



Chapter

COMPUTER SYSTEMS

1

UNIT INTRODUCTION

- A computer is fundamentally a digital machine because it operates based on digital data and binary logic. A computer processes data digitally by converting input information into binary code (0s and 1s), storing it in memory, and utilizing its central processing unit (CPU) to execute a series of binary-based instructions.

1.1 DATA REPRESENTATION IN A DIGITAL COMPUTER

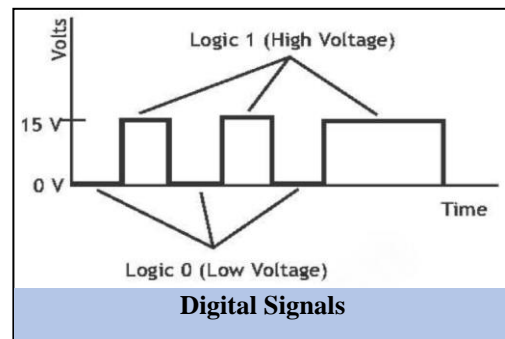
- A Digital Computer is an electronic machine as it consists of millions of electronic switches. An electronic switch is very similar to an electric switch that is used to turn ON/OFF light.
- Thus, switch has only two states. Binary number system used for counting also consists of two digits, 0 and 1.

Example

The binary code 1000100, may represent the letter D. A computer must be able to recognize codes that represent numbers, letters and special characters. These codes are known as alphanumeric codes.

Alphanumeric codes include the following characters:

- (i) Lower-case letters, (a to z)
 - (ii) Upper-case letters, (A to Z)
 - (iii) Numeric digits, (0 to 9)
 - (iv) Special characters (punctuation marks)
 - (v) Special symbols (% , \$, & , # , + , etc)
- The American Standard Code for Information Interchange (ASCII) is the most commonly used alphanumeric code. It uses 7 bits or 8-bits to represent a character and the total number of characters it represents is 128.



MULTIPLE CHOICE QUESTIONS

- What is the purpose of alphanumeric codes in a computer?**
 - Manage the hardware components
 - Represent number, letter & special character
 - Perform arithmetic calculations
 - Control the power supply
- What is the primary function of electronic switches in a digital computer?**
 - Convert analog signals to digital
 - Represent ON & OFF states
 - Increase voltage levels
 - Store data
- What does the ASCII code use to represent a character?**
 - 4 bits
 - 8 bits
 - 7 or 8 bits
 - 16 bits

SHORT QUESTIONS

- How many operations computer can perform?
- How does data represent in digital computers?
- Why ASCII code is used in computers?

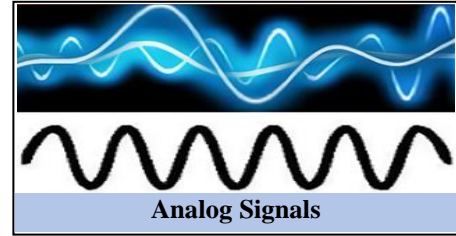
1.2 ANALOG AND DIGITAL SIGNALS

Analog Signals

Analog signals are continuous signals that vary smoothly over time. These signals can take any value within a given range and are represented by a continuous waveform. Analog signals are often used in traditional audio .

Examples of Analog Signals

- Analog audio signals (e.g., analogy audio cassette tapes)
- Analog video signals (e.g., VHS tapes)
- Human speech
- Analog temperature readings
- Analog voltage signals

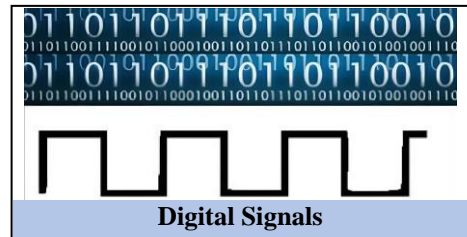


Digital Signals:

Digital signals are discrete, binary signals that represent information using a series of discrete values. These values are typically represented by 0s and 1s, where 0 usually represents the absence of a signal or a low voltage, and 1 represents the presence of a signal or a high voltage. Digital signals are commonly used in modern electronic devices and communication systems.

Examples of Digital Signals:

- Binary code (0s and 1s)
- Digital audio signals (e.g., MP3 files)
- Digital images (e.g., JPEG files)
- Digital video signals (e.g., MP4 files)
- Digital data in computers



Differences between Analog and Digital Signals:

ANALOG SIGNALS	DIGITAL SIGNALS
Representation	
Continuous waveform.	Discrete values (0s and 1s) .
Signal Nature	
Infinite possibilities of values within a range.	Limited discrete values.
Noise susceptibility	
Susceptible to noise and interference.	More resistance to noise and interference.
Transmission	
Degrades over long distances.	Can be transmitted over long distances with less degradation.
Storage and Reproduction	
Prone to quality loss during copying or storage.	Can be copied and reproduced without quality loss.
Scalability	
Not easily scalable.	Easily scalable.
Complexity of processing	
Analog processing is often more complex.	Digital processing is more typically more straightforward.
Examples	
Analog audio signals, analog temperature readings.	Digital audio signals, digital images.

MULTIPLE CHOICE QUESTIONS

- (1) Which type of signal is less susceptible to noise and interference?
 - (a) Analog signals
 - (b) Digital signals
 - (c) Both analog & digital signals
 - (d) Neither analog nor digital signals
- (2) Which of the following examples represents a digital signal?
 - (a) VHS tapes
 - (b) Analog temperature readings
 - (c) MP4 files
 - (d) Audio cassette tapes

- (3) Which signal degrades less over long distances?
- (a) Analog signals (b) Digital signals
(c) Computer signals (d) Hybrid signals

SHORT QUESTIONS

- (1) What is signal?
(2) Why digital signal are used in modern computing devices?
(3) What is the difference between analog and digital signal.
(4) What is signal? How analog signals are represented graphically?

1.3 DIGITAL LOGIC AND LOGIC GATES


Digital logic is fundamental in creating electronic devices such as calculator, computer, digital watches, etc. It is used to create digital circuits which consist of large number of logic gates. Logic gates are building blocks of digital circuits. A logic gate performs a particular logical function.

1.3.1 LOGIC GATES AND THEIR TRUTH TABLES

The commonly used logic gates are which are explained below:


- (i) AND gate
(ii) OR gate
(iii) NAND gate
(iv) NOR gate
(v) NOT gate
(vi) Exclusive-OR gate
(vii) Exclusive-NOR gate

AND Gate: AND gate operates such that the output will be at level 1 (HIGH) only when all inputs are 1 (HIGH). The mathematical expression for the two-input AND gate is written as $F = xy$.

	$F = xy$	<table> <tr><th>x</th><th>y</th><th>F</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	x	y	F	0	0	0	0	1	0	1	0	0	1	1	1
x	y	F															
0	0	0															
0	1	0															
1	0	0															
1	1	1															


Symbol, expression and truth table of AND Gate

OR Gate: OR gate produces a 1 output when any input is 1. Its mathematical expression is $F = x + y$, where the + stands for the OR operation and not normal addition. For a three-input OR gate, it would be $F = x + y + z$, and so on. The truth table for OR gate for two variables.

	$F = x + y$	<table> <tr><th>x</th><th>y</th><th>F</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1
x	y	F															
0	0	0															
0	1	1															
1	0	1															
1	1	1															


Symbol, expression and truth table of OR Gate

NAND Gate: NAND gate combines the AND and NOT gates, such that the output will be 0 only when all the inputs are 1. Its logic expression is $F = \overline{xy}$ which indicates that inputs x and y are first ANDed and then the result is inverted. Inversion is indicated by a bar. Thus, an AND gate always produces an output that is the inverse (opposite) of an AND gate. The truth table for NAND gate for two variables.

	$F = \overline{xy}$	<table> <tr><th>x</th><th>y</th><th>F</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0
x	y	F															
0	0	1															
0	1	1															
1	0	1															
1	1	0															


Symbol, expression and truth table of NAND Gate

NOR Gate: NOR gate combines the OR and NOT gates, such that the output will be 0 when any input is 1. Its logic expression is $F = \overline{x + y}$, which indicates that x and y are first ORed and then the result inverted. Inversion is indicated by a bar.

	$F = \overline{x + y}$	<table> <tr><th>x</th><th>y</th><th>F</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	0
x	y	F															
0	0	1															
0	1	0															
1	0	0															
1	1	0															


Symbol, expression and truth table of NOR Gate

NOT Gate: NOT gate is a single input gate. It converts LOW to HIGH and vice versa as shown in the truth table. Therefore, it is commonly known as an inverter. Its logic expression is $F = \overline{x}$.

	<table border="1"><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	F	0	1	1	0
x	F						
0	1						
1	0						

Symbol, expression and truth table of NOT Gate

Exclusive-OR Gate: Exclusive-OR (XOR) gate has a graphic symbol similar to that of OR gate, except for the additional curved line on the input side. It produces a 1 output only when the two inputs are at different logic levels.

	$F = x\overline{y} + \overline{x}y$	<table> <tr><th>x</th><th>y</th><th>F</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0
x	y	F															
0	0	0															
0	1	1															
1	0	1															
1	1	0															

Symbol, expression and truth table of XOR Gate

1.3.2 TRUTH TABLE

A *truth table* represents a Boolean function or digital logic circuit in table form. It shows how a logic circuit's output or Boolean expression responds to all possible combination of the inputs using logic '1' for true and logic '0' for false.

Properties:

- (i) Truth table consists of rows and columns.
- (ii) It shows relationship between inputs and output from a Boolean function or digital logic circuit.
- (iii) It shows output for all the possible combinations of inputs using 0 for LOW and 1 for HIGH
- (iv) All the combinations of inputs are listed in columns on the left, working in the middle and output is shown in the right most column.
- (v) The input columns are constructed in the order of binary counting with a number of bits equal to the number of inputs.

Example : Truth Table for the Boolean function. $F = (x \cdot y \cdot \bar{z}) + (x \cdot y \cdot z) + (\bar{x} \cdot y)$

INPUTS					WORKING			OUTPUTS
x	Y	Z	\bar{x}	\bar{z}	$(x \cdot y \cdot \bar{z})$	$(x \cdot y \cdot z)$	$(\bar{x} \cdot y)$	$F = (x \cdot y \cdot \bar{z}) + (x \cdot y \cdot z) + (\bar{x} \cdot y)$
0	0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	1	1	0	0	1	1
0	1	1	1	0	0	0	1	1
1	0	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	1	1	0	0	1
1	1	1	0	0	0	1	0	1

Truth table

1.3.3 BOOLEAN IDENTITIES

Boolean identities are mathematical expressions or equations that are always true, regardless of the values of the variables involved.

- AND operations
- OR operations
- NOT operations

These identities are fundamental for simplifying and analyzing logical expressions.

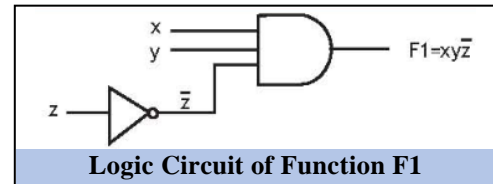
Identity Type	Boolean Identity
Identity Law (OR)	$A + 0 = A$
Identity Law (AND)	$A \cdot 1 = A$
Domination Law (OR)	$A + 1 = 1$
Domination Law (AND)	$A \cdot 0 = 0$
Complement Law (OR)	$A + \bar{A} = 1$
Complement Law (AND)	$A \cdot \bar{A} = 0$
Double Negation Law	$\bar{\bar{A}} = A$
Idempotent Law (OR)	$A + A = A$
Idempotent Law (AND)	$A \cdot A = A$
Associative Law (OR)	$(A + B) + C = A + (B + C)$
Associative Law (AND)	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
Distributive Law (AND over OR)	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
Distributive Law (AND over OR)	$A + (B \cdot C) = (A + B) \cdot (A + C)$
Absorption Law (OR)	$A + (A \cdot B) = A$
Absorption Law (AND)	$A \cdot (A + B) = A$
Negative Law (De Morgan's Theorem)	$\overline{A+B} = \bar{A} \cdot \bar{B}$
Negative Law (De Morgan's Theorem)	$\overline{A \cdot B} = \bar{A} + \bar{B}$

BOOLEAN IDENTITIES

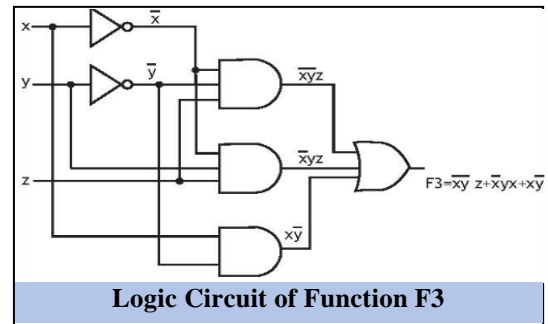
1.3.4 BOOLEAN FUNCTION AND ITS CONVERSION TO LOGIC CIRCUIT

A Boolean function is an expression formed with binary variables, the logical operators (OR, AND & NOT), parenthesis and equal sign. A binary variable can take the value of 0 or 1. For a given value of the variables, the function can be either 0 or 1.

Example:1: Conversion of Boolean function $F1 = xy\bar{z}$ to logic circuit. To convert this function to logic circuit, a single AND gate is required for the term xyz . A NOT gate is also required to convert \bar{z} to z , before it is input to the AND gate.



Example:2: Conversion of Boolean function $F3 = \bar{x}y\bar{z} + x\bar{y}z + xy\bar{y}$ to logic circuit. This function has three terms. Therefore, three AND gates are required for these terms. Two NOT gates are required to obtain \bar{x} and \bar{y} . The output of AND gates is to be input into an OR gate to perform the OR operation between all the three terms. The logic circuit of this function.



1.3.5 SIMPLIFICATION OF BOOLEAN FUNCTION USING KARNAUGH MAP (K-MAP)

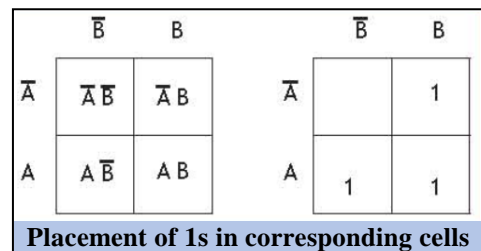
Karnaugh Map (K-Map)

Karnaugh Map (K-Map) was introduced by Maurice Karnaugh in 1953. It provides a simple method for simplifying Boolean functions. When a simplified Boolean function is converted into a logic circuit, it requires a smaller number of gates and hence costs less. K-map is a pictorial form of a truth table. It consists of square boxes called cells. All the possible combinations of variables involved in a Boolean function are written inside the cells in their respective positions. A two variable K-map contains $2^2=4$ cells, three variables $2^3=8$ cells and so forth.

Example:1

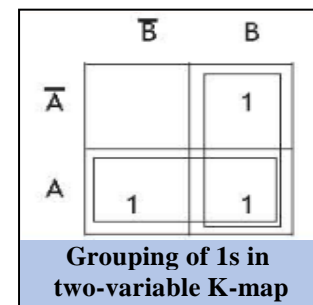
Simplify the Boolean function $F1 = AB + \bar{A}B + A\bar{B}$ using K-map

- The first step to simplify the Boolean function is to plot the terms of the function on the Karnaugh map. This function has three terms, for each term, a 1 will be placed in the corresponding cell.
- The next step is grouping cells for simplification. Grouping means combining cells in adjacent cells. The K-map contains a pair of 1s that is horizontally adjacent and another pair of 1s that is vertically adjacent.
- Combine two terms by eliminating the variable that is in both normal and complemented form. In the horizontal group, \bar{B} appears in both normal and complemented form. Therefore, \bar{B} will be eliminated in this group and only \bar{A} is left. Similarly, in the vertical group, A appears in both normal and complemented form. Therefore, A will be eliminated in this group and only B will be left.
- Finally, the result is written as the sum of variables as: **$F1 = A + B$**



The following are the rules for simplifying a two-variable Boolean function

- For each term of the function, place 1 in the corresponding cell in Karnaugh map.
- Make groups of two cells that contain 1. Groups may be horizontal or vertical but not diagonal.
- Groups may overlap.
- Eliminate the variable that is in normal and complemented form in the group.



- Write the simplified function in the form of sum of variables that were not eliminated in groups.
- If a K-map contains two 1s in diagonal cells then group cannot be formed which means the function cannot be simplified.

Simplification of Three-Variable Boolean Function using K-map

A three-variable K-map for variables \bar{A} , \bar{B} and C . It consists of eight cells having two rows and four columns. Rows are labelled with the complement and normal form of the variable A . Each column is labelled with two variables, B and \bar{C} , in their normal or complemented form. Each cell contains a product term of variables A , \bar{B} and \bar{C} in its respected cell.

	\bar{B}	\bar{B}	B	B
\bar{A}	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}B\bar{C}$	$\bar{A}BC$
A	$A\bar{B}\bar{C}$	$A\bar{B}C$	ABC	$AB\bar{C}$
	\bar{C}	C	C	\bar{C}

Three variable K-map

Example:2

Simplify the Boolean function. $F1 = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC$

- The first step to simplify the Boolean function is to plot all the terms of the function on the three variable Karnaugh map.
- Make two horizontal groups of two 1s.
- Combine two terms by eliminating the variable that is in both normal and complemented form in a group. In the group that is on the top, the variable C appears in both normal and complemented form.
- The simplified function can be written as the sum of the variable C from both groups as given. **$F2 = \bar{A}B + \bar{A}\bar{B}$** .

	\bar{B}	\bar{B}	B	B
\bar{A}	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}B\bar{C}$	$\bar{A}BC$
A	$A\bar{B}\bar{C}$	$A\bar{B}C$	ABC	$AB\bar{C}$
	\bar{C}	C	C	\bar{C}

Placement of 1s in corresponding cells

	\bar{B}	\bar{B}	B	B
\bar{A}			1	1
A		1		
	\bar{C}	C	C	\bar{C}

Grouping of 1s in four variable K-map

1.3.7 PRINCIPLE OF DUALITY IN BOOLEAN ALGEBRA

The Principle of *Duality* in Boolean algebra states that for any given Boolean expression/function, a dual expression can obtain by interchanging the:

- AND (.)
- OR (+) operators while complementing (negating) the variables. In other words, it highlights the symmetry between two pairs of operations:
- AND (conjunction)
- OR (disjunction) as well as 0 (false) and 1 (true).

Some examples to illustrate the Principle of Duality

Table 1.3 – Some examples to illustrate the Principle of Duality	
Expression	Dual
$1 = 0$	$0 = 1$
$0 = 1$	$1 = 0$
$1.0 = 0$	$0 + 1 = 1$
$A.0 = 0$	$A + 1 = 1$
$0.A = 0$	$1 + A = 1$
$A.1 = 0$	$A + 0 = 1$
$1.A = 0$	$1 + A = 1$
$A.A = 0$	$A + A = 1$
$A.B = B.A$	$A + B = B + A$
$X.(Y.Z) = (X.Y).Z$	$X + (Y + Z) = (X + Y) + Z$
$A.(A + B) = A$	$A + A.B = A$
$XY + Y + ZXY = 0$	$(X + Y).Y.(Z + X + Y) = 1$

1.3.8 USES OF LOGIC GATES

Logic gates are essential components in digital electronics and computing. They are used in numerous applications in various fields.

Memory Circuits: A **flip-flop** is a digital circuit that stores binary information and is widely used in digital electronics for building memory elements and sequential logic circuits. A latch is another type of digital circuit that stores binary information. Latches are often used in memory storage elements and data path circuits.

Clock Synchronization :Logic gates help in clock synchronization and signal processing in digital systems.

Data Encoding and Decoding: Logic gates are used to encode and decode data for transmission and reception in communication systems.

Multiplication and Division: Complex mathematical operations like multiplication and division can be performed using a combination of logic gates.

Digital Signal Processing (DSP):Logic gates are used in DSP circuits for filtering, modulation, and demodulation.

Data Encryption and Decryption :Cryptographic algorithms use logic gates for data encryption and decryption.

Calculator Circuits :Basic calculators use logic gates to perform arithmetic calculations.

Traffic Light Control :Logic gates are used in traffic light control systems to manage traffic flow.

Robotics : Logic gates play a crucial role in controlling the movement and decision-making of robots.

Security Systems :Logic gates are used in security systems to control access, alarms, and surveillance.

Automotive Electronics: In vehicles, logic gates are used for engine control, airbag deployment, and anti-lock brake systems (ABS).

Home Automation: Logic gates are employed in smart home systems to automate tasks like lighting and temperature control.

Medical Devices :Medical equipment uses logic gates for monitoring and controlling various functions.

Aerospace Applications: Logic gates are used in navigation systems, autopilots, and guidance systems for aircraft and spacecraft. Improve your skill box Digital electronics experiment

MULTIPLE CHOICE QUESTION

- (1) Which logic gate is represented by a triangle with an inverted input?
(a) AND gate (b) OR gate
(c) NOT gate (d) NAND gate
- (2) In a truth table, where are all the possible combinations of inputs listed?
(a) Output column (b) Working column
(c) Input column (d) Header row
- (3) What is the typical arrangement of cells in a two-variable K-Map?
(a) 1 row and 4 columns (b) 2 rows and 2 columns
(c) 2 rows and 4 columns (d) 4 rows and 1 column

SHORT QUESTIONS

- (1) What are logic gates?
- (2) What is difference between AND gate & OR gate?
- (3) What is truth table?
- (4) Define Boolean identities .
- (5) What is Boolean function?
- (6) What is K-map?
- (7) Discuss principle of duality in detail.

1.4 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

Software Development Life Cycle (SDLC) is the process of creating a new software or system. In software engineering the SDLC concept reinforces many kinds of software development techniques. These techniques process the framework for planning and controlling the creation of software.

The software development life cycle (SDLC) is the process of:

- (i) Planning
- (ii) Writing
- (iii) Modifying
- (iv) Maintaining software
- The term "life cycle" was first used in the 1950s to describe the stages involved in developing a new computer system.

1.4.1 DIFFERENT PHASES OF SDLC

The following are phases/steps in SDLC.

- (i) Defining the Problem Phase
- (ii) Planning Phase
- (iii) Feasibility Study Phase
- (iv) Analysis Phase
- (v) Requirement Engineering Phase
- (vi) Design Phase
- (vii) Development/Coding Phase
- (viii) Testing/Verification Phase
- (ix) Deployment/Implementation Phase
- (x) Documentation Phase
- (xi) Maintenance/Support Phase

- **DEFINING THE PROBLEM PHASE:** In this phase the problem to be solved or system to be developed is clearly defined. All the requirements are documented and approved from the customer or the company which consists of all the product requirements to be designed and developed during the development life cycle.
- **PLANNING PHASE:** In the project *planning phase*, the project's goal is identified, and the necessary requirements for product development are assessed. A thorough evaluation of resources, including personnel and costs, is conducted, accompanied by the conceptualization of the new product.
- **FEASIBILITY STUDY PHASE**
Feasibility study is an essential aspect of project planning and decision-making in the Software Development Life Cycle (SDLC).

Technical Feasibility: *Technical feasibility* assesses the practicality of implementing a proposed project from a technological standpoint.

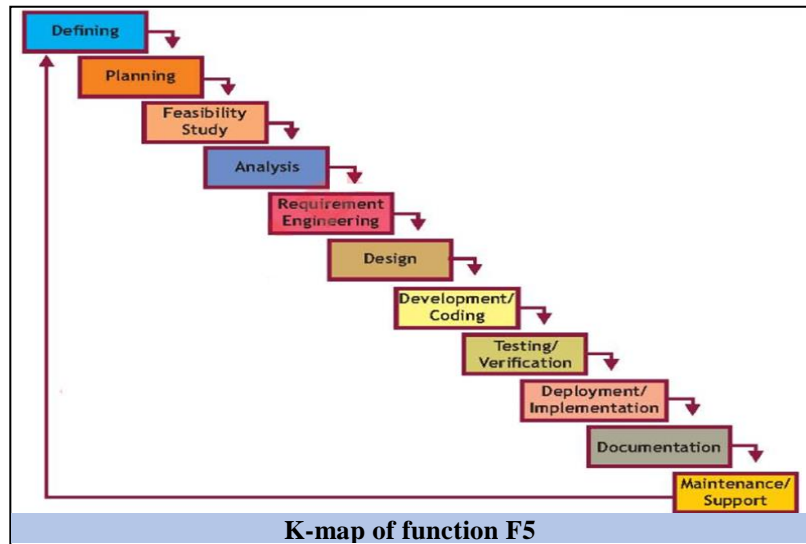
Technical Feasibility Considerations: The developing company will evaluate whether the existing infrastructure (hardware/software) can support the proposed system.

Economic Feasibility: *Economic feasibility* evaluates the financial viability of a proposed system by comparing its costs and benefits.

Operational Feasibility: *Operational feasibility* assesses the extent to which a proposed system aligns with the organization's operational processes and goals.

Legal Feasibility: *Legal feasibility* evaluates whether a proposed system complies with applicable laws, regulations, and standards.

Schedule Feasibility: *Schedule feasibility* assesses whether a system can be completed within a specified timeframe.



- **ANALYSIS PHASE**

During the analysis phase the project team determines the end-user requirements. Often this is done with the assistance of client focus groups, which provide an explanation of their needs and what their expectations are for the new system.

- (i) Can the proposed software system be developed with the available resources and budget?
- (ii) Will this system significantly improve the organization?
- (iii) Does the existing system even need to be replaced etc?

- **REQUIREMENT ENGINEERING PHASE**

Requirement Engineering is a crucial phase in the Software Development Life Cycle (SDLC) that focuses on gathering, analysing, documenting, and managing requirements for the development of the proposed system.

- (i) Requirement gathering
- (ii) Requirement validation
- (iii) Requirements management

Requirement Gathering

Requirements gathering is a pivotal stage in the Software Development Life Cycle (SDLC), aiming to identify and document the needs and expectations of stakeholders. Various techniques are employed for this purpose, and they can be broadly classified into different types.

- (i) **Interviews** :It involves direct conversations with stakeholders to gather information about their needs, expectations, and preferences.
- (ii) **Surveys and Questionnaires**: This method involves distributing surveys and questionnaires to collect information from a large number of stakeholders. s or questionnaires to surveys or question.
- (iii) **Observation**: This technique involves observing users in their natural work environment to understand how they currently perform tasks and identify areas for improvement.
- (iv) **Document Analysis**: This approach includes reviewing existing documentation, reports, and manuals to extract relevant information about the current system or processes.

Requirements Validation

Requirement validation focuses on scrutinizing the gathered requirements to ensure they align with the stakeholders' intentions. This process distinguishes itself from verification, which takes place after requirements have been accepted.

Requirements Management

Requirements management is a continual process aimed at guaranteeing that the software consistently fulfils the expectations of both the acquirer and users.

- **DESIGN PHASE**

The design phase in the Software Development Life Cycle (SDLC) is a crucial step where the system architecture is planned and detailed specifications are created based on the requirements gathered during the analysis phase.

- (i) **Unified Modelling Language (UML)**

Unified Modelling Language (UML) and various Design patterns play significant roles in this phase. Unified Modelling Language (UML) is a standardized visual modelling language widely used in software engineering and system design.

- (ii) **Design patterns**

Design patterns play a pivotal role in the SDLC by offering reusable solutions, promoting best practices, enhancing communication among team members, and contributing to the creation of maintainable and scalable software systems.

- (i) Algorithm
- (ii) Flow chart

Algorithms

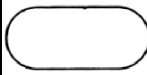
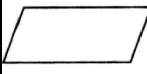
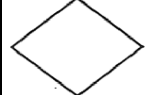
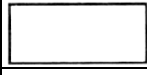
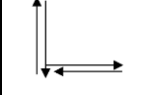

Algorithms are precise and systematic procedures designed to guide the step-by-step solution of a problem. They provide a structured and detailed set of instructions for solving a particular problem or performing a specific task.

Example

ALGORITHM TO FIND THE PERCENTAGE MARKS		ALGORITHM TO FIND THE STUDENT'S RESULT	
(i)	Start	(i)	Start
(ii)	Read Total Marks, TM	(ii)	Read Percentage Marks, PM
(iii)	Read Obtain Marks, OM	(iii)	If PM >= 40 Then Print "Pass" Else Print "Fail"
(iv)	Percentage Marks (PM) = OM / TM * 100	(iv)	End
(v)	Print "Percentage Marks", PM		
(vi)	End		

Flowcharts : A flowchart is a diagrammatic representation used to illustrate an algorithm or a process. It visually presents the sequence of steps in the algorithm through special shapes (symbols) and connects them with arrows to depict their sequence.

Some most commonly used flowchart symbols:

Symbol	Name	Description
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Parallelogram	Used for any Input/Output(I/O) operation, indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Process	Used for Arithmetic operations and data manipulations
	Flow line	Shows direction of flow
	Connector	Used to connect one part of the flowchart to the other
Flowchart		

• DEVELOPMENT/CODING PHASE

In the development/coding phase, developers translate the plans formulated in the design phase into actions. They develop the database structures, design codes for the data flow processes, and design the tangible user interface screens. This stage involves the preparation and iterative processing of test data to enhance the precision and efficiency of the written code. The actual coding is carried out using programming languages, a process commonly referred to as Computer programming.

```

1. #include<iostream>
2. int main()
3. {
4.     int cs;
5.     float total, per;
6.     cout << "Please Enter subject marks of Computer Science: \n";
7.     cin >>cs; // taking values from user
8.     total = 50;
9.     per = cs/total*100; // calculating percentage
10.    cout << "\nTotal Marks = " << total;
11.    cout << "\nMarks in Percentage = " << per<<"%";
12.    return 0;
13. }
```

Please Enter subject marks of Computer Science: 34
Total Marks = 50
Marks in Percentage = 68%

Output

Development/Coding Phase

• TESTING/VERIFICATION PHASE

In the testing/verification phase, all aspects of the system are tested for functionality and performance. Testing involves the execution of programming modules to detect errors, commonly known as bugs.

Software testers employ various techniques, including:

- Black box testing

- White box testing

Black Box Testing :Black Box Testing is a software testing method where the internal workings or logic of a system are not known to the tester. The focus is on evaluating the system's outputs based on specified inputs without considering its internal code structure.

White Box Testing: White Box Testing is a testing approach where the tester has knowledge of the internal code, logic, and architecture of the system being tested. It involves evaluating the system's internal structures, code paths, and overall code quality.

- **DEPLOYMENT/IMPLEMENTATION PHASE**

The Deployment/Implementation Phase involves a series of activities aimed at making the software/system accessible for use.

- Installation and activation of the hardware and software.
 - In some cases, the users and the computer operation personals are trained on the developed software system.
 - Conversion the process of changing from the old system to the new one is called conversion.
- Deployment/Implementation Methods**
- Direct:** In this method, the old system is entirely replaced by the new system simultaneously. The transition is abrupt, and once implemented, the old system becomes obsolete.
 - Parallel:**The parallel method involves running both the old and new systems concurrently for a certain period. This approach allows for the identification and rectification of major issues with the new system without risking data loss.
 - Phased:** The phased implementation method facilitates a gradual transition from the old system to the new one. It involves the incremental introduction of the new system while progressively phasing out the old system.
 - Pilot:**In the Pilot method, the new system is initially deployed for a small user group. These users engage with, assess, and provide feedback on the new system. Once the system is deemed satisfactory, it is then rolled out for use by all users.
- **DOCUMENTATION PHASE:** Documentation serves as a method of communication among the people responsible for developing, implementing, and maintaining the newly developed system. Installing and operating a newly designed system or modifying an established application requires a detailed record of that system's design.
 - **MAINTENANCE/SUPPORT PHASE:** In SDLC, the system maintenance is an ongoing process. The system is monitored continually for performance in accordance with user requirements and needed system modifications are incorporated. When modifications are identified, the system may reenter the planning phase.

1.4.2 SOFTWARE DEVELOPMENT MODELS

Software Development Models refer to the fundamental activities and approaches used in software development to plan, design, build, test, implement, and maintain software systems. These processes provide a structured framework for managing software projects.

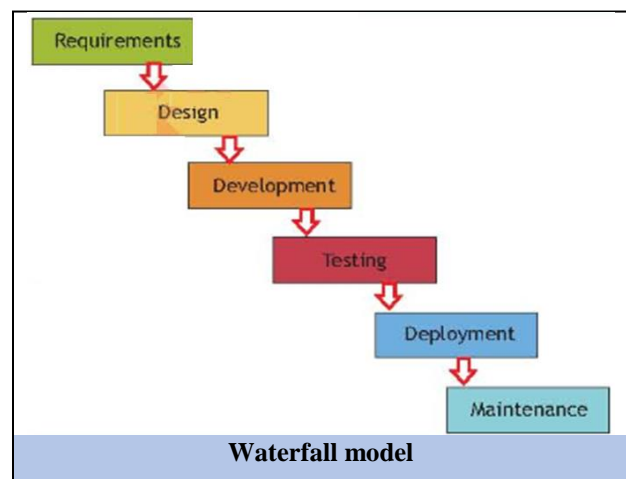
- Waterfall Model
- Agile Model

Waterfall model

The waterfall model is a sequential or linear development approach, in which development is seen as flowing steadily downwards (like a waterfall) through several phases.

This model has five phases

- Requirements
- Design
- Development
- Testing
- Deployment
- Maintenance



- **Requirements:** The aim of this phase is to understand the exact requirements of the customer and to document them properly.
 - (i) **Requirements Gathering and Analysis:** The development team collects the essential requirements from all the possible stakeholders and analyses them. During the analysis, they eliminate incomplete and inconsistent requirements.
 - (ii) **Requirements Specification:** After requirements analysis, the development team documents them in a Software Requirement Specification (*SRS*) document. It contains both functional and non-functional requirements of the software.
 - **Design:** In this step the development team transforms requirements into a format that can be easily converted into the code in chosen programming language. All the data collected is stored in a Software Design Document (SDD) document. This document helps to establish the software's architecture.
 - **Development:** The *Development phase* involves the actual coding of the software based on the design specifications. During this phase, design is implemented. If the SDD is complete, the coding phase proceeds smoothly, because all the information needed by software developers is contained in the *SDD*.
 - **Testing:** During *testing*, the code is thoroughly tested for bugs and debugged or modified if required. After this the Implementation (also called deployment) phase involves making the software in use in the real environment after it undergoes multiple rigorous tests. This phase is highly important as the quality of the end product is determined by the effectiveness of the testing carried out.
 - **Deployment:** The *Deployment phase* involves making the software available in the intended environment for end-users. This includes activities such as installation, configuration, and data migration to ensure a smooth transition from the development environment to live production. It marks the point at which the software is ready for use.
 - **Maintenance phase:** Over a period of time, a software product may require some updates to remain functional in the real-world environment. The *maintenance phase* takes care of this activity by timely tuning the software as per the requirement. It is very important phase of waterfall development model.
- There are two main types of maintenances**
- (i) **Corrective Maintenance:** It involves correcting errors left undiscovered during the development and testing stages.
 - (ii) **Perfective Maintenance:** This entails enhancing the functionality of the software product as and when required keeping in mind the future trends and customers demand.

Advantages of the Waterfall Model

- (i) It is suitable for projects where all the requirements are predefined and understood clearly.
- (ii) It is easy to understand and implement.
- (iii) All the activities to be performed in each phase are clearly defined.
- (iv) The release date and the final cost are already estimated before the development begins.
- (v) It is easier to assign tasks to different team members.
- (vi) All processes, actions, and results are well documented.

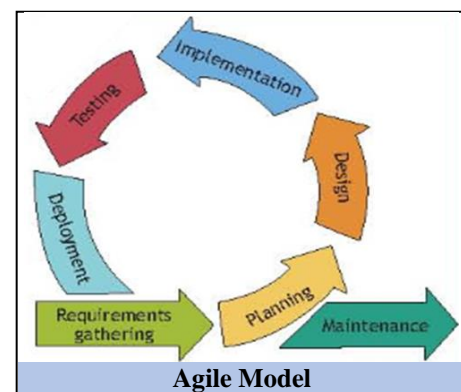
Disadvantages of the Waterfall Model

- (i) Inflexibility to changes is a significant drawback, making mid-project adjustments challenging.
- (ii) Long delivery times are inherent as the entire project must be completed before delivery.
- (iii) High risk of project failure due to late client visibility and potential dissatisfaction.
- (iv) Late defect detection is a concern as testing is performed toward the end of development.

Agile Model: The *Agile model* is a software development process that is based on the iterative and incremental approach, emphasizing flexibility, collaboration, and customer feedback.

This model has six phases is model

- (i) Requirements gathering
- (ii) Planning
- (iii) Design
- (iv) Implementation
- (v) Testing
- (vi) Deployment
- (vii) Maintenance



Requirements Gathering: In the Agile software development model, the "Requirements Gathering" phase is characterized by continuous collaboration and dynamic responsiveness. Rather than relying on a comprehensive upfront documentation, Agile teams engage in ongoing discussions with stakeholders.

Planning: Agile planning is iterative and done in short cycles called sprints. The team, including developers and stakeholders, collaboratively plans the work for the upcoming sprint. Priorities are set, and tasks are pulled from the product backlog into the sprint backlog based on their importance and feasibility.

Design: The design phase in Agile Model is a collaborative process that adapts to changing requirements and emerging insights. Unlike traditional approaches, Agile design activities are ongoing and occur at various levels throughout the development lifecycle.

Implementation: The implementation phase in Agile Model involves developing the features identified in the design phase. Agile development occurs incrementally in short cycles known as sprints. During the "Development" phase, cross-functional teams implement prioritized user stories based on the sprint backlog.

Testing: In the Agile context, testing is an integral and continuous activity that happens in parallel with development. The "Testing" phase involves both automated and manual testing, ensuring that each increment is thoroughly validated. Testers collaborate closely with developers to identify and address issues promptly.

Deployment: The Deployment phase in Agile enables incremental and continuous delivery of product increments. Completed features are deployed/implemented to a staging environment for validation before being released to production.

Maintenance: Maintenance in Agile involves addressing any issues identified after deployment. Regular review and feedback sessions characterize the "Maintenance" phase in Agile. At the end of each iteration, a sprint review is conducted to showcase completed features to stakeholders. .

Advantages of the Agile Model

- (i) Agile model offers flexibility and adaptability, allowing adjustments for evolving requirements.
- (ii) In this model continuous client involvement ensures higher satisfaction.
- (iii) It enables early and incremental delivery of a functional product.
- (iv) Continuous improvement is facilitated through regular demonstrations.
- (v) Improved communication is emphasized, fostering collaboration.

Disadvantages of the Agile Model

- (i) Dependency on customer availability poses a challenge.
- (ii) The potential for scope creep exists with the flexibility to accommodate
- (iii) Coordination challenges emerge with larger team sizes.
- (iv) Limited emphasis on documentation can be a drawback.
- (v) Not ideal for projects with stable and well-defined requirements.

MULTIPLE CHOICE QUESTIONS

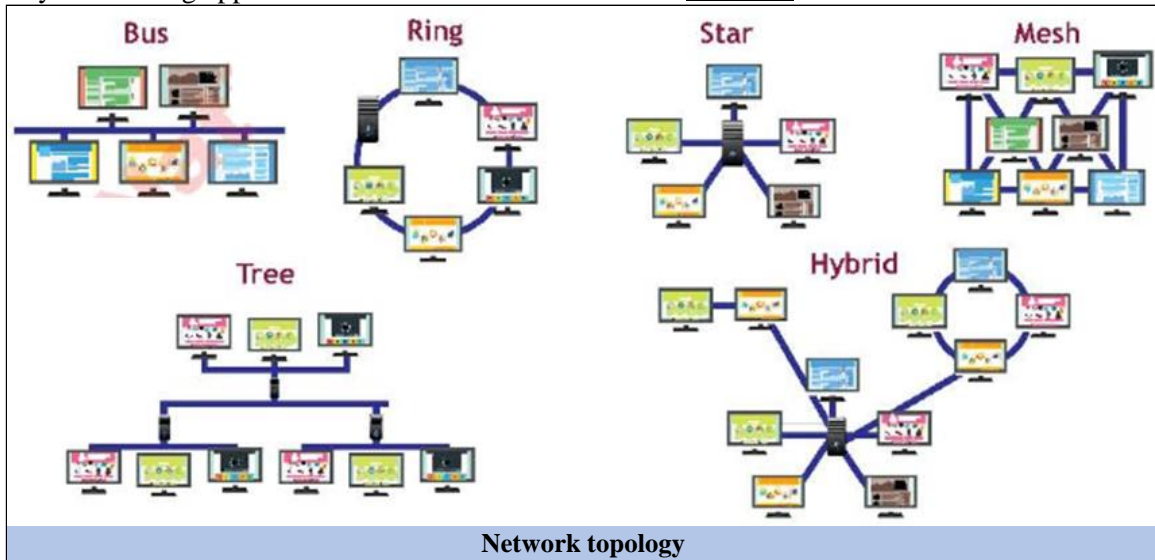
- (1) Which of the following is NOT typically assessed during the feasibility study phase?
 - (a) Technical resources & technology
 - (b) Legal compliance & regulation
 - (c) Financial cost & benefits
 - (d) System design & architecture
- (2) In a flowchart, which symbol is used to represent an input or output operation?
 - (a) Terminal
 - (b) Diamond
 - (c) Parallelogram
 - (d) Rectangle
- (3) In agile model, what challenges can arise with the larger team sizes?
 - (a) Enhanced communication
 - (b) Coordination difficulties
 - (c) Better flexibility
 - (d) Increased customer involvement

SHORT QUESTIONS

- (1) What is SDLC?
- (2) What is requirement engineering?
- (3) How design phase is developed in SDLC?
- (4) What is testing?
- (5) What is deployment?
- (6) Which more reliable agile model or water fall model?

1.5 NETWORK TOPOLOGY

- Network topology is a systematic arrangement of computers and other devices in a network. It is an important concept when building or managing a computer network. It is the backbone of any networking application. Network devices are called "Nodes".

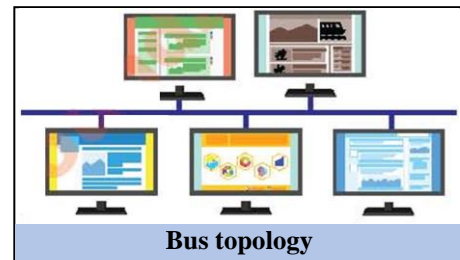


- (i) Physical network topology, as the name suggests, refers to the physical connections and interconnections between nodes and the network-the wires, cables, and so forth.
- (ii) Logical network topology is a little more abstract and strategic, referring to the conceptual understanding of how and why the network is arranged the way it is, and how data moves through it.

1.5.1 TYPES OF NETWORK TOPOLOGIES

Various network topologies are available for configuring a network. The most commonly used topologies include bus, star, ring, mesh, tree and hybrid.

Bus Topology: Bus topology is a network topology in which devices are connected to one cable or line running through the entire network. **Ethernet** is commonly used in bus topologies. Ethernet operates using the CSMA/CD (Carrier Sense Multiple Access with Collision Detection) protocol to manage access to the shared communication medium.



Advantages of Bus Topology

- (i) The main advantage of bus topology is that it is easy to install and maintain, it a great choice for smaller networks.
- (ii) Bus topology is very cost-effective because all the nodes are connected to one cable.
- (iii) Bus topology has a fast data transmission rate.
- (iv) Bus topology is relatively easy to expand by simply connecting additional nodes to the cable.

Disadvantages of Bus Topology

- (i) In bus topology all nodes share the same bandwidth and no network congestion.
- (ii) In this topology if one node goes down, it can cause a network failure because there is only one cable.

Applications of Bus Topology

- (i) Bus topology is a great choice for small office networks because of its cost-effectiveness.
- (ii) Bus topology can also be used in home networks due to its low cost and ease of use

Star Topology

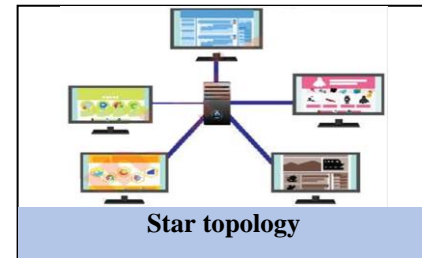
In a star topology, every device is linked to a central hub or switch. **Clients** include computers, laptops, smartphones, and other devices that connect to the network to access services or resources offered by a **Server**.

Advantages of Star Topology

- (i) Star topology is relatively easy to install.
- (ii) Star topology is very reliable because it has no single point of failure.
- (iii) Star topology has a high-performance rate due to its dedicated links between nodes and the hub.

Disadvantages of Star Topology

- (i) It can be more expensive than other types of networks because cost of hubs or switches .
- (ii) It is more prone to failure ,If the hub or switch fails, the entire network will be affected.

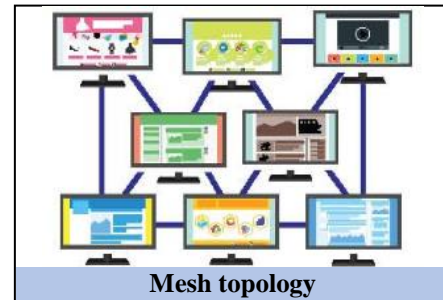


Mesh topology

Mesh topology is a network configuration where every node establishes connections with all other nodes within the network.

Advantages of Mesh Topology

- (i) Mesh topology provides a high level of redundancy as each node is connected to all other nodes in the network.
- (ii) Mesh topology is very reliable due to its redundant connections. The redundant connections in mesh topology contribute to its exceptional reliability.
- (iii) Mesh topology demonstrates high flexibility and can be easily scaled.



Disadvantages of Mesh Topology

- (i) The implementation of mesh topology can be costly due to the requirement for multiple links between nodes, increasing expenses related to hardware and installation.
- (ii) Setting up a mesh topology network can be complex due to the multiple connections required between nodes.

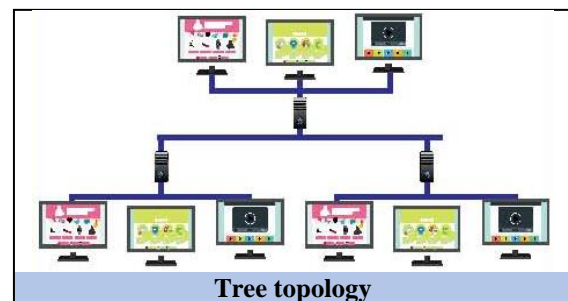
Tree Topology

Tree topology (or hierarchical topology) is a network topology that utilizes a hierarchical or tree-like layout.

This structure exhibits a hierarchy of nodes, with a central or top-level node commonly referred to as the "**Root node**".

Advantages of Tree Topology

- (i) Tree topology is more scalable , it allows for the addition and removal of nodes without disrupting the entire network.
- (ii) Tree topology provides a high level of reliability due to its redundant connections of nodes.
- (iii) Tree topology is cost-effective as it eliminates the need for complex wiring .

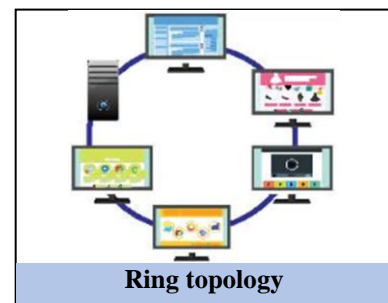


Disadvantages of Tree Topology

- (i) Installing a tree topology network can be complex due to the multiple connections .
- (ii) Troubleshooting a tree topology network can be difficult as there are multiple paths for data transmission.

Ring Topology

Ring topology is a network topology in which all devices are connected to one another in a circular loop. Ring topology is often used Fiber Distributed Data Interface (FDDI) networks, also called dual ring network. In a Ring Topology, the key term associated with the communication protocol is "**Token Ring**".



Dual Ring topology

Ring topology often use Fiber Distributed Data Interface (**FDDI**) networks, also called dual ring network. In a dual ring topology, two separate rings are connected to form one large loop. This provides greater security and efficiency as data can travel in both directions around the loop in a **clockwise** or **counter clockwise** direction

Advantages of Ring Topology

- (i) Ring topology provides a high level of reliability as each device has an alternate route in case of failure.
- (ii) Ring topology is cost-effective as it eliminates the need for complex wiring and allows for more efficient data transmission.

Disadvantages of Ring Topology

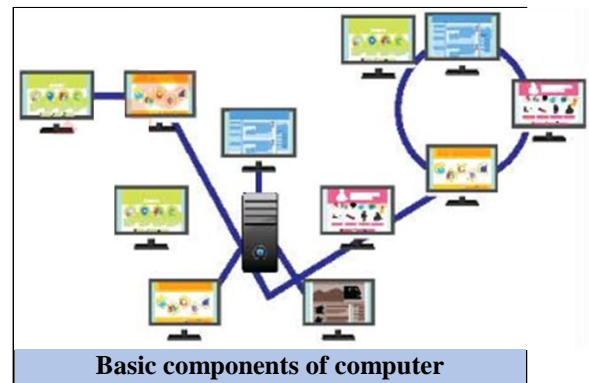
- (i) Ring topology faces the challenge of limited bandwidth since each device on the ring must share the same communication path.
- (ii) If this connection is disrupted or broken at any point, the entire network may become unavailable.
- (iii) The troubleshooting process for a ring topology network is difficult and can be challenging.

Hybrid Topology

Hybrid Topology integrates different network configurations to establish a more efficient and dependable network.

Advantages of Hybrid Topology

- (i) Hybrid topology is flexible and easily adaptable to changes in the network.
- (ii) With multiple pathways for data transmission, hybrid topology offers higher reliability.
- (iii) The configuration of hybrid topology facilitates high scalability, accommodating modifications or expansions in the network.
- (iv) By merging two or more basic networks, hybrid topology reduces the overall cost of a network by reducing the amount of hardware and wiring needed.
- (v) Hybrid topology enhances security by providing multiple layers of protection, making unauthorized access to the network more challenging.



Disadvantages of Hybrid Topology

- (i) Hybrid topology can be more complex to set up and maintain.
- (ii) Troubleshooting a hybrid topology network is more challenging.

1.5.2 SCALABILITY AND RELIABILITY OF NETWORK TOPOLOGIES

Scalability and reliability are two important aspects of networking systems.

Scalability: Star topology is moderately scalable. Adding more devices is possible by connecting them to the central hub or switch.

Reliability: Star topology is relatively reliable. If a device or cable fails, it typically only affects that specific device's connectivity, not the entire network capacity.

Scalability and Reliability in Bus Topology

Scalability: Bus topology is not highly scalable. Adding more devices can lead to signal degradation, as electrical signals must travel the entire length of the bus.

Reliability: Bus topology is less reliable because a single break in the main cable can disrupt the network.

Scalability and Reliability in Ring Topology

Scalability: Ring topology is moderately scalable. Adding more devices to the ring is possible.

Reliability: Ring topology can be highly reliable. Data can travel in both directions, reducing the risk of a single point of failure.

Scalability and Reliability in Mesh Topology

Scalability: Mesh topology offers high scalability. Devices can be added with ease, and each device has multiple paths to communicate with others, reducing congestion.

Reliability: Mesh topology is highly reliable. The redundancy of multiple paths ensures that if one path or device fails, data can take an alternative route, minimizing downtime.

Scalability and Reliability in Hybrid Topology

Scalability: Hybrid topologies can be highly scalable, depending on the combination used. It allows for combining the strengths of multiple topologies.

Reliability: Reliability in hybrid topologies varies based on the components used. Redundancy from mesh components can enhance reliability.

1.5.3 TESTING THE SCALABILITY AND RELIABILITY OF A NETWORK SYSTEM

Testing the scalability and reliability of a network system is essential to ensure that it can handle increasing workloads and maintain high availability under various conditions.

TESTING SCALABILITY

- (i) **Load Testing:** Conduct load testing to evaluate the network's performance under heavy traffic conditions. Tools like Apache, JMeter, LoadRunner, or locust.io can simulate a large number of users or devices accessing the network simultaneously.
- (ii) **Stress Testing:** *Stress testing* involves pushing the network beyond its capacity to identify breaking points. Gradually increase the load or traffic until the network performance starts to experience significant latency or fail.
- (iii) **Scalability Testing:** Perform scalability testing by adding resources (e.g., servers, switches, or routers) to the network and measuring its ability to handle increased demand.
- (iv) **Benchmarking:** Compare network performance against industry standards or competitors to assess how well it scales.
- (v) **Realistic Scenarios:** Use real-world usage scenarios to test scalability, considering peak traffic times and growth projections.
- (vi) **Performance Monitoring:** Implement performance monitoring tools to continuously monitor the network's scalability in production environments.

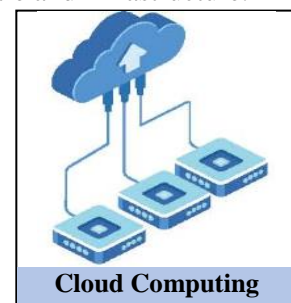
TESTING RELIABILITY

- (i) **Availability Testing:** Test the network's availability by simulating various failure scenarios, including hardware failures, network congestion, and server crashes. Measure the time it takes to recover from failures.
- (ii) **Redundancy Testing:** Evaluate the effectiveness of redundancy mechanisms, such as load balancing, failover, and backup systems. Simulate the failure of redundant components to ensure seamless failover.
- (iii) **Disaster Recovery Testing:** Test disaster recovery plans to ensure data and services can be restored in the event of disastrous failures or data breaches.
- (iv) **Fault Tolerance Testing:** Verify the network's ability to continue functioning even when individual components fail. Intentionally inject faults or errors into the network and observe how it responds.
- (v) **Security Testing:** Security is a key aspect of reliability. Perform penetration testing and perform vulnerability assessments to identify and address security weaknesses.
- (vi) **Load Balancing Testing:** Verify that load balancing mechanisms evenly distribute traffic across servers or resources to prevent overloading and maintain reliability.
- (vii) **Documentation and Reporting:** Maintain thorough documentation of tests performed and their results. Provide detailed reports to stakeholders.

1.5.4 CLOUD COMPUTING

Cloud computing is a technology model that provides access to computing resources and services over the internet, rather than owning and maintaining physical hardware and infrastructure.

- Servers
 - Storage
 - Databases
 - Networking
 - Software
 - Analytics
- through the internet.



Key characteristics of cloud computing include

- (i) **On-Demand Self-Service:** Users can provision and manage computing resources as needed, without requiring human intervention from service providers.
- (ii) **Broad Network Access:** Services and resources are accessible over the network and can be accessed by various devices with internet connectivity.
- (iii) **Resource Pooling:** Computing resources are shared and pooled to serve multiple customers. Resources dynamically adjust to meet demand.
- (iv) **Rapid Elasticity:** Resources can be quickly scaled up or down to accommodate changes in demand, providing flexibility and cost efficiency.
- (v) **Measured Service:** Usage of computing resources is monitored, controlled, and billed based on the actual usage.
- (vi) **Security:** Cloud providers implement robust security measures to protect data and resources.

Cloud computing is typically categorized into three service models and four models:

SERVICE MODELS

- (i) **Infrastructure as a Service (IaaS):** This model delivers virtualized computing resources, including virtual machines, storage, and networks, via the internet.
- (ii) **Platform as a Service (PaaS):** It offers a platform that includes tools, services, and frameworks for application development, without the complexity of managing underlying infrastructure.
- (iii) **Software as a Service (SaaS):** SaaS delivers software applications through the internet on a subscription basis, eliminating the need for users to locally install, maintain, or update software.

DEPLOYMENT MODELS

- (i) **Public Cloud:** In this model cloud resources are owned and operated by a third-party cloud service provider and are made available to the general public.
- (ii) **Private Cloud:** In this model cloud resources are used exclusively by a single organization, providing more control and customization over the infrastructure.
- (iii) **Hybrid Cloud:** It combines elements of both public and private clouds, allowing data and applications to be shared between them.
- (iv) **Community Cloud:** In this the cloud infrastructure is shared by several organizations with common interests, such as regulatory requirements or industry standards.

Examples of Cloud Computing Services

- (i) **Amazon Web Services (AWS):** Offers a wide range of cloud services, including computing power, storage, databases, machine learning, and more.
- (ii) **Microsoft Azure:** Provides cloud services for computing, analytics, storage, and networking, along with tools for building, deploying, and managing applications.
- (iii) **Google Cloud Platform (GCP):** Offers cloud services for computing, storage, machine learning, and data analytics, along with various development tools.
- (iv) **IBM Cloud:** Provides a range of cloud computing services, including IaaS, PaaS, SaaS, and hybrid cloud solution.

1.5.6 SCALABILITY AND RELIABILITY IN CLOUD COMPUTING

Scalability and reliability are two important characteristics of cloud computing, each serving a distinct but interconnected purpose in ensuring that cloud services meet the demands of users and applications effectively.

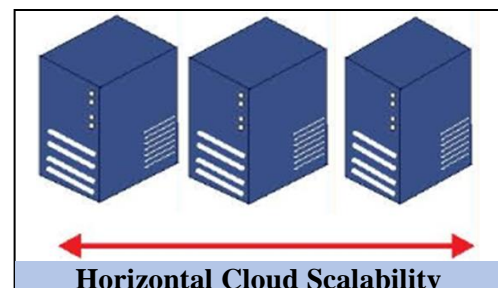
- **Scalability in Cloud Computing**

Scalability in cloud computing refers to the ability of a cloud system or service to handle increasing workloads, either by expanding resources (scaling out) or upgrading existing ones (scaling up).

- (i) **Horizontal Scalability (Scaling Out)**

Horizontal scaling means increasing the number of servers that run the application, and distributing the workload among them. .. However, horizontal scaling also has some challenges and trade-offs. It needs to design the application to support distributed architecture, and use techniques such as.

- Load balancing
- Replication
- Sharing
- Caching

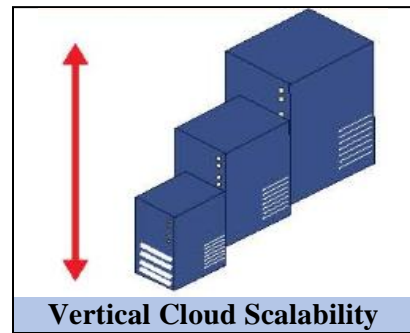


(ii) Vertical Scalability (Scaling Up)

Vertical scaling means increasing the capacity of a single server by adding more resources, such as CPU, RAM, disk space, or network bandwidth. This can improve the performance and reliability of the application, as it can handle more requests and process them faster.

• **Reliability in Cloud Computing**

Reliability in cloud computing relates to the ability of a cloud service or infrastructure to consistently deliver its intended functionality and maintain uptime, often referred to as "high availability." Reliability ensures that applications and services hosted in the cloud are accessible and perform as expected, minimizing downtime and disruptions.



MULTIPLE CHOICE QUESTIONS

- (1) **Which term describes the method bus topology uses to manage data collisions?**
(a) Data buffering (b) TCP/IP
(c) Token passing (d) CSMA/CD
- (2) **In tree topology, the top-level node is commonly referred to as:**
(a) Root node (b) Branch node
(c) Leaf node (d) Hub node
- (3) **Which cloud computing service model provides a platform with tools, services, and frameworks for application development?**
(a) IaaS (b) PaaS
(c) SaaS (d) Hybrid Cloud

SHORT QUESTIONS

- (1) **What do you know about network topology?**
- (2) **What is mesh topology?**
- (3) **Define hybrid topology.**
- (4) **Discuss testing the scalability and reliability of network system.**
- (5) **What do you know about cloud computing?**
- (6) **What is the difference between Load testing and stress testing?.**

1.6 CYBERSECURITY

Cybersecurity is the protection of internet-connected systems such as computers, servers, mobile devices, electronic systems, networks, and data from malicious attacks. It is also known as information technology security or electronic information security.

1.6.1 IMPORTANCE OF CYBERSECURITY

Cybersecurity is vital in any organization, no matter how big or small the organization .

- (i) **Protecting Sensitive Data:** Cybersecurity helps protect sensitive data such as personal information, financial data, and intellectual property from unauthorized access and theft.
- (ii) **Prevention of Cyber Attacks:** Cyber-attacks, such as Malware infections, Ransomware, Phishing, and Distributed Denial of Service (DDoS) attacks, can cause significant disruptions to businesses and individuals.



- (iii) **Safeguarding Critical Infrastructure:** Critical infrastructure, including power grids, transportation systems, healthcare systems, and communication networks, heavily relies on interconnected computer systems.
- (iv) **Maintaining Business Continuity:** Cyber-attacks can cause significant disruption to businesses, resulting in lost revenue, damage to reputation, and in some cases, even shutting down the business.
- (v) **Compliance with Regulations:** Many industries are subject to strict regulations that require organizations to protect sensitive data. Failure to comply with these regulations can result in significant fines and legal action. Cybersecurity helps ensure compliance with these regulations.
- (vi) **Protecting National Security:** Cyber-attacks can be used to compromise national security by targeting critical infrastructure, government systems, and military installations.
- (vii) **Preserving Privacy:** In an era where personal information is increasingly collected, stored, and shared digitally, cybersecurity is crucial for preserving privacy.

1.6.2 CYBERSECURITY THREATS

Cybersecurity threats are harmful acts performed by individuals or groups with destructive intent that aims to gain unauthorized access, damage, disrupt, or steal an information technology asset, computer network, intellectual property, or any other form of sensitive data.

Malware (Malicious Software)

Malware is a broad category of software specifically designed to harm or exploit computer systems, steal data, or gain unauthorized access.

Some common types of malwares include

- (i) **Viruses:** Malicious code that attaches itself to legitimate programs and spreads when the infected program is executed.
- (ii) **Worms:** Self-replicating malware that spreads across networks without user intervention.
- (iii) **Trojans:** Software that appears legitimate but hides malicious functions, such as remote control or data theft.
- (iv) **Ransomware:** Encrypts files or entire systems and demands a ransom for decryption keys.
- (v) **Spyware:** Collects user information without their knowledge, often for advertising or surveillance purposes.

Phishing: *Phishing* is a social engineering attack where cybercriminals impersonate trusted entities to deceive users into revealing sensitive information, such as personal information as login credentials, credit card details,

- (i) **Spear Phishing:** Targeted phishing attacks directed at specific individuals or organizations, often using personalized information.
- (ii) **Email Phishing:** *Cybercriminals* send deceptive emails that appear to be from legitimate sources, encouraging recipients to click on malicious links or download infected attachments.

Denial of Service (DoS) and Distributed Denial of Service (DDoS)

- (i) **DoS:** A single attacker floods the target with traffic, often using multiple devices.
- (ii) **DDoS:** Multiple compromised devices coordinate to flood the target, making it more challenging to mitigate.

Ransomware: *Ransomware* encrypts a victim's data and demands a ransom for the decryption key. Payment does not guarantee data recovery, and victims may lose access to critical information.

Insider Threats: Insider threats involve individuals within an organization (employees, contractors, or business partners) who misuse their access to systems and data for malicious purposes,

Cloud security threats: *Cloud security* threats are potential risks and vulnerabilities that can compromise the security of data, applications, and infrastructure hosted in cloud environments.



1.6.3 PROTECTION AGAINST CYBER-THREATS

Protecting computer systems against cyberattacks is crucial in today's digital age.

Use strong passwords: Strong password security is an important aspect of cybersecurity. A strong password helps protect users' accounts and sensitive information from unauthorized access.

Characteristics of a Strong Password

A strong password typically possesses the following characteristics.

- (i) **Length:** A strong password should be long, usually at least 12 characters. Longer passwords are harder to crack.
- (ii) **Complexity:** It should contain a mix of uppercase and lowercase letters, numbers, and special characters (e.g., @, #, \$, %).
- (iii) **Unpredictability:** Avoid using easily guessable information like names, birthdays, or common phrases.
- (iv) **Uniqueness:** Use different passwords for different accounts. Reusing passwords increases the risk if one account is compromised.



Examples of Strong Passwords

These passwords include: A mix of uppercase letters, lowercase letters, numbers, and special characters, making it complex and unpredictable.

- P@ssword\$Secur31,

Keep your software up to date

"Keeping the software up to date" is a fundamental practice in cybersecurity.



2FA (Two-Factor Authentication)

Authentication is the process of verifying the identity of a user or system trying to access a resource. It ensures that the entity is who it claims to be.

2FA is an authentication method that requires users to provide two different forms of verification before granting access. Typically, it involves something the user knows (password) and something they have (e.g. an OTP (one-time password) from a mobile app).

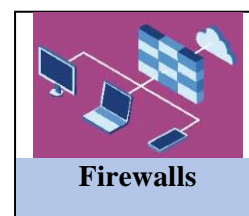


Example: Logging into an online banking account with a password and a unique code, OTP (one-time password) sent to the user's smartphone.

Be wary of suspicious emails: Be cautious of unsolicited emails, particularly those that ask for personal or financial information or contain suspicious links or attachments.

Educate yourself: Stay informed about the latest cybersecurity threats and best practices by reading cybersecurity blogs and attending cybersecurity training programs.

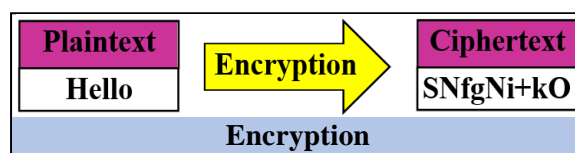
Firewalls: Firewalls are network security devices or software that monitor and control incoming and outgoing network traffic.



Antivirus and Anti-malware Software: Antivirus and anti-malware software

are designed to detect, quarantine, and remove malicious software, such as viruses, Trojans, and spyware, from a system.

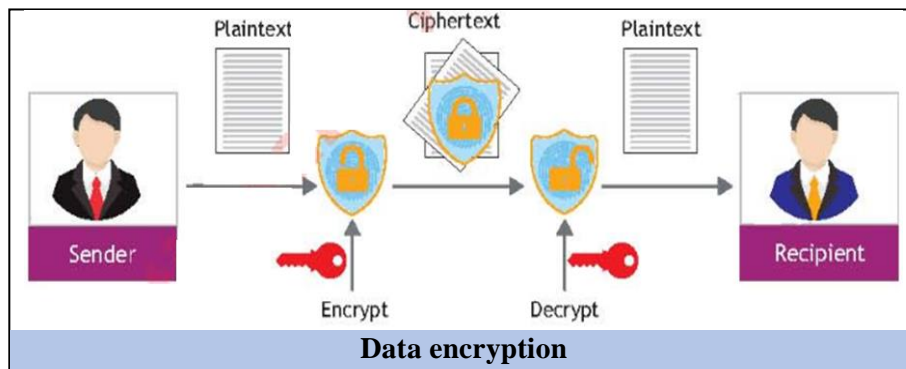
Encryption: Encryption transforms data (plaintext) into a coded format (cipher text) that can only be deciphered with the appropriate decryption key. It ensures the confidentiality and integrity of sensitive data.



Backup and Disaster Recovery: Regular data backups and disaster recovery plans ensure that in case of a cyberattack or data breach, critical data can be restored, minimizing downtime and data loss.

Cryptography: *Cryptography* is a scientific approach of securing information by transforming it into an unreadable format using mathematical algorithms. It ensures data confidentiality, integrity, and authentication.

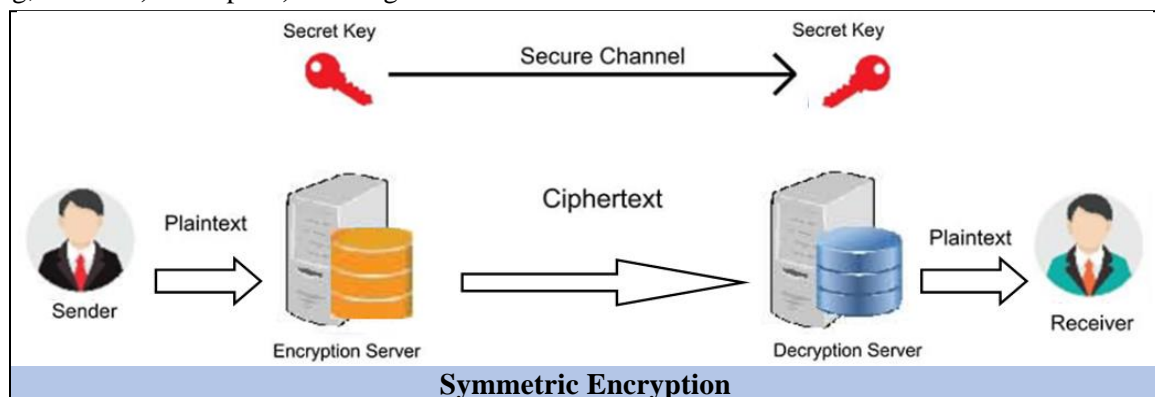
1.6.4 ENCRYPTION: *Data encryption* is a process of converting **plaintext** data into an unreadable format called **cipher text** using encryption algorithms and keys. The primary purpose of data encryption is to protect sensitive information from unauthorized access.



HOW DATA ENCRYPTION WORKS

- (i) **Plaintext:** This is the original, human-readable form of the data that you want to protect. It can be any type of digital information, such as text, files, or communication messages.
- (ii) **Encryption Algorithm:** An encryption algorithm is a mathematical formula or process that transforms plaintext into cipher text.
- (iii) **Encryption Key**
 - A key is a piece of information used by the encryption algorithm to control the transformation of plaintext into cipher text and vice versa. The key can be a numeric value, a passphrase, or a combination of characters. The security of the encryption depends on the strength of the key.
 - **Cryptography and Encryption:** Cryptography is the science of concealing messages with a secret code. Encryption is the way to encrypt and decrypt data. **There are two types of encryption algorithms**
 - (i) Symmetric
 - (ii) Asymmetric

Symmetric Encryption: *Symmetric encryption* is a fundamental data protection technique, relying on a single cryptographic key for both encrypting plaintext and decrypting cipher text. Symmetric encryption is one of the most widely used encryption techniques. It is used in many major industries, including, Défense, Aerospace, Banking & Health care.



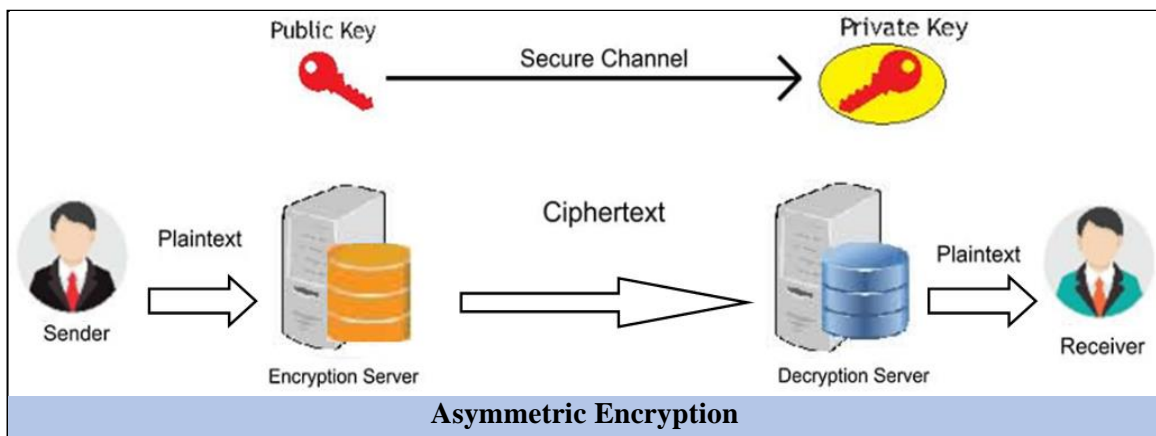
Advantages of Symmetric Encryption

- (i) **Speed and Efficiency:** Symmetric encryption is generally faster and more computationally efficient than asymmetric encryption.
- (ii) **Strong Security:** When used with strong, randomly generated keys, symmetric encryption provides a high level of security.
- (iii) **Simplicity:** Symmetric encryption is simpler to implement and faster to execute because it involves a single key for both encryption and decryption.

Disadvantages of Symmetric Encryption

- (i) **Key Distribution:** One of the major challenges with symmetric encryption is securely distributing and managing the secret keys.
- (ii) **Limited Use for Secure Communication:** Symmetric encryption isn't suitable for secure communication between parties who have never met before or don't share a pre-established key.

Asymmetric Encryption: *Asymmetric encryption* uses two separate keys: a public key and a private key. Often a public key is used to encrypt the data while a private key is required to decrypt the data. The private key is only given to users with authorized access.



Advantages of Asymmetric Encryption

- (i) **Key Distribution:** Each user or entity possesses a unique pair of keys, public and private, where public keys can be openly shared.
- (ii) **Non-refutation:** Asymmetric encryption provides non-refutation, which means the sender of a message cannot deny sending it because only their private key can decrypt the message.
- (iii) **Secure Communication with Untrusted Parties:** Asymmetric encryption is ideal for secure communication between parties who have never met before or don't share a secret key.

Disadvantages of Asymmetric Encryption

- (i) **Computational Complexity:** Asymmetric encryption algorithms are computationally intensive compared to symmetric encryption, which can lead to slower performance, especially when encrypting or decrypting large volumes of data.
- (ii) **Key Length:** Longer key lengths are needed to achieve the same level of security as shorter symmetric keys, which can increase the size of data and messages.

CONTRAST BETWEEN SYMMETRIC AND ASYMMETRIC DATA TRANSMISSIONS

SYMMETRIC ENCRYPTION	ASYMMETRIC ENCRYPTION
Usage	
It uses a single shared key for both encryption and decryption.	It uses a pair of public and private keys for encryption and decryption.
Key	
In this, the same key is used by both the sender and the receiver.	In this, public key is used for encryption, and private key is used for decryption.

Amounts of data	
It is well-suited for encrypting large amounts of data.	It is generally slower, and suitable for smaller amounts of data or secure key exchange.
Channel	
It requires a safe channel to transmit the secret key for key distribution.	It enables secure key exchange over insecure channels without requiring a pre- established secure channel.
Complexity	
It is computationally less complex compared to asymmetric encryption.	It Involves more complicated mathematical operations, making it computationally more complex.
Performance	
It provides better performance in terms of speed and efficiency.	It is slower due to the complexity of mathematical operations involved.
Scenarios	
It is suitable for scenarios where a secure key distribution channel is established.	It is more suited for scenarios where secure key exchange over insecure channels is required.
Applications	
It is often used in situations where performance and efficiency are critical.	It is commonly used for secure data transmission, digital signatures, and securing communication channels.

MULTIPLE CHOICE QUESTIONS

- (1) Which type of malware replicates itself across networks without user intervention?
 - (a) Virus
 - (b) Worm
 - (c) Trojan
 - (d) Ransomware
- (2) Why is keeping software up to date important in cybersecurity?
 - (a) Performance
 - (b) Compatibility
 - (c) Security
 - (d) Cost
- (3) In symmetric encryption, what is used for both encryption and decryption?
 - (a) Public key
 - (b) Private key
 - (c) Shared key
 - (d) Random key

SHORT QUESTIONS

- (1) WHAT IS CYBER SECURITY?
- (2) WHAT IS ENCRYPTION?
- (3) WHAT IS CRYPTOGRAPHY?
- (4) WHAT ARE CYBERSECURITY THREATS?
- (5) HOW DOES FIREWALL PROTECT OUR SYSTEM FROM THREATS?
- (6) HOW EMAIL PHISHING IS MOST ATTACK IN PHISHING?