TRANSPORT LAYER(CO1)

Topic Objective

- The student will be able to understand the basic concept and terms of transport layer
- Understand the working of various transport layer protocols

TRANSPORT LAYER(CO1)

- The transport layer is the core of the OSI model.
- Protocols at this layer oversee the delivery of data from an application program on one device to an application program on another device.
- They act as a liaison between the upper-layer protocols (session, presentation, and application) and the services provided by the lower layers.

TRANSPORT LAYER

Duties of the transport layer:

- The services provided are similar to those of the data link layer.
- provides services across an internetwork made of many networks.
- While the transport layer controls all three of the lower layers.

TRANSPORT LAYER

Quality of Service

The transport protocol improves the QoS (Quality of Service) provided by the network layer.

Following are the QoS parameters:

- **Connection establishment delay:**

  the amount of time elapsing between a transport connection being requested and the confirmation being received by the user of the transport service.

- **Connection establishment failure probability:**

  the chance of a connection not being established within the maximum establishment delay time.

TRANSPORT LAYER(CO4)

- **Throughput:**

  measures the number of bytes of user data transferred per second, measured over some time interval.

- **Transit delay:**

  the time between a message being sent by the transport user on the source machine and its being received by the transport user on the destination machine.

- **The Residual error ratio:**

  the number of lost or garbled messages as a fraction of the total sent.

TRANSPORT LAYER(CO5)

- **The Protection**

  a way for the transport user to specify interest in having the transport layer provide protection against unauthorized third parties (wiretappers) reading or modifying the transmitted data.

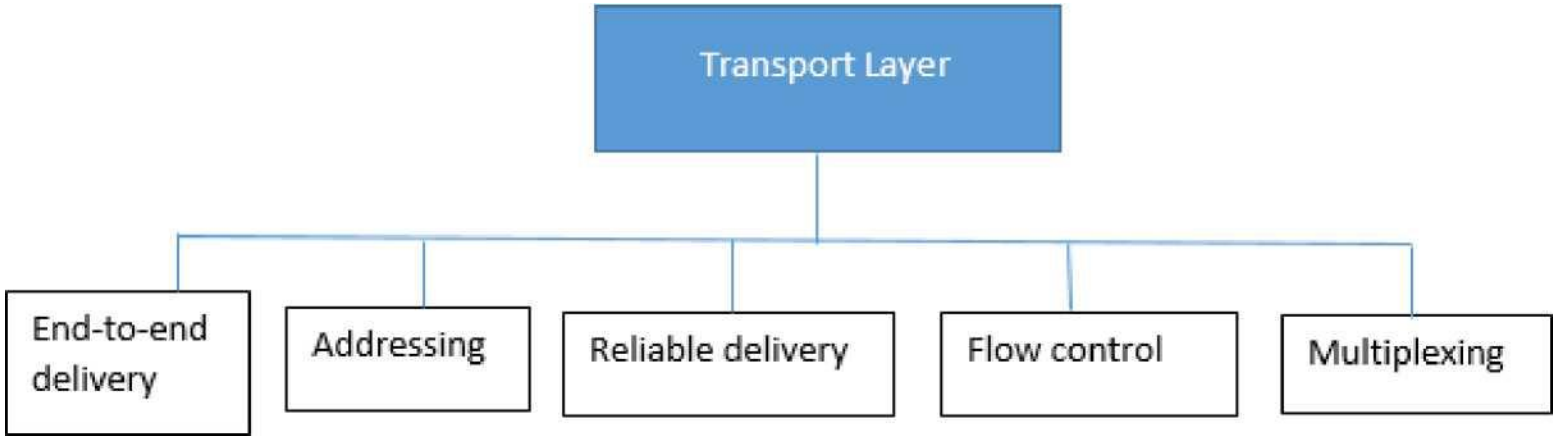
- **The Priority**

  a way for a transport user to indicate that some of its connections are more important than other ones, and in the event of congestion, to make sure that the high-priority connections get serviced before the low-priority ones.

TRANSPORT LAYER(CO1)

The services provided by transport layer protocols can be divided into five broad categories:

- end-to-end deliver,
- addressing,
- reliable delivery,
- flow control, and
- multiplexing.



TRANSPORT LAYER

End -to -end delivery

The network layer oversees the end-to-end delivery of individual packets but does not see any relationship between those packets, even those belonging to a single message.

It treats each as an independent entity. The transport layer, on the other hand, makes sure that the entire message (not just a single packet) arrives intact. Thus, it oversees the end-to-end (source -to-destination) delivery of an entire message.

TRANSPORT LAYER

Addressing

The transport layer interacts with the functions of the session layer. Communication occurs not just from end machine to end machine but from end application to end application. Data generated by an application on one machine must be received not just by the other machine but by the correct application on that other machine. To ensure accurate delivery from service access point to service access point, we need another level of addressing in addition, the protocol needs to know which upper-layer protocols are communicating.

TRANSPORT LAYER

Reliable Delivery

At the transport layer, reliable delivery has four aspects: error control, sequence control, loss control, and duplication control.

- **Error Control -** When transferring data, the primary goal of reliability is error control. Node-to-node reliability of DLL does not ensure end-to-end reliability.

- **Sequence Control -** for ensuring that the various pieces of a transmission are correctly reassembled.

  - **Segmentation and Concatenation -** the transport protocol divides data into smaller, usable blocks. The dividing process is called segmentation. When smaller data then combines them into a single data unit. The combing process is called concatenation.
  - **Sequence Numbers -** each segment carries a field that indicates whether it is the final segment of a transmission or a middle segment with more still to come.

TRANSPORT LAYER

- **Loss Control -** The transport layer ensures that all pieces of a transmission arrive at the destination, not just some of them.
- **Duplication Control -** guarantee that no pieces of data arrive at the receiving system duplicated.

TRANSPORT LAYER

Flow Control

Like the data link layer, the transport layer is responsible for flow control. However, flow control at this layer is performed end-to-end rather than across a single link. Transport layer flow control also uses a sliding window protocol. However, the window at the transport layer can vary in size to accommodate buffer occupancy.

TRANSPORT LAYER

Multiplexing

occurs two ways:

upward, meaning that multiple transport layer connections use the same network connection, or

downward, meaning that one transport-layer connection uses multiple network connections.

The transport layer uses virtual circuits based on the services of the lower three layers. Normally, the underlying networks charge for each virtual circuit connection.

To make more cost effective use of an established circuit, the transport layer can send several transmissions bound for the same destination along the same path by upward multiplexing.

TRANSPORT LAYER

ARPANET

ARPA established a packet-switching network of computers linked by point-to-point leased lines called Advanced Research Project Agency Network (ARPANET).

The conventions developed by ARPA to specify how individual computers could communicate across that network became TCP/IP.

The transport layer is represented in TCP/IP by two protocols:

TCP and **UDP**.

TRANSPORT LAYER(CO3)

USER DATAGRM PROTOCOL (UDP)

simpler of the two standard TCP/IP transport protocols. It is an end-to-end transport level protocol that adds only port addresses, check sum error control, and length information to the data from the upper layer. The packet produced by the UDP is called a user datagram

- **Source port address**

The source port address is the address of the application program that has created the message.

- **Destination port address**

The destination port address is the address of the application program that will receive the message.

TRANSPORT LAYER

- **Total length**

  The total length field defines the total length of the user datagram in bytes.

- **Check sum**

  The check sum is a 16-bit field used in error detection.

UDP provides only the basic functions needed for end-to-end delivery of a transmission.
It does not provide any sequencing or recording functions and cannot specify the damaged packet when reporting an error (for which it must be paired with ICMP). UDP contains only a checksum; it does not contain an ID or sequencing number for a particular data segment

TRANSPORT LAYER(CO3)

Transmission Control Protocol (TCP)

provides full transport layer services to applications.

TCP is a reliable stream transport port-to-port protocol, means connection-oriented:

By creating this connection, TCP generates a virtual circuit between sender and receiver that is active for the duration of a transmission

TCP begins each transmission by altering the receiver that datagrams are on their way (connection establishment) and ends each transmission with a connection termination. In this way, the as a connection-oriented service, is responsible for the reliable delivery of the entire stream of bits contained in the message originally generated by the sending application.

TRANSPORT LAYER

Reliability is ensured by provision for error detection and retransmission of damaged frames;

all segments must be received and acknowledged before the transmission is considered complete and the virtual circuit is discarded.

TRANSPORT LAYER

- **The TCP Segment**

  The scope of the services provided by TCP requires that the segment header be extensive.
  A brief description of each field is in order.

- **Source port address**

  The source port address defines the application program in the source computer.

- **Destination port address**

  The destination port address defines the application program in the destination computer.

TRANSPORT LAYER

- **Sequence number**

  A stream of data from the application program may be divided into two or more TCP segments. The sequence number field shows the position of the data in the original data stream.

- **Acknowledgement number**

  The 32-bit acknowledgement number is used to acknowledge the receipt of data from the other communicating device. This number is valid only if the ACK bit in the control field is set.

TRANSPORT LAYER

- **Header Length (HLEN)**

  The four-bit HLEN field indicates the number of 32-bit (four-byte) words in the TCP header. The four bits can define a number up to 15.This is multiplied by 4 to give the total number of bytes in the header. Therefore, the size of the header can be a maximum of 60 bytes (4x15).Since the minimum required size of the header is 20 bytes, 40 bytes are thus available for the options section.

- **Reserved**

  A six-bit field is reserved for future use.

TRANSPORT LAYER

- **Control**

Each bit of the six-bit control field functions individually and independently. A bit can either define the use of a segment or serve as a validity check for other fields. The urgent bit, when set, validates the urgent pointer field. Both this bit and the pointer indicate that the data in the segment are urgent.

The **ACK bit**, when set, validates the acknowledgement number field. Both are used together and have different functions, depending on the segment type.

The **PSH bit** is used to inform the sender that a higher throughput is needed. If possible, data must be pushed through paths with higher throughput. The reset bit is used to reset the connection when there is confusion in the sequence numbers.

TRANSPORT LAYER

The **SYN bit** is used for sequence number synchronization in three types of segments: connection request, connection confirmation (with the ACK bit set), and confirmation acknowledgement (with the ACK bit set).

The **FIN bit** is used in connection termination in three types of segments: termination request, termination confirmation (with the ACK bit set), and acknowledgement of termination confirmation (with the ACK bit set).

- **Window size**

The window is a 16-bit field that defines the size of the sliding window.

TRANSPORT LAYER

- **Checksum**

  The checksum is a 16-bit field used in error detection.

- **Urgent pointer**

  This is the last required field in the header. Its value is valid only if the URG bit in the control field is set. In this case, the sender is informing the receiver that there are urgent data in the data portion of the segment. This pointer defines the end of urgent data and the start of normal data.

- **Options and padding**

  The remainder of the TCP header defines the optional fields. They are used to convey additional information to the receiver or for alignment purposes.

TRANSPORT LAYER

TCP protocol operations may be divided into three phases.

Connections must be properly established in a multi-step handshake process *(**connection establishment**)* before entering the ***data transfer*** phase. After data transmission is completed, the ***connection termination*** closes established virtual circuits and releases all allocated resources.

A TCP connection is managed by an operating system through a programming interface that represents the local end-point for communications, the ***Internet socket.***

- **TCP connection states**

During the lifetime of a TCP connection the local end-point undergoes a series of state changes:

TRANSPORT LAYER

o **LISTEN**

(server) represents waiting for a connection request from any remote TCP and port.

o SYN-SENT

(client) represents waiting for a matching connection request after having sent a connection request.

o SYN-RECEIVED

(server) represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

o ESTABLISHED

(both server and client) represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

TRANSPORT LAYER

o **FIN-WAIT-1**

(both server and client) represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

o **FIN-WAIT-2**

(both server and client) represents waiting for a connection termination request from the remote TCP.

o CLOSE-WAIT

(both server and client) represents waiting for a connection termination request from the local user.

o CLOSING

(both server and client) represents waiting for a connection termination request acknowledgment from the remote TCP.

TRANSPORT LAYER

o **LAST-ACK**

(both server and client) represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

o TIME-WAIT

(either server or client) represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.

o CLOSED

(both server and client) represents no connection state at all.

TRANSPORT LAYER

- **Connection establishment**

To establish a connection, TCP uses a three-way handshake. Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. To establish a connection, the threeway (or 3-step) handshake occurs:

1. **SYN:** The active open is performed by the client sending a SYN to the server. The client sets the segment's sequence number to a random value A.
2. **SYN-ACK:** In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, B.

TRANSPORT LAYER

3.ACK: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A+1, and the acknowledgement number is set to one more than the received sequence number i.e. B+1.

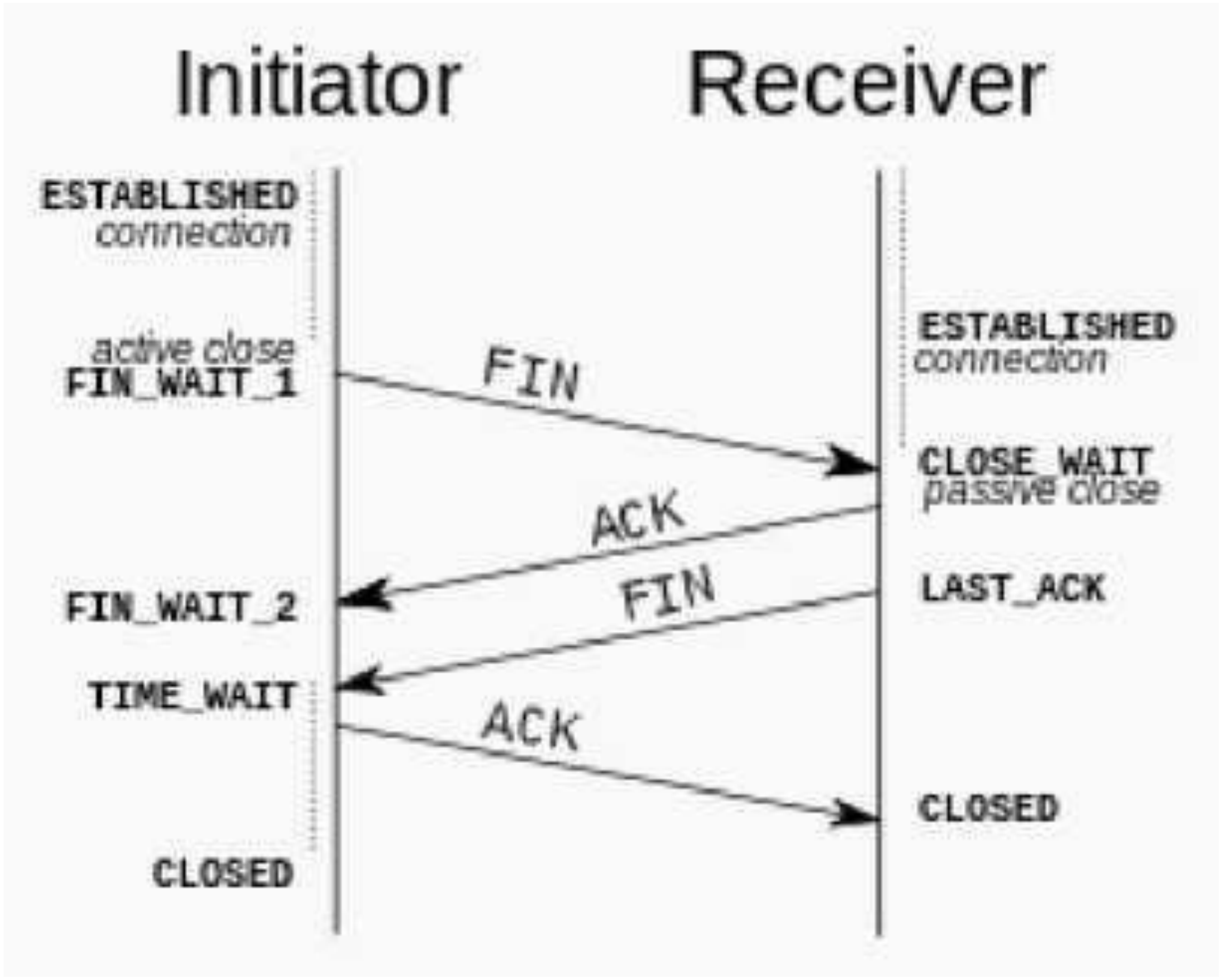
At this point, both the client and server have received an acknowledgment of the connection. The steps 1, 2 establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter (sequence number) for the other direction and it is acknowledged. With these, a full-duplex communication is established.

TRANSPORT LAYER

- **Connection termination**

The connection termination phase uses a four-way handshake, with each side of the connection terminating independently. When an endpoint wishes to stop its half of the connection, it transmits a FIN packet, which the other end acknowledges with an ACK. Therefore, a typical tear-down requires a pair of FIN and ACK segments from each TCP endpoint. After the side that sent the first FIN has responded with the final ACK, it waits for a timeout before finally closing the connection, during which time the local port is unavailable for new connections; this prevents confusion due to delayed packets being delivered during subsequent connections.

TRANSPORT LAYER



Session LAYER(CO1)

Topic Objective

- The student will be able to understand the design issue and Functions of Session layer
- Understand the working of RPC

Recap of Previous topic

- Transport layer being the fourth layer of OSI model
- Supports reliability in the transmission
- TCP and UDP being important protocols of the Transport Layer

Session LAYER(CO1)

- The session layer is level five of the seven layer of OSI model.
- It responds to service requests from the presentation layer and issues service requests to the transport layer.
- The Session Layer allows users on different machines to establish active communication sessions between them.
- It's main aim is to establish, maintain and synchronize the interaction between communicating systems.
- Session layer manages and synchronize the conversation between two different applications.
- streams of data are marked and are resynchronized properly, so that the ends of the messages are not cut prematurely and data loss is avoided. The session layer provides the mechanism for opening, closing and managing a session between end-user application processes,

Session LAYER

- Session layers are commonly used in application environments that make use of remote procedure calls (RPCs).
- An example of a session layer protocol X.225 or ISO 8327.
- In case of a connection loss this protocol may try to recover the connection. If a connection is not used for a long period, the session layer protocol may close it and re-open it.
- It provides for either full duplex or half-duplex operation and provides synchronization points in the stream of exchanged messages.

Session LAYER

Functions of Session Layer

1. Dialog Control: This layer allows two systems to start communication with each other in half-duplex or full-duplex.
2. Token Management: This layer prevents two parties from attempting the same critical operation at the same time.
3. Synchronization: This layer allows a process to add checkpoints which are considered as synchronization points into stream of data.

Session LAYER

Design Issues with Session Layer

. To allow machines to establish sessions between them in a seamless fashion.

. Provide enhanced services to the user.

. To manage dialog control.

. To provide services such as Token management and Synchronization.

. Authentication . Permissions

. Session restoration (checkpointing and recovery)

Session LAYER(CO5)

Authentication

is the act of establishing or confirming something (or someone) as *authentic,* that is, that claims made by or about the thing are true. This might involve confirming the identity of a person, the origins of an artifact, or assuring that a computer program is a trusted one.

Permissions or Access control

One familiar use of authentication and authorization is access control. A computer system supposed to be used only by those authorized must attempt to detect and exclude the unauthorized. Access to it is therefore usually controlled by insisting on an authentication procedure to establish with some established degree of confidence the identity of the user, thence granting those privileges as may be authorized to that identity.

Session LAYER

In some cases, ease of access is balanced against the strictness of access checks. For example, the credit card network does not require a personal identification number, and small transactions usually do not even require a signature. The security of the system is maintained by limiting distribution of credit card numbers, and by the threat of punishment for fraud.

Checkpoints

Session layer is responsible for creating several checkpoints, checkpoints are also treated as recovery points i.e. in case of failure the system rollback to its previous checkpoint configuration or action.

Session LAYER

Remote Procedure Call (RPC)

Remote Procedure Call (RPC) provides a different paradigm for accessing network services. Instead of accessing remote services by sending and receiving messages, a client invokes services by making a local procedure call. The local procedure hides the details of the network communication. When making a remote procedure call:

1. The calling environment is suspended, procedure parameters are transferred across the network to the environment where the procedure is to execute, and the procedure is executed there.

Session LAYER

2. When the procedure finishes and produces its results, its results are transferred back to the calling environment, where execution resumes as if returning from a regular procedure call. The main goal of RPC is to hide the existence of the network from a program. As a result, RPC doesn't quite fit into the OSI model:

The message-passing nature of network communication is hidden from the user. The user doesn't first open a connection, read and write data, and then close the connection. Indeed, a client often does not even know they are using the network!

Session LAYER

3. RPC often omits many of the protocol layers to improve performance. Even a small performance improvement is important because a program may invoke RPCs often. RPC is especially well suited for client-server (e.g., query-response) interaction in which the flow of control alternates between the caller and callee. Conceptually, the client and server do not both execute at the sametime. Instead, the thread of execution jumps from the caller to the callee and then back again.

Session LAYER

- **Steps involved during RPC:**

The following steps take place during an RPC:

1. A client invokes a client stub procedure, passing parameters in the usual way. The client stub resides within the client's own address space.

2. The client stub marshalls the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.

3. The client stub passes the message to the transport layer, which sends it to the remote server machine.

Session LAYER

4. On the server, the transport layer passes the message to a server stub, which demarshalls the parameters and calls the desired server routine using the regular procedure call mechanism.

5. When the server procedure completes, it returns to the server stub (e.g., via a normal procedure call return), which marshalls the return values into a message. The server stub then hands the message to the transport layer.

6. The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.

7. The client stub de-marshalls the return parameters and execution returns to the caller.

Session LAYER

- **RPC Issues that must be addressed:**

Marshalling: Parameters must be marshalled into a standard representation.

Semantics: Call-by-reference not possible: the client and server don't share an address space. That is, addresses referenced by the server correspond to data residing in the client's address space.

Binding: How does the client know who to call, and where the service resides?

Transport protocol: What transport protocol should be used?

Exception handling: How are errors handled?

Presentation LAYER(CO1)

Topic Objective

- The student will be able to understand the design issue and Functions of Presentation layer
- Understand the various data compression techniques

Recap of Previous topic

- Session layer being the fifth layer of OSI model
- Supports the access management and communication
- RPC being one of the ways to maintain remote location access

Presentation Layer - OSI Model(CO1)

The primary goal of this layer is to take care of the syntax and semantics of the information exchanged between two communicating systems.

Presentation layer takes care that the data is sent in such a way that the receiver will understand the information(data) and will be able to use the data.

Languages (syntax) can be different of the two communicating systems. Under this condition presentation layer plays a role translator.

Presentation Layer - OSI Model

Functions of Presentation Layer

1. **Translation:** Before being transmitted, information in the form of characters and numbers should be changed to bit streams. The presentation layer is responsible for interoperability between encoding methods as different computers use different encoding methods. It translates data between the formats the network requires and the format the computer.
2. **Encryption:** It carries out encryption at the transmitter and decryption at the receiver.
3. **Compression:** It carries out data compression to reduce the bandwidth of the data to be transmitted. The primary role of Data compression is to reduce the number of bits to be transmitted. It is important in transmitting multimedia such as audio, video, text etc.

Presentation Layer - OSI Model

Design Issues with Presentation Layer

. To manage and maintain the Syntax and Semantics of the information transmitted.

. Encoding data in a standard agreed upon way. Eg: String, double, date, etc.

. Perform Standard Encoding on wire.

Presentation Layer - OSI Model(CO5)

Data compression techniques

Compression - to reduce the volume of data to be exchanged

1- **Lossless**

2- **Lossy**

Presentation Layer - OSI Model

- **Lossless compression**

  The integrity of the data is preserved as the compression and decompression algorithms are exact inverses of each other, no part of data is lost in the process. The different techniques are:

a) **<u>Run-Length coding</u>**

RLE simplest method to remove redundancy. Replaces a repeated sequence, run, of the same symbol with two entities : a count and the symbol itself.

Presentation Layer - OSI Model

b) Dictionary coding

LZW (Lempel-Ziv-Welch) encoding Based on creation of a dictionary of strings in the text. It encodes common sequence of characters instead of encoding each character separately. The dictionary is created as the message is scanned, and if a sequence of characters that is an entry in the dictionary is found in the message, the code of that entry is sent instead of the sequence. The creation of the dictionary is dynamic.

Presentation Layer - OSI Model

Algorithm

```
LZWEncoding(message)
{
   Initialize(Dictionary)
   Char = Input(first character)
   S = Char
   while(more characters in message) {

      Char = Input(next character);
      If((S +char)is in Dictionary)
      {
```

Presentation Layer - OSI Model

```
        S = S+char;
      }
      Else
      {
        addToDictionary(S+char);
          Output(index of S in Dictioanry);
          S = char;
        }
    }
Output(index of S in Dictionary);
}
```

Presentation Layer - OSI Model

Algorithm

```
LZWDecoding(code)
{
    Initialize(Dictionary);
    C = Input(first codeword);
    Output(Dictionary[C]);
    While( more codewords in code)
```

Presentation Layer - OSI Model

```
{
    S = Dictionary[C];
    C = Input(next codeword); If( C is in Dictionary)
    {     addToDictionary(S+firstSymbolOf Dictionary[C]);
              Output(Dictionary[C]);}
      Else
             {firstSymbolOfDictionary[C]);
             addToDictionary(S+firstSymbolOf(S));
             Output(S+firstSymbolOf (S));
              }
           }
    }
```

Presentation Layer - OSI Model

c) Huffman coding

It assigns shorter code to symbols that occur more frequently and longer codes to those that occur less frequently.

For Huffman coding first create a Huffman tree. The Huffman tree is a tree in which the leaves of the tree are the symbols. It is made so that the most frequent symbol is the closest to the root of the tree and the least frequent symbol is the farthest from the root.

Huffman tree

1. We put the entire character set in a row. Each character is now a node at the lowest level of the tree.
2. We select the two nodes with the smallest frequencies and join them to form a new node, resulting in a simple two-level tree. The frequency of the new node is the combined

Presentation Layer - OSI Model

frequencies of the original two nodes. This node, one level up from the leaves, is eligible for combination with the other nodes.

3. We repeat step 2 until all of the nodes, on every level, are combined into a single tree.

4. After the tree is made, we assign bit values to each branch. Since the Huffman tree is a binary tree, each node has a maximum of two children.

Coding table

After the tree has been made, a coding table is created to encode and decode the data. The code for each character can be found by starting at the root and following the branches that lead to that character. The code itself is the bit value of each

Presentation Layer - OSI Model

branch on the path, taken in sequence. Note that, The characters with high frequencies have shorter code than the characters with lower frequencies. And in this coding system no code is a prefix of another code.

One drawback of Huffman coding is that both encoder and decoder need to use the same encoding table.

Presentation Layer - OSI Model

d) Arithmetic coding

Introduced by Rissanen and Langdon in 1981, the entire message is mapped to a small interval inside [0,1). The small interval is then encoded as a binary pattern. Arithmetic coding is based on the fact that we can have an infinite number of small intervals inside the halfopen interval [0,1). Each of these small intervals can represent one of the possible messages we can make using a finite set of symbols.

Presentation Layer - OSI Model

Algorithm

```
ArithmeticEncoding(message)
{
   CurrentInterval=[0,1);
   While(more symbols in the message)
   {
     S=Input(next symbol);
     Divide current interval into subintervals
     subInt=subinterval related to s;
     currentInterval = subInt;
   }
  Output(bits related to the currentInterval);
}
```

Presentation Layer - OSI Model

```
ArithmeticDecoding(code)
{
    C=Input(code)
    Num=find real number related to code
    currentInterval=[0,1);
    while(true)
    {
        Divide the currentInterval into subintervals
        subInt=subinterval related to num;
        Output(symbol related to subint);
        If(symbol is the terminating symbol)
        currentInterval=subInt;
    }
}
```

Presentation Layer - OSI Model

- **Lossy Compression**

Lossless compression has limits on the amount of compression. Increasing the compression rate accuracy will be decreased. Text can't be compressed but images, video and audio can be.

1-Predictive Coding

Is used when an analog signal is digitized. When an analog signal is converted to digital using sampling, after sampling each sample needs to be quantized to create binary values. Compression can be achieved in the quantization step by using predictive coding. This technique is Pulse code modulation, where samples are quantized separately. The neighboring quantized samples however, are closely related and have similar values. In predictive coding, instead of quantizing each sample

Presentation Layer - OSI Model

separately, the differences are quantized. The differences are smaller than the actual samples and thus require fewer bits.

2-Delta modulation

This is the simplest method. Let xn be the value of the original function at sampling interval n, yn be the reconstructed value. The sender quantizes en, the difference between each sample (xn) and the preceding reconstructed value (yn-1).

The sender then transmits Cn. The receiver reconstructs sampe yn from the received Cn.

Other include Differential PCM, Adaptive DM etc.

Other multimedia is compressed by using various techniques like images are compressed as JPEG, GIF. And videos as MPEG4, Audio as MP3 etc.

Presentation Layer - OSI Model(CO5)

Cryptography

The Three Security Goals are Confidentiality, Integrity, and Availability

All information security measures try to address at least one of three goals:

. Protect the confidentiality of data
. Preserve the integrity of data
. Promote the availability of data for authorized use

Presentation Layer - OSI Model

Attacks:

One of the three security goals can be threatened by security attacks

Threat to confidentiality

1. **Snooping** - unauthorized access to or interception of data
2. **Traffic Analysis** - by monitoring online traffic some other data can be accessed

Presentation Layer - OSI Model

Threat to integrity

1. **Modification** - the attacker modifies the information to make it beneficial to ownself.

2. **Masquerading** - spoofing when the attacker impersonates somebody else

3. **Replaying** - attacker attains the copy of the message and replays it.

4. **Repudiation** - performed by one of the party in the communication

Presentation Layer - OSI Model

Threat to availability

I.Denial of service -

Techniques

There are some security services to achieve security goals and prevent attacks. The techniques are

Cryptography - secret writing. Transform messages to make them secure and immune to attacks. It includes - symmetric key encipherment, assymetric key encipherment and hashing.

Steganography - covered writing. Concealing the message itself by covering it with something else.

Presentation Layer - OSI Model

Cryptography Symmetric-

key ciphers:

Uses the same key for both encryption and decryption and key can be used for bi-directional communication.

Original message is plaintext and message sent is ciphertext using encryption algorithm and a shared key. And at receiver end ciphertext converted to plaintext by decryption algorithm and shared key.

1. **Substitution ciphers** - replaces one symbol with another.

a. **Monoalphabetic ciphers** - a character in the plaintext is always changed to the same character in the ciphertext regardless of its position in the text.

Presentation Layer - OSI Model

b.Polyalphabetic ciphers - each occurrence of a character may have a different substitute.

2.Transposition ciphers - reorders symbols

3.Stream and block ciphers

a. Stream ciphers
b. Block ciphers
c. Combination

4.Modern block ciphers - encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext. The algorithm uses a k-bit key. A modern block cipher is made of a combination of transposition units, substitution units and XOR operations, as well as shifting elements, swapping elements, splitting elements, and combining elements.

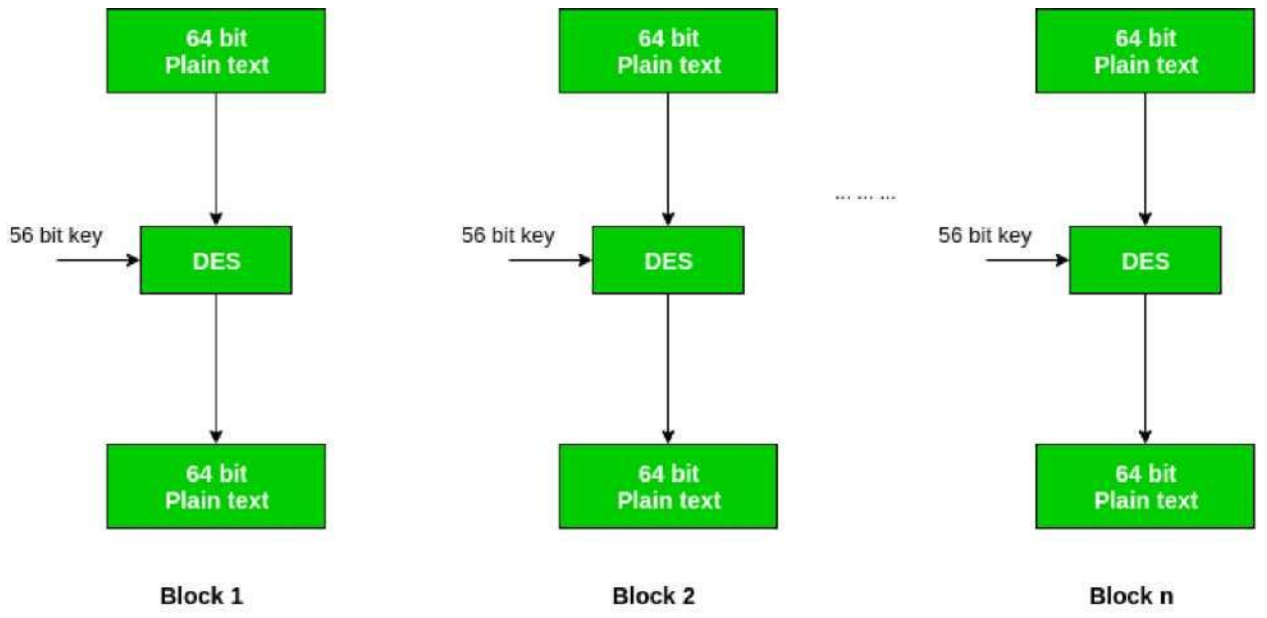
Presentation Layer - OSI Model

a. Data Encryption standard – DES

Data encryption standard (DES) has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been found slightly on decline.

DES is a block cipher, and encrypts data in blocks of size of 64 bit each, means 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits. The basic idea is

Presentation Layer - OSI Model



We have mention that DES uses a 56 bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56 bit key. That is bit position 8, 16, 24, 32, 40, 48, 56 and 64 are discarded.

Presentation Layer - OSI Model

Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key.

DES is based on the two fundamental attributes of cryptography: substitution (also called as confusion) and transposition (also called as diffusion). DES consists of 16 steps, each of which is called as a round. Each round performs the steps of substitution and transposition. Let us now discuss the broad-level steps in DES.

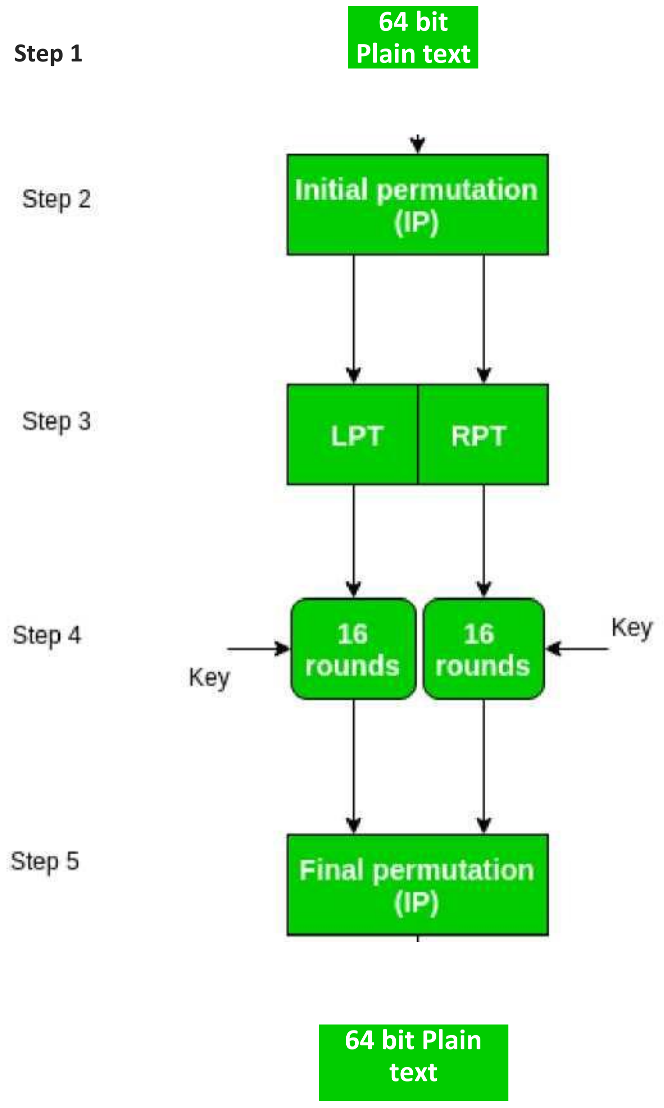
1. In the first step, the 64 bit plain text block is handed over to an initial Permutation (**IP**) function.
2. The initial permutation performed on plain text.
3. Next the initial permutation (IP) produces two halves of the permuted block; says Left Plain Text (**LPT**) and Right Plain Text (**RPT**).

Presentation Layer - OSI Model

4.Now each LPT and RPT to go through 16 rounds of encryption process.

5.In the end, LPT and RPT are rejoined and a Final Permutation (**FP**) is performed on the combined block 6.The result of this process produces 64 bit cipher text.

Presentation Layer - OSI Model



Presentation Layer - OSI Model

Initial Permutation (IP) -

As we have noted, the Initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as show in figure.

For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block and so on.

NIET Greater Noida

GET FUTURE READY

Presentation Layer - OSI Model

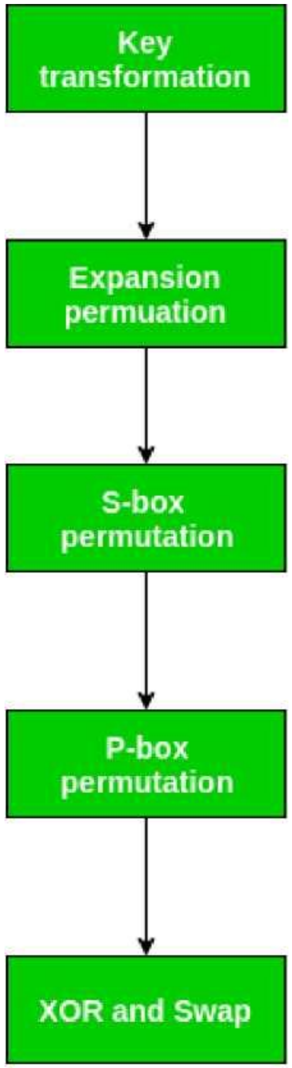
This is nothing but jugglery of bit positions of the original plain text block. the same rule applies for all the other bit positions which shows in the figure.

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	33	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Figure ■ Initial permutation table

As we have noted after IP done, the resulting 64-bit permuted text block is divided into two half blocks. Each half block consists of 32 bits, and each of the 16 rounds, in turn, consists of the broad level steps outlined in figure.

Presentation Layer - OSI Model



Presentation Layer - OSI Model

Step-1: Key transformation -

We have noted initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each a 56-bit key is available. From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called as key transformation. For this the 56 bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round.

Presentation Layer - OSI Model

For example, if the round number 1, 2, 9 or 16 the shift is done by only position for other rounds, the circular shift is done by two positions. The number of key bits shifted per round is show in figure.

Round	l	2	3	4	5	6	7	S	9	10	11	12	13	14	15	16
#key bits shifted	l	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Fig **Lire** - number of key bits, shifted per round

After an appropriate shift, 48 of the 56 bit are selected. for selecting 48 of the 56 bits the table show in figure given below. For instance, after the shift, bit number 14 moves on the first position, bit number 17 moves on the second position and so on. If we observe the table carefully, we will realize that it contains only 48 bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce a 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as

Presentation Layer - OSI Model

selection of a 48-bit sub set of the original 56-bit key it is called Compression Permutation.

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	S	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Figure ■ compression permutation

Because of this compression permutation technique, a different subset of key bits is used in each round. That's make DES not easy to crack.

Presentation Layer - OSI Model

Step-2: Expansion Permutation -

Recall that after initial permutation, we had two 32-bit plain text areas called as Left Plain Text(LPT) and Right Plain Text(RPT). During the expansion permutation, the RPT is expanded from 32 bits to 48 bits. Bits are permuted as well hence called as expansion permutation. This happens as the 32 bit RPT is divided into 8 blocks, with each block consisting of 4 bits. Then, each 4 bit block of the previous step is then expanded to a corresponding 6 bit block, i.e., per 4 bit block, 2 more bits are added.

Presentation Layer - OSI Model

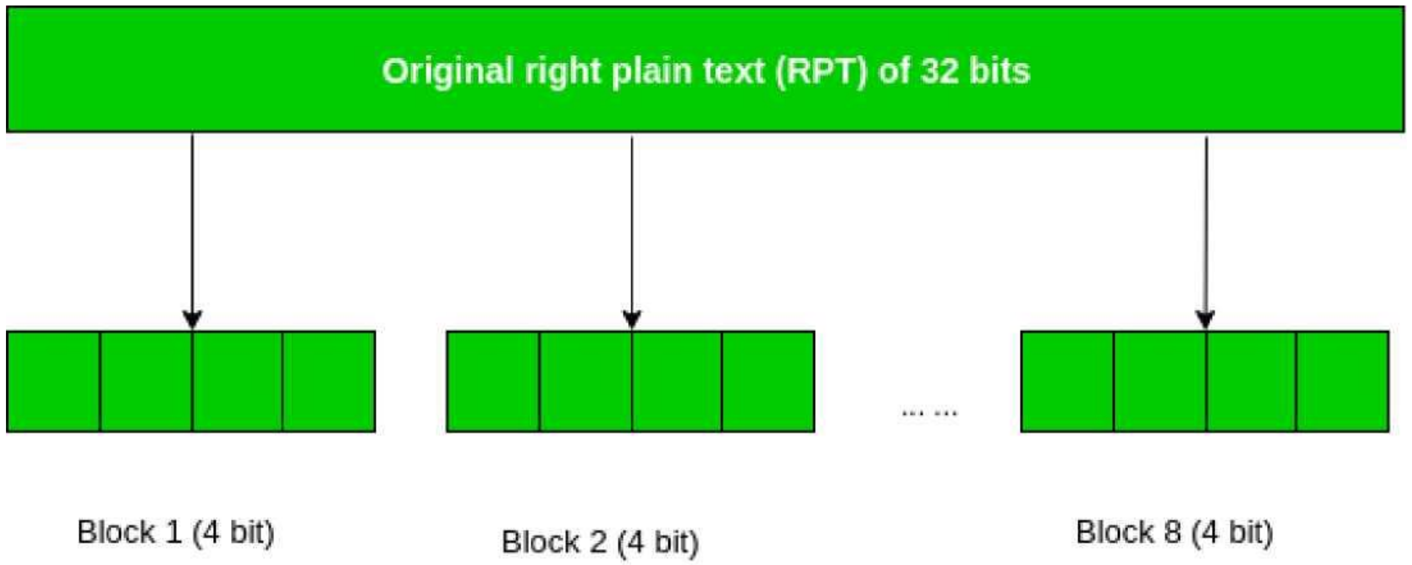


Figure - division of 32 bit RPT into 8 bit blocks

This process results into expansion as well as permutation of the input bit while creating output. Key transformation process compresses the 56-bit key to 48 bits. Then the expansion permutation process expands the 32-bit RPT to 48-bits.

Presentation Layer - OSI Model

Now the 48- bit key is XOR with 48-bit RPT and resulting output is given to the next step, which is the S-Box substitution.

Presentation Layer - OSI Model

Asymmetric-Key Ciphers

Based on personal secrecy and based on applying mathematical function to numbers. It uses two separate keys : one private and one public.

1. **RSA (Rivest-Shamir-Adleman)** is an algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of the keys can be given to anyone. The other key must be kept private. The algorithm is based on the fact that finding the factors of a large composite number is difficult: when the integers are prime numbers, the problem is called prime factorization. It is also a key pair (public and private key) generator.

Presentation Layer - OSI Model (CO5)

RSA involves a public key and private key. The public key can be known to everyone; it is used to encrypt messages. Messages encrypted using the public key can only be decrypted with the private key. The keys for the RSA algorithm are generated the following way:

a. Choose two different large random prime numbers p and

q

b. Calculate n = pq

i. n is the modulus for the public key and the private keys

c. Calculate the totient: *(n) =(p-1)(q-1).

d. Choose an integer e such that 1 < e < *(n), and e is co-prime to *(n) ie: e and *(n) share no factors other than 1; gcd(e, *(n)) = 1.

i. e is released as the public key exponent

Presentation Layer - OSI Model

e.Compute d to satisfy the congruence relation de *=.1.* Mod(*(n)) ie: de = 1+k*(n) for some integer k. (Simply to say : Calculate d= (1+k(*(n))/e)

i. d is kept as the private key exponent

The public key is made of the modulus n and the public (or encryption) exponent e .

The private key is made of the modulus n and the private (or decryption) exponent d which must be kept secret.

. For efficiency a different form of the private key can be stored: o p and q: the primes from the key generation, o d mod(p-1) and d mod(q-1): often called *dmpl* and *dmql. o* q^{-1} mod(p): often called *iqmp*

. All parts of the private key must be kept secret in this form. p and q are sensitive since they are the factors of n, and

Presentation Layer - OSI Model

allow computation of d given e. If p and q are not stored in this form of the private key then they are securely deleted along with other intermediate values from key generation.

. Although this form allows faster decryption and signing by using the Chinese Remainder Theorem (CRT) it is considerably less secure since it enables side channel attacks (en) . This is a particular problem if implemented on smart cards, which benefit most from the improved efficiency. (Start with $y = x^e \bmod(n)$ and let the card decrypt that. So it computes $y^d \bmod(p)$ or $y^d \bmod(q)$ whose results give some value z. Now, induce an error in one of the computations. Then gcd(z-x,n) will reveal p or q.)

Presentation Layer - OSI Model

Encrypting message

Alice gives her public key (n & e) to Bob and keeps her private key secret. Bob wants to send message M to Alice.

First he turns M into a number m smaller than n by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to:

$C=m^e \bmod(n)$

This can be done quickly using the method of exponentiation by squaring. Bob then sends c to Alice.

Presentation Layer - OSI Model

Decrypting message

Alice can recover m from c by using her private key d in the following procedure:

Given m, she can recover the original distinct prime numbers, applying the Chinese remainder theorem to these two congruences yields

$m^{ed} = m \bmod(pq).$

Thus,

$c^{d} = m \bmod n.$

An Example of RSA

Presentation Layer - OSI Model

1. Choose two random prime numbers.
2. :p = 61 and q = 53 Compute n = pq
3. :n = 61*53 = 3233
4. Compute the totient : *(n) =(p-1)(q-1).
5. +(n) = (61-1)(53-1) = 3120
6. Choose e>1 coprime to 3120
7. :e=17
8. Choose d to satisfy de E1. Mod(*(n))
9. :d=2753
10. :17*2753 = 46801= 1+15*3120.

The public key is (n=3233, e=17). For a padded message m the encryption function $c=m^e$ mod n becomes:

$c= m^{17}$ mod 3233

NIET Greater Noida GET FUTURE READY

Presentation Layer - OSI Model

The private key is (n=3233, d=2753). The decryption function $m=c^d$ mod n becomes:

$m = c^{2753}$ mod 3233

For example, to encrypt m = 123, we calculate $c = 123^{17}$ mod 3233 = 855 To decrypt c =855, we calculate

$M = 85\ 5^{2753}$ mod 3233 = 123

Presentation Layer (CO1)

Recap of Previous topic

- Presentation layer being the sixth layer of OSI model
- Supports the encryption and decryption of data
- Data compression techniques used to transfer data securely and compactly