and other forms of disclosure of the data.
2. Integrity: Requires that data can be modified only by authorized users. Modification includes writing, changing, changing status, deleting and creating.
3. Availability: Requires that data are available to authorized parties.
4. Authenticity: Requires that host or service be able to verify the identity of a user.

Types of Network Attacks
There are four primary classes of attacks.
1. Reconnaissance : Reconnaissance is the unauthorized discovery and mapping of systems, services, or vulnerabilities. It is also known as information gathering and, in most cases, it precedes another type of attack. Reconnaissance is similar to a thief casing a neighborhood for vulnerable homes to break into, such as an unoccupied residence, easy-to-open doors, or open windows.
2. Access : System access is the ability for an intruder to gain access to a device for which the intruder does not have an account or a password. Entering or accessing systems usually involves running a hack, script, or tool that exploits a known vulnerability of the system or application being attacked.
3. Denial of Service : Denial of service (DoS) is when an attacker disables or corrupts networks, systems, or services with the intent to deny services to intended users. DoS attacks involve either crashing the system or slowing it down to the point that it is unusable. But DoS can also be as simple as deleting or corrupting information. In most cases, performing the attack involves simply running a hack or script. For these reasons, DoS attacks are the most feared.
4. Worms, Viruses, and Trojan Horses : Malicious software can be inserted onto a hos to damage or corrupt a system, replicate itself, or deny access to networks, systems, or services. Common names for this type of software are worms, viruses, and Trojan horses.

Cryptography

- Cryptography is derived from the Greek words: kryptos, "hidden", and graphein, "to write" - or "hidden writing".
- Cryptography is the science of using mathematics to encrypt and decrypt data.
- Cryptography enables you to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient. While cryptography is the science of securing data, cryptanalysis is the science of analyzing and breaking secure communication.
- Classical cryptanalysis involves an interesting combination of analytical reasoning, application of mathematical tools, pattern finding, patience, determination, and luck. Cryptanalysts are also called attackers.
- Cryptology embraces both cryptography and cryptanalysis.

Encryption and Decryption



Plain-text and Cipher-text

- The original message, before being transformed, is called plaintext. After the message is transformed, it is called cipher-text.
- An encryption algorithm transforms the plain text into ciphertext;
- a decryption algorithm transforms the cipher-text back into plain- text.
- The sender uses an encryption algorithm, and the receiver uses a decryption algorithm.

Cipher

- Encryption and decryption algorithms as ciphers. The term cipher is also used to refer to different categories of algorithms in cryptography. This is not to say that every sender-receiver pair needs their very own unique cipher for a secure communication. On the contrary, one cipher can serve millions of communicating pairs.

Key

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

Alice, Bob, and Eve

- In cryptography, it is customary to use three characters in an information exchange scenario; we use Alice, Bob, and Eve.
- Alice is the person who needs to send secure data.
- Bob is the recipient of the data.
- Eve is the person who somehow disturbs the communication between Alice and Bob by intercepting messages to uncover the data or by sending her own disguised messages.
- These three names represent computers or processes that actually send or receive data, or intercept or change data.

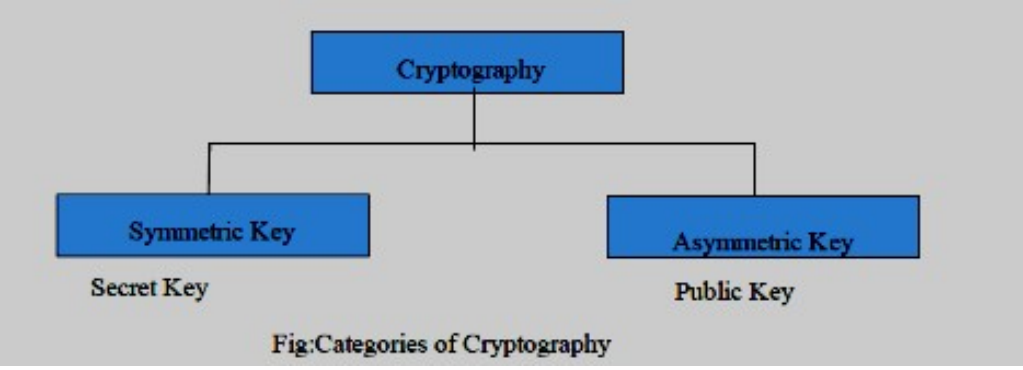
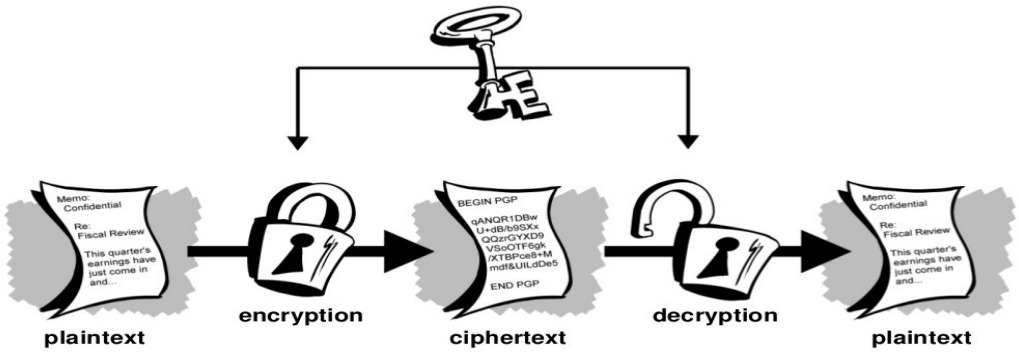


Fig:Categories of Cryptography

Symmetric-key

- In conventional cryptography, also called secret-key or symmetric-key encryption, one key is used both for encryption and decryption.
- The Data Encryption Standard (DES) is an example of a conventional cryptosystem that is widely employed by the Federal Government.
- Figure below shows an illustration of the conventional encryption process.

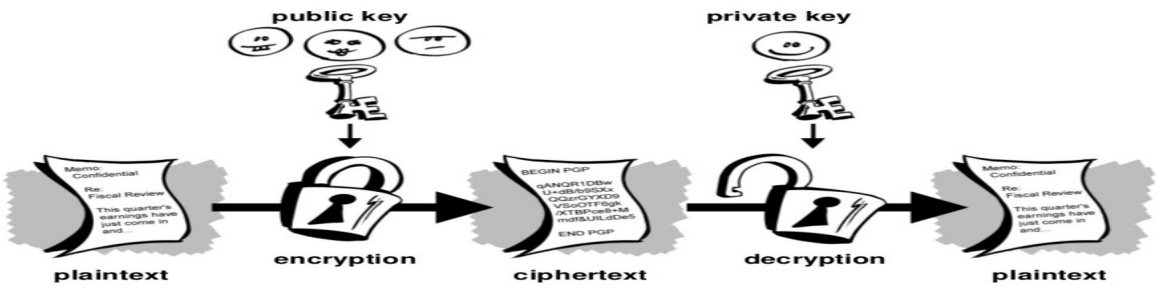


- Conventional encryption has benefits. It is very fast. It is especially useful for encrypting data that is not going anywhere.
- conventional encryption alone as a means for transmitting secure data can be quite expensive simply due to the difficulty of secure key distribution.
- For a sender and recipient to communicate securely using conventional encryption, they must agree upon a key and keep it secret between themselves.
- If they are in different physical locations, they must trust a courier, the Bat Phone, or some other secure communication medium to prevent the disclosure of the secret key during transmission.
- Anyone who overhears or intercepts the key in transit can later read, modify, and forge all information encrypted or authenticated with that key.

Asymmetric-Key Cryptography

- Public key cryptography is an asymmetric scheme that uses a pair of keys for encryption: a public key, which encrypts data, and a corresponding private, or secret key for decryption.
- You publish your public key to the world while keeping your private key secret. Anyone with a copy of your public key can then encrypt information that only you can read .

- It is computationally infeasible to deduce the private key from the public key.
- Anyone who has a public key can encrypt information but cannot decrypt it.
- Only the person who has the corresponding private key can decrypt the information.



The Essential steps in Asymmetric-key cryptography are the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the keys in a public register or other accessible file. This is the public key. The companion key is kept private. Each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows the Alice's private key.

Comparison:

Let us compare symmetric-key and asymmetric-key cryptography.

- Encryption can be thought of as electronic locking;
- Decryption as electronic unlocking.
- The sender puts the message in a box and locks the box by using a key;
- The receiver unlocks the box with a key and takes out the message. The difference lies in the mechanism of the locking and unlocking and the type of keys used.
- In symmetric-key cryptography, the same key locks and unlocks the box.
- In asymmetric-key cryptography, one key locks the box, but another key is needed to unlock it.

Traditional Cipher used in Symmetric-key Cryptography:

Two types:

1. Substitution cipher
2. Transposition cipher

Substitution cipher:

A substitution cipher substitutes one symbol with another. If the symbols in the plain- text are alphabetic characters, we replace one character with another. For example, we can replace character A with D, and character T with Z. If the symbols are digits (0 to 9), we can replace 3 with 7, and 2 with 6. It is also known and Ceaser's Cipher who invented it.

For example,
If we encode the word "SECRET" using Caesar's key value of 3, we offset the alphabet so that the 3rd letter down (D) begins the alphabet.
So starting with

ABCDEFGHIJKLMNOPQRSTUVWXYZ and sliding everything up by 3, you get
DEFGHIJKLMNOPQRSTUVWXYZABC where D=A, E=B, F=C, and so on.
Using this scheme, the **plaintext, "SECRET" encrypts as "VHFUHW."**
To allow someone else to read the ciphertext, you tell them that the **key is 3.**

Transposition Ciphers

- In a transposition cipher, there is no substitution of characters; instead, their locations change.
- A character in the first position of the plaintext may appear in the tenth position of the ciphertext.
- A character in the eighth position may appear in the first position.
- In other words, a transposition cipher reorders the symbols in a block of symbols.
- Key In a transposition cipher, the key is a mapping between the position of the symbols in the plaintext and cipher text.

  For example, the following shows the key using a block of four characters:

Plaintext: 2 4 1 3
Ciphertext: 1 2 3 4

In encryption, we move the character at position 2 to position 1, the character at position 4 to position 2, and so on. In decryption, we do the reverse.

Encryption algorithm:

The most commonly used symmetric encryption are block ciphers. A block cipher processes the plain text input in fixed size blocks and produces a block of cipher text of equal size for each plain text block.

The two most important symmetric algorithms, both of which are block ciphers, are

- **Data Encryption Standard (DES)**
- **Advanced Encryption Standard (AES)**

Asymmetric Key Cryptography:

Some examples of public-key cryptosystems are :

- **Elgamal (named for its inventor, Taher Elgamal),**
- **RSA (named for its inventors, Ron Rivest, Adi Shamir, and Leonard Adleman),**
- **Diffie-Hellman (named for its inventors),**
- **DSA ,the Digital Signature Algorithm (invented by David Kravitz).**