

UNIT I

FUNDAMENTALS OF NEURAL NETWORKS

1. Introduction:

Basic Concepts of Neural Networks

1.1. What is a Neural Network?

A Neural Network is a computational model inspired by the way biological neural systems, such as the human brain, process information. It is designed to recognize patterns and solve problems by mimicking how neurons in the brain work together.

Neural Networks consist of interconnected layers of nodes, or "neurons," where each node performs a mathematical operation. The structure typically includes:

- **Input Layer:** Receives the data.
- **Hidden Layers:** Process the data through various transformations.
- **Output Layer:** Produces the final result.

1.2. Why Neural Networks?

Neural Networks are powerful tools for:

- Image recognition (e.g., facial recognition in smartphones).
- Speech processing (e.g., virtual assistants like Siri or Alexa).
- Predictive analytics (e.g., weather forecasting).

1.3. Key Terminologies in Neural Networks

- **Neuron:** The basic unit that processes data by applying a function to inputs.
- **Weights:** Parameters that determine the importance of each input.
- **Bias:** Helps the network make decisions by adding flexibility to the activation function.
- **Activation Function:** Decides whether a neuron's output should be passed to the next layer. Examples include sigmoid, ReLU, and tanh.

Real-Life Example: Predicting House Prices

Imagine you're helping your parents decide how much to sell your house for. Several factors, such as location, size, number of bedrooms, and age of the house, influence the price.

1. Input Layer:

- Location
- Size in square feet
- Number of bedrooms
- Age of the house

2. Hidden Layers:

- The neural network processes this information to find patterns. For example, houses in prime locations and larger houses generally have higher prices.

3. Output Layer:

- Predicts the estimated price of the house.

Each factor is assigned a weight. If the house is in a good location, the network will assign a higher weight to the "location" input. Similarly, a smaller weight might be given to "age" if the house's age is less critical in determining its price.

How It Works:

The neural network adjusts the weights and biases during training (using historical data of house prices) until it accurately predicts the price for new data. This process is called **learning** or **training**.

1.4. Biological Analogy

The human brain has billions of neurons. Each neuron receives signals through dendrites, processes them, and sends the result through axons. Similarly:

- Inputs = Signals received by dendrites.
- Weights = Strength of the signal.
- Activation Function = Decides whether to "fire" the signal to the next neuron.

Real-Life Example: Spam Email Detection

When you receive an email, your inbox uses a neural network to decide whether it's spam:

- Inputs: Email features like subject line, keywords, sender's address.
- Hidden Layers: The network identifies patterns like common spam phrases.
- Output: Classifies the email as "Spam" or "Not Spam."

By training the model on millions of examples, it learns to recognize even subtle differences between spam and genuine emails.

2. Human Brain

2.1. Introduction to the Human Brain

The human brain is one of the most complex and powerful organs in the body. It controls all the functions of the body, processes information from the senses, and helps us think, learn, and make decisions.

The brain is made up of approximately **86 billion neurons**, which are specialized cells that transmit information. Neurons communicate with each other using electrical and chemical signals. This communication forms the foundation for everything we do, from moving our fingers to solving complex math problems.

2.2. Structure of the Human Brain

The human brain has three main parts:

1. **Cerebrum:**
 - The largest part of the brain.
 - Responsible for functions like thinking, learning, memory, and voluntary movements.
 - Divided into two hemispheres (left and right), with each controlling the opposite side of the body.
2. **Cerebellum:**
 - Located under the cerebrum.
 - Manages coordination, balance, and motor skills.
 - Helps us perform tasks like walking, running, or riding a bicycle.
3. **Brainstem:**
 - Connects the brain to the spinal cord.
 - Controls basic functions like breathing, heartbeat, and digestion.

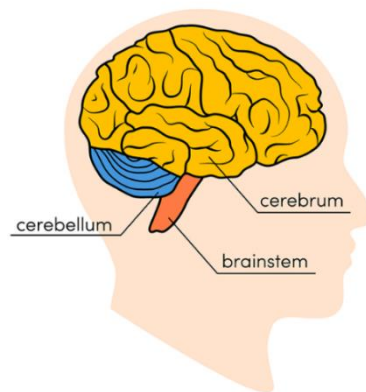


Fig: human brain

2.3. How the Brain Processes Information

The brain receives input from the environment through the **five senses**:

- **Eyes** for vision.
- **Ears** for hearing.
- **Skin** for touch.
- **Nose** for smell.
- **Tongue** for taste.

For example:

- When you see a red apple, your eyes send signals to your brain. The brain processes this visual information, identifies the apple, and even associates it with its taste or previous experiences.

2.4. Neurons: The Brain's Building Blocks

Neurons are the brain's fundamental units. They work in three stages:

1. **Input:** Neurons receive signals from the environment or other neurons.
2. **Processing:** The brain processes this information to make sense of it.
3. **Output:** The brain sends commands to the body based on the processed information.

Real-Life Example: Driving a Car

- **Input:** Your eyes see a red traffic light.
- **Processing:** Your brain recognizes the red light as a signal to stop.
- **Output:** Your brain sends signals to your foot to press the brake pedal.

2.5. Connection to Artificial Neural Networks

Artificial neural networks (ANNs) are inspired by the human brain. Similar to neurons in the brain, ANNs consist of artificial "neurons" that are connected in layers. They take input, process it, and produce output.

Real-Life Example of ANN:

- Imagine a voice assistant like Alexa.
 - **Input:** You say, "What's the weather today?"
 - **Processing:** The system processes your voice, understands the query, and checks weather data.
 - **Output:** It replies with, "Today's weather is sunny."

2.6. Why is the Human Brain Important in AI?

Scientists and engineers study the brain to build systems that mimic its capabilities. For example:

- How the brain learns helps improve **machine learning algorithms**.
- How neurons connect and communicate inspires the design of **deep learning models**.

Example: Face Recognition

Just like your brain recognizes your friend in a crowd, neural networks can identify faces using cameras and AI. The system learns patterns (like the shape of eyes and nose) to make accurate identifications.

Key Takeaways

- The human brain is the inspiration behind artificial neural networks.
 - Neurons in the brain communicate using electrical and chemical signals, similar to how artificial neurons process data.
 - Real-life tasks, like recognizing faces or stopping at a red light, show how the brain processes input, makes decisions, and gives outputs.
-

3. Model of Artificial Neurons

Artificial neurons are the building blocks of artificial neural networks, inspired by the biological neurons found in the human brain. To understand the model of an artificial neuron, let's break it down step by step and look at real-life examples to make it easier to grasp.

3.1. Biological Inspiration

- In the human brain, neurons are specialized cells that transmit information. Each neuron receives signals (inputs) through its dendrites, processes the information in the cell body, and then sends an output signal through its axon.
- Artificial neurons mimic this behaviour mathematically. Instead of dendrites, we have inputs; instead of processing in the cell body, we use mathematical functions; and instead of axons, we produce an output.

3.2. Components of an Artificial Neuron

An artificial neuron has the following components:

a. Inputs (Δ)

Inputs are the values or data fed into the neuron. These could be numbers that represent real-world information.

- **Example:** In predicting house prices, the inputs could be factors like the number of bedrooms, size of the house, and location.

b. Weights (ω)

Each input is multiplied by a weight. A weight determines how important an input is to the overall output of the neuron.

- **Example:** In predicting house prices, location might have a higher weight than the number of bedrooms because location significantly affects the price.

c. Summation Function

All the weighted inputs are added together to produce a single value. Mathematically, this is written as:

where:

- is the input value
- is the corresponding weight
- is the summation value

d. Activation Function

The summation value is passed through an activation function, which determines whether the neuron should "fire" (produce an output) or not. Common activation functions include:

- **Sigmoid Function:** Produces outputs between 0 and 1.
- **ReLU (Rectified Linear Unit):** Outputs 0 for negative values and the input itself for positive values.

e. Output

The result of the activation function is the output of the neuron.

- **Example:** In a spam email classifier, the output might be 1 (spam) or 0 (not spam).

3.3. Real-Life Example: Self-Driving Cars

Let's see how an artificial neuron works in a real-life scenario:

Problem:

A self-driving car needs to decide whether to stop at a traffic signal. Inputs to the artificial neuron might include:

1. **Distance to the traffic light** (in meters)
2. **Traffic light colour** (represented as numbers: Red = 1, Yellow = 0.5, Green = 0)
3. **Speed of the car** (in km/h)

Steps:

1. **Assign Inputs:**
 - Distance: 20 meters
 - Traffic light colour: 1 (Red)
 - Speed: 50 km/h
2. **Set Weights:**
 - Distance weight: 0.3
 - Traffic light colour weight: 0.6

- Speed weight: 0.1
 - 3. **Calculate Weighted Sum:**
 - 4. **Apply Activation Function:** Using a threshold activation function, if , the neuron outputs 1 (stop), otherwise 0 (don't stop).
 - 5. **Output:** Since , the output is 1, meaning the car should stop.
-

4. McCulloch-Pitts Model

The **McCulloch-Pitts model** is one of the first artificial neural network models. It was introduced in 1943 by Warren McCulloch and Walter Pitts. This model mimics the behaviour of a biological neuron in the brain.

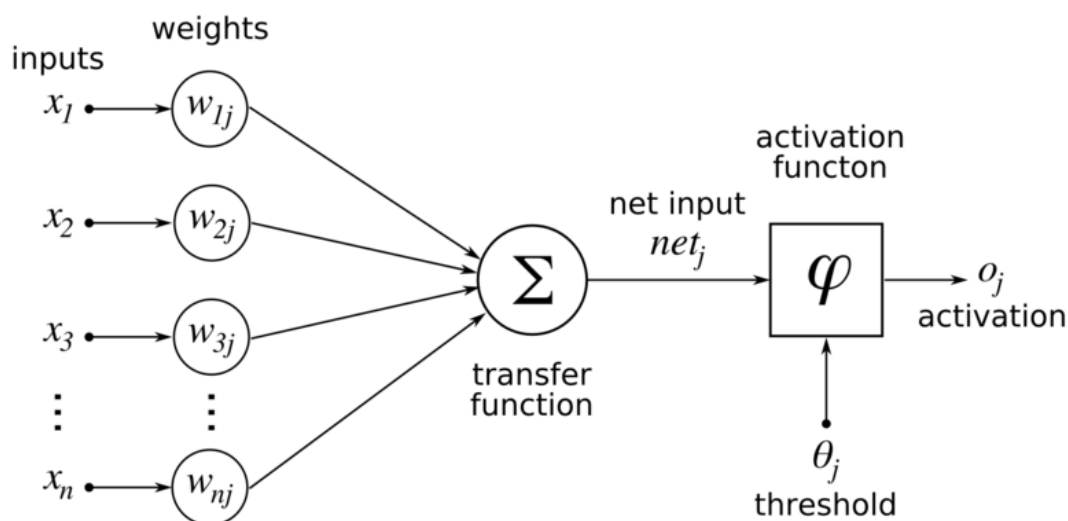
4.1. What is a Neuron?

- A **neuron** is a nerve cell in the brain that processes information and sends signals to other neurons.
- Just like our brain has billions of neurons to process thoughts, computers can use artificial neurons to solve problems.

4.2. What is the McCulloch-Pitts Neuron?

The McCulloch-Pitts neuron is a simple mathematical model of how a biological neuron works. It works in three steps:

1. **Input Signals:** It receives signals (inputs) from the outside world or other neurons.
2. **Summation:** It adds up all the input signals after multiplying them with their respective weights (importance).
3. **Threshold Function:** If the total is greater than or equal to a threshold value, it produces an output signal (like ON or OFF).



4.3. Key Components of McCulloch-Pitts Neuron

1. **Inputs (x):** These are the signals received by the neuron.
2. **Weights (w):** These are values assigned to each input to indicate its importance.
3. **Summation (Σ):** The weighted sum of all inputs.
4. **Threshold (θ):** A value that determines when the neuron "fires" or produces an output.
5. **Output (y):** The final decision (e.g., 1 for ON or 0 for OFF).

4.4. How Does It Work?

The McCulloch-Pitts neuron works as follows:

1. Multiply each input by its corresponding weight.
2. Add all the weighted inputs.
3. Compare the sum to a threshold value.
 - If the sum is greater than or equal to the threshold, the neuron outputs 1 (ON).
 - Otherwise, it outputs 0 (OFF).

4.5. Real-Life Example: Turning on a Light Bulb

Imagine you are designing a simple system to turn on a light bulb based on three conditions:

1. **Condition 1:** It's dark outside.
2. **Condition 2:** Someone is in the room.
3. **Condition 3:** The switch is turned ON.

Each condition is an **input**:

- If a condition is true, the input is 1.
- If it's false, the input is 0.

Let's assign weights to each condition:

- Condition 1 (dark): Weight = 2
- Condition 2 (presence): Weight = 3
- Condition 3 (switch): Weight = 5

Set a threshold, $\theta = 6$.

Step-by-Step Calculation

1. Check the inputs:
 - Dark (1), Someone is in the room (1), and Switch is ON (1).
 - Inputs = [1, 1, 1].
2. Multiply each input by its weight:
 - $(\text{Dark} \times 2) + (\text{Presence} \times 3) + (\text{Switch} \times 5) = 2 + 3 + 5 = 10$.
3. Compare the sum to the threshold:
 - $10 \geq 6$ (threshold).
4. Output:
 - The neuron "fires" (output = 1), turning on the light bulb.

4.6. Advantages of McCulloch-Pitts Model

1. **Simple Design:** Easy to understand and implement.
2. **Binary Decision:** Makes clear ON/OFF decisions like a switch.
3. **Foundation:** It laid the groundwork for modern artificial neural networks.

4.7. Limitations of McCulloch-Pitts Model

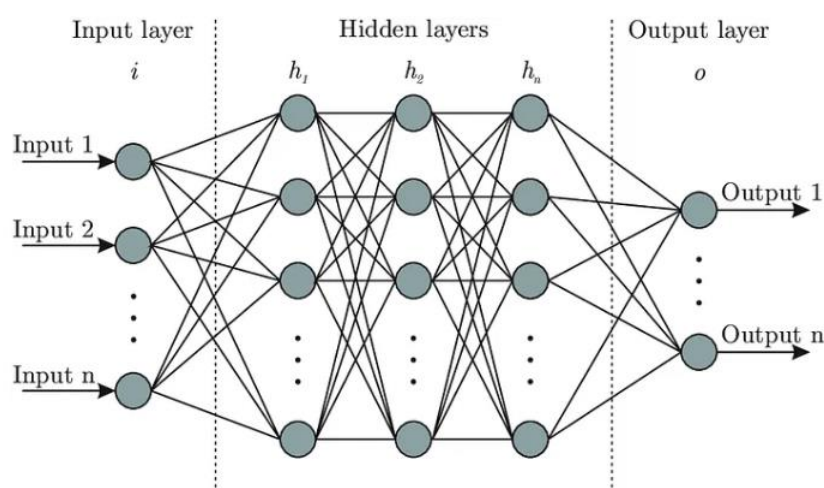
1. **Limited Learning:** It doesn't learn from experience or data.
 2. **Static Weights:** The weights and threshold are fixed and must be manually assigned.
 3. **No Complex Outputs:** It can only give binary outputs (1 or 0)
-

5. Neural Network Architecture

A neural network is a computational model inspired by the structure and functioning of the human brain. It consists of layers of interconnected nodes (neurons) designed to process information.

Key Components:

- **Input Layer:** Receives the input data.
- **Hidden Layers:** Intermediate layers where computations are performed.
- **Output Layer:** Produces the final result.
- **Weights and Biases:** Connections between neurons have weights that determine the strength of the signal, while biases adjust the output.



Example: Imagine a house pricing system. The input layer might take the size of the house, number of rooms, and location. Hidden layers process this data using mathematical operations, and the output layer predicts the price.

5.1. Characteristics of Neural Networks

Neural networks have the following key features:

1. **Non-linearity:** Neural networks can model non-linear relationships between inputs and outputs.
 - *Example:* Predicting weather based on humidity and temperature, which do not have a linear relationship.
2. **Adaptivity:** They learn from data and improve over time.
 - *Example:* A recommendation system like Netflix improving its suggestions as you watch more shows.
3. **Fault Tolerance:** Even if some neurons fail, the network can still function.
 - *Example:* A speech recognition system still works even with noisy input.
4. **Generalization:** Neural networks can generalize well to unseen data if trained properly.
 - *Example:* A handwriting recognition model accurately identifying new handwriting styles.

5.2. Learning Methods

Neural networks learn by adjusting weights and biases based on the data provided. The main learning methods are:

1. **Supervised Learning:** The model learns using labelled data, where inputs have corresponding outputs.
 - *Example:* Teaching a system to classify images of cats and dogs using a dataset with labelled images.
2. **Unsupervised Learning:** The model identifies patterns in unlabelled data.
 - *Example:* Grouping customers based on purchasing behaviour without predefined categories.
3. **Reinforcement Learning:** The model learns by interacting with its environment and receiving rewards or penalties.
 - *Example:* A self-driving car learning to navigate roads by trial and error.

5.3. Taxonomy of Neural Network Architecture

Neural networks can be classified into different types based on their structure and application:

1. **Feedforward Neural Network:** Data flows in one direction from input to output.
 - *Example:* Predicting house prices.
2. **Recurrent Neural Network (RNN):** Includes loops to process sequential data.
 - *Example:* Language translation or speech recognition.

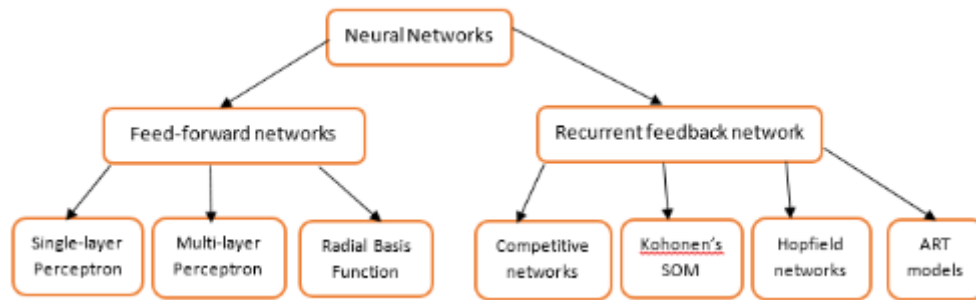


Fig: Neural Network Architecture

6. Early Neural Network Architectures

Early neural networks laid the foundation for modern advancements. Here are some notable ones:

6.1. Perceptron (1958)

- **Inventor:** Frank Rosenblatt
- **Architecture:** The Perceptron is one of the earliest models of a neural network and is a type of single-layer feed forward neural network. It consists of an input layer, a processing node (called the perceptron), and an output layer.
- **Function:** The perceptron performs binary classification using a linear decision boundary, where the input is multiplied by weights, summed, and passed through an activation function (usually a step function).
- **Limitations:** It can only solve linearly separable problems, like AND or OR logic gates, and fails for non-linearly separable problems (e.g., XOR).
- **Impact:** It introduced the basic structure of a neural network but faced criticism after its inability to solve XOR problems was highlighted by Marvin Minsky and Seymour Papert in their book *Perceptron's* (1969).

Example: Classifying emails as spam or not spam.

6.2. ADALINE (Adaptive Linear Neuron, 1960):

- **Inventor:** Bernard Widrow and Ted Hoff
- **Architecture:** Similar to the perceptron but with a key difference in the activation function used.
- **Function:** Instead of the step function, Adaline uses a linear activation function, and it updates weights using the Least Mean Squares (LMS) rule, minimizing the error between the predicted and actual values.

Example: Predicting stock market based on previous analysis.

6.3. Hopfield Network (1982)

- **Inventor:** John Hopfield
- **Architecture:** A recurrent neural network (RNN) with fully connected neurons that operate in an auto-associative manner.
- **Function:** It operates as a content-addressable memory system, where patterns can be stored and later retrieved from partial or noisy input.
- **Properties:** Symmetric connections between neurons, binary outputs, and the network converges to a stable state.
- **Applications:** Memory retrieval, pattern recognition, and optimization problems.
 - *Example:* Recalling a specific image from noisy data.

4. Multi-Layer Perceptron (MLP) (1980s)

- **Inventors:** Various researchers (notably including Paul Werbos and Geoffrey Hinton)
 - **Architecture:** A more advanced version of the perceptron, consisting of multiple layers of neurons (input layer, hidden layers, and output layer).
 - **Function:** Uses backpropagation, a key algorithm that allows MLP to train more complex, non-linear models by adjusting the weights based on the gradient of the error.
 - **Activation Functions:** Sigmoid, tanh, and later ReLU and Softmax.
 - **Impact:** The introduction of multi-layer networks with backpropagation made it possible to train neural networks on complex tasks, such as image and speech recognition.
 - *Example:* Detecting fraudulent transactions.
-

7. Perceptron Model

The perceptron is one of the simplest types of artificial neural networks. It was introduced by Frank Rosenblatt in 1958 and forms the foundation for more complex neural network models.

7.1. What is a Perceptron?

A perceptron is a computational model inspired by the way biological neurons work. It takes multiple input values, applies weights to them, sums them up, and then passes the result through an activation function to produce an output.

The perceptron can only solve problems that are linearly separable, meaning it can classify data points that can be separated by a straight line.

7.2. Structure of a Perceptron

1. **Inputs (Φ):** These are the features or data points fed into the perceptron. For example, in determining whether an email is spam or not, inputs could be the presence of certain words.
2. **Weights (ω):** Each input is multiplied by a corresponding weight that represents the input's importance.
3. **Bias (b):** A constant value added to adjust the output, enabling the model to fit the data better.
4. **Summation (Σ):** The weighted inputs and bias are summed up.
5. **Activation Function:** This determines whether the neuron "fires" or not (e.g., outputs 0 or 1).

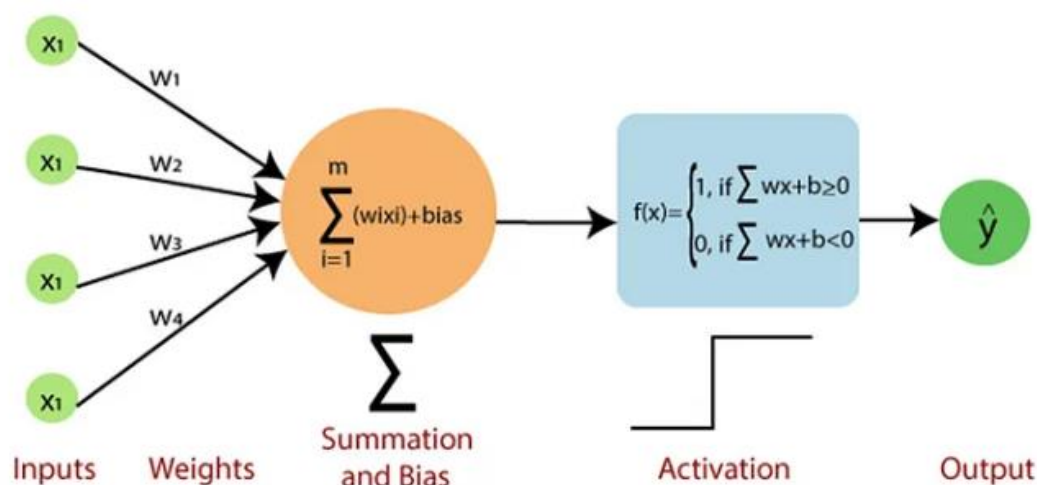


Fig: Structure of a Perceptron

7.3. Real-Life Example: Perceptron Model

Imagine you are creating a system to predict if a fruit is an apple (1) or not an apple (0). The inputs could be:

- : Is the fruit red? (Yes = 1, No = 0)
- : Is the fruit round? (Yes = 1, No = 0)

Weights are assigned based on how important each feature is. For example:

- : Being red is very important.
- : Being round is slightly less important.

If the weighted sum of these inputs plus a bias exceeds a threshold, the perceptron outputs 1 (apple); otherwise, it outputs 0 (not an apple).

8. Perceptron Learning Algorithm

The perceptron learning algorithm is used to train the perceptron by adjusting its weights and bias to minimize classification errors.

Steps of the Algorithm:

1. **Initialize weights and bias:** Start with small random values.
2. **For each training sample:**
 - Compute the perceptron's output .
 - Compare with the actual label (target output).
 - Update weights and bias if there is an error using:

Where:

- : Learning rate (a small positive number to control the speed of learning)
 - : Target output (1 for correct classification, 0 otherwise)
 - : Predicted output
3. **Repeat:** Iterate over the dataset until all samples are correctly classified or a maximum number of iterations is reached.

Real-Life Example: Email Spam Filter

Suppose you are designing a perceptron to classify emails as spam (1) or not spam (0). The inputs could include:

- : Does the email contain the word "free"? (Yes = 1, No = 0)
 - : Is the email sent from an unknown sender? (Yes = 1, No = 0)
1. Initialize weights (e.g.,) and bias (e.g.,).
 2. For an email with inputs and labelled as spam (), calculate the output:

If , update weights and bias using the learning rule. For example, if :

Repeat this process for all emails in the dataset until the perceptron classifies them correctly.

9. Error Correction

Definition:

Error correction is a process used in neural networks to minimize the difference between the predicted output of the model and the actual desired output. This difference is called the "error." The network learns by adjusting its parameters (weights and biases) to reduce this error.

Key Concept:

1. **Error Calculation:** The error is calculated as the difference between the actual output (target value) and the predicted output by the network.

$$\text{Error} = \text{Target Output} - \text{Predicted Output}$$

2. **Learning Process:** The network updates the weights and biases in such a way that the error is minimized. This process is iterative and continues until the error reaches an acceptable level.

Real-Life Example:

Spam Email Filtering:

- Suppose you create a neural network to classify emails as "Spam" or "Not Spam."
- Initially, the model might incorrectly classify some spam emails as "Not Spam." This is an error.
- The error correction mechanism adjusts the model's weights to better identify patterns in spam emails (like specific words or phrases). Over time, the network becomes better at distinguishing spam emails from non-spam ones.

For instance:

- Target Output: Spam (1)
 - Predicted Output: Not Spam (0.4)
 - Error: $1 - 0.4 = 0.6$
 - The network will reduce this error by adjusting its weights using a learning algorithm.
-

10. Gradient Descent Rules

Definition:

Gradient descent is an optimization algorithm used to minimize the error (or loss) in a neural network by adjusting the weights and biases. The goal is to find the optimal values for the parameters that result in the lowest error.

Key Concept:

1. **Gradient:** The gradient is the derivative of the error with respect to the weights. It indicates the direction and rate of change of the error.
2. **Descent:** The weights are updated in the direction opposite to the gradient to minimize the error.

The weight update rule is given by:

$$w_{\text{new}} = w_{\text{old}} - \eta \cdot \frac{\partial E}{\partial w}$$

Where:

- w_{new} : Updated weight
- w_{old} : Current weight
- η : Learning rate (a small constant determining the step size)
- $\frac{\partial E}{\partial w}$: Gradient of the error with respect to the weight

Real-Life Example:

Self-Driving Cars:

- A neural network in a self-driving car learns to predict steering angles based on road images.
- Initially, the car might predict a wrong angle, leading to deviations from the lane. This deviation represents the error.
- Gradient descent updates the weights in the neural network to improve the predictions. Over time, the car learns to predict accurate steering angles, reducing lane deviations.

For instance:

- If the car steers too much to the right, the error gradient will point toward reducing the "right-steering" weights.
- The model adjusts its weights slightly in each iteration, moving closer to the optimal configuration for accurate driving.

UNIT II

BACKPROPAGATION NETWORKS

1. Introduction:

Backpropagation (short for "Backward Propagation of Errors") is a learning algorithm used in artificial neural networks (ANNs) to improve their accuracy. It helps the network learn from mistakes by adjusting its weights and biases to minimize errors.

Think of it like this:

- Imagine you are learning to throw a basketball into a hoop.
- The first few times, you might miss.
- But each time you adjust your throw—maybe you aim higher or use less force.
- Eventually, you get better at making the shot.
- Backpropagation helps a neural network learn in a similar way by adjusting its internal parameters to improve accuracy.

1.1. Why is Backpropagation Important?

Before backpropagation, training neural networks was slow and inefficient. With backpropagation:

- Neural networks can **learn faster** and improve their predictions.
- Errors can be **corrected automatically** in a structured way.
- It allows deep learning models (like those used in **speech recognition, self-driving cars, and image recognition**) to function efficiently.

1.2. How Does Backpropagation Work?

Backpropagation is based on two main steps:

1. Forward Propagation:

- Input data passes through the network, layer by layer, until it reaches the output layer.
- The network makes a prediction.
- Example: A neural network for digit recognition might see a handwritten "5" and predict it's a "3" instead.

2. Backward Propagation (Error Correction):

- The network calculates the error by comparing the predicted output with the correct output.
- It then adjusts the weights of the connections to reduce the error using a method called **Gradient Descent** (which helps find the optimal values).
- This process repeats until the network learns to make accurate predictions.

1.3. Real-Life Example of Backpropagation

Example 1: Learning to Ride a Bicycle

- At first, when you try to ride a bicycle, you might fall or wobble.
- Your brain **analyses the mistake** (error) and makes adjustments (like shifting balance, adjusting pedalling speed).
- Over time, with practice and corrections, you improve.
- This is similar to how backpropagation works—it **helps a neural network adjust and improve based on errors**.

Example 2: Virtual Assistants (Siri, Alexa, Google Assistant)

- When you talk to a virtual assistant, it sometimes misunderstands your words.
 - If you correct it (for example, by choosing the right option or repeating yourself), it **learns from that mistake**.
 - The next time you say something similar, it gets better at understanding.
 - Backpropagation helps these systems **improve speech recognition accuracy** over time.
-

2. Supervised Learning with Neural Networks

2.1. What is Supervised Learning?

Supervised learning is a type of machine learning where a model learns from labelled data. This means that for every input, there is a correct output already given, and the model learns to map inputs to outputs by finding patterns in the data.

- **"Supervised" means "guided or taught"**—just like how a teacher corrects students' mistakes in class.
- The computer is **trained with examples**, and it learns from the mistakes using feedback.

2.2. How Does Supervised Learning Work?

Supervised learning works in three main steps:

1. **Training Phase:**
 - The model is given a dataset with **inputs (features)** and **correct outputs (labels)**.
 - The neural network makes predictions and compares them with the actual outputs.
 - If the prediction is wrong, the network **adjusts its weights** to improve accuracy.
2. **Testing Phase:**
 - Once trained, the model is tested on new, unseen data to check how well it has learned.
3. **Prediction Phase:**
 - After training & testing, the model can make predictions on new data.

2.3. Real-Life Example: Learning to Recognize Handwritten Digits

Imagine you are training a neural network to recognize handwritten numbers (like a digit recognition system used by banks for reading cheques).

1. **Step 1: Training the Neural Network**
 - We show the model **thousands of images of handwritten numbers (0-9)**.
 - Each image has a **correct label** (e.g., a picture of '7' is labelled as '7').
 - The neural network **analyses patterns in the images** and learns to associate them with numbers.
2. **Step 2: Testing the Model**
 - Once trained, we test the model with new handwritten numbers it has never seen before.
 - If the model correctly identifies most numbers, it means it has **learned well**.
3. **Step 3: Real-World Use**
 - When you deposit a cheque at a bank, the ATM scans the handwritten amount and recognizes the digits correctly using a trained neural network.

2.4. How Neural Networks Learn in Supervised Learning?

Neural networks learn using an algorithm called **Backpropagation (Backward Propagation of Errors)**.

Key Steps in Learning:

1. **Forward Pass:**
 - The input goes through the neural network and produces an output (prediction).
2. **Error Calculation:**
 - The model checks how far its prediction is from the correct answer (error calculation).
3. **Backpropagation (Backward Pass):**
 - The error is sent **backward** through the network to adjust the weights.
 - This ensures that the network learns from its mistakes and improves.
4. **Weight Update using Gradient Descent:**
 - The network updates its weights using an optimization technique called **Gradient Descent** to minimize the error.
5. **Repeat the Process:**
 - Steps 1-4 are repeated **many times** until the model becomes very accurate.

2.5. Real-Life Example: Learning to Solve Math Problems

Think about how you learn to solve math problems:

1. **Step 1 (Training):**
 - Your teacher gives you practice problems with answers.
 - You try solving them and check if you got them right.

2. **Step 2 (Error Checking):**
 - If you made a mistake, your teacher points it out.
 - You understand what went wrong.
3. **Step 3 (Correction & Learning):**
 - You try solving similar problems again.
 - Over time, you **learn** how to get the correct answers without mistakes.

This is exactly how supervised learning works in neural networks!

2.6. Types of Supervised Learning Problems

There are two main types of supervised learning problems:

1. Classification (Predicting a category)

- The output is a category (e.g., Yes/No, Spam/Not Spam, Dog/Cat).
- Example: Gmail classifies emails as **Spam or Not Spam** based on previous emails labelled by users.

2. Regression (Predicting a number)

- The output is a continuous value (e.g., price, temperature, age).
 - Example: A house price prediction system predicts the **price of a house** based on factors like size, location, and number of bedrooms.
-

3. Architecture of Backpropagation Network

A backpropagation network consists of three main layers:

1. **Input Layer**
2. **Hidden Layer(s)**
3. **Output Layer**

Each layer is made up of **neurons**, which are the fundamental processing units of the network.

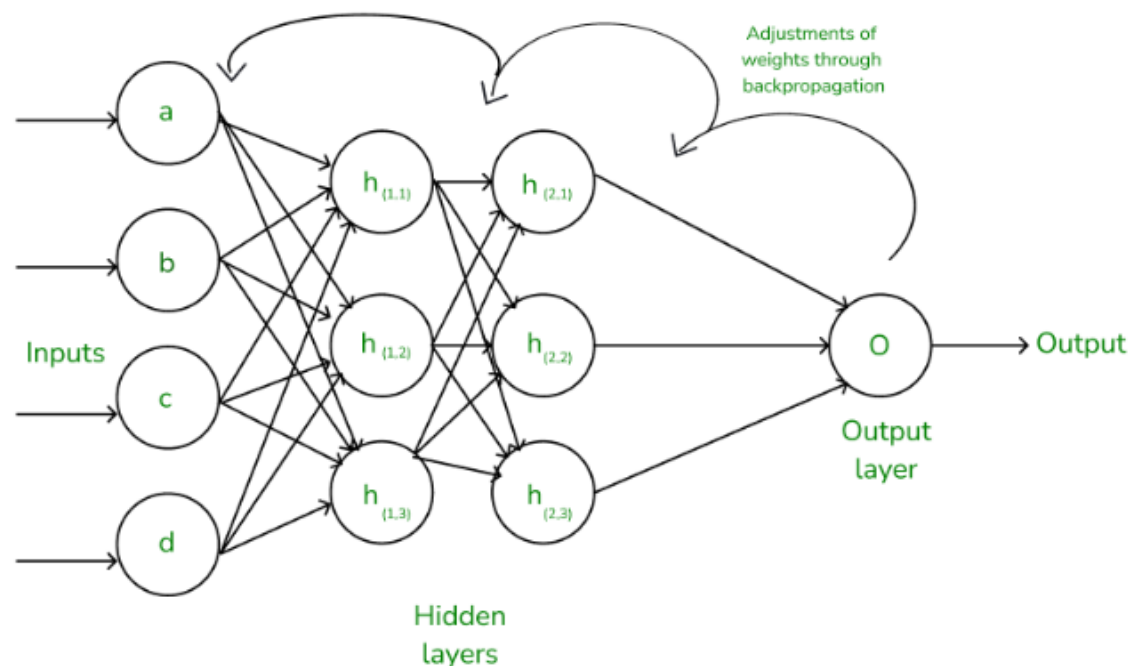


Fig: Architecture of Backpropagation Network

(i) Input Layer

- This is the first layer of the network.
- It receives raw data (features) from the external environment.
- Each input neuron represents one feature of the data.

❖ Real-Life Example:

Think of an **image recognition system** that classifies images as either a "Dog" or "Cat".

- The input layer takes pixel values of an image as input.
- If the image has 100 pixels, the input layer will have 100 neurons (one for each pixel).

(ii) Hidden Layer(s)

- These layers process the input data by performing mathematical operations.
- The more hidden layers a network has, the **deeper** it is (Deep Learning).
- Each neuron in the hidden layer applies an **activation function** to decide whether to pass the information forward.

❖ Real-Life Example:

Continuing with the **image recognition** example:

- The hidden layers extract patterns from the image, such as edges, colours, and shapes.
- The deeper the network, the better it understands complex features (like eyes, nose, ears of a cat or dog).

(iii) Output Layer

- This is the final layer that gives the result.
- The number of neurons in the output layer depends on the **type of problem**:
 - **Binary classification (Yes/No, Dog/Cat)** → 1 neuron
 - **Multi-class classification (Car, Bike, Truck, Bus)** → More than 1 neuron

❖ Real-Life Example:

For the **dog vs. cat classifier**:

- If the output is **1**, it means the image is a **dog**.
- If the output is **0**, it means the image is a **cat**.

3.1. Working of Backpropagation Algorithm (Step-by-Step)

The backpropagation network works in two phases:

1. **Forward Propagation** → The input data passes through the network, and the output is calculated.
2. **Backward Propagation (Error Correction)** → The network adjusts its weights to reduce errors.

Step 1: Forward Propagation

- The input data is passed through each layer.
- The neurons in each layer apply weights and activation functions.
- The final output is generated.

❖ Example:

If you give an image of a **dog**, the network will predict whether it is a dog or not.

Step 2: Compute Error

- The network compares the predicted output with the actual output.
- The difference between them is called the **error (loss)**.

❖ Example:

If the network predicted "Cat" but the actual image was a "Dog", then an **error** is generated.

Step 3: Backward Propagation (Error Correction)

- The error is propagated backward to adjust the weights.
- The **Gradient Descent Algorithm** is used to minimize the error step by step.

✓ Example:

Imagine a **teacher correcting a student's mistakes**:

- If a student writes **wrong answers** in an exam, the teacher marks them incorrect.
- The student **learns from mistakes** and corrects them in the next test.
- Similarly, the network **updates its weights** and improves its accuracy.

3.2. Key Features of Backpropagation Network

- ✓ **Supervised Learning** – It learns from labelled data (i.e., input data with correct answers).
- ✓ **High Accuracy** – Used in deep learning to solve complex problems.
- ✓ **Error Minimization** – Continuously reduces error with each training step.

3.3. Real-Life Applications of Backpropagation Networks

1. **Face Recognition (Unlocking Phones with Face ID)**
 - The network learns to recognize facial features.
 - It compares the detected face with stored images and grants access if it matches.
 2. **Self-Driving Cars**
 - Neural networks process sensor data (camera, radar, LiDAR).
 - It learns to recognize road signs, detect obstacles, and drive safely.
 3. **Chatbots & Virtual Assistants (Siri, Alexa, Google Assistant)**
 - Uses backpropagation to learn from past conversations.
 - Improves response accuracy over time.
-

4. The Perceptron Model

4.1. Introduction to Perceptron

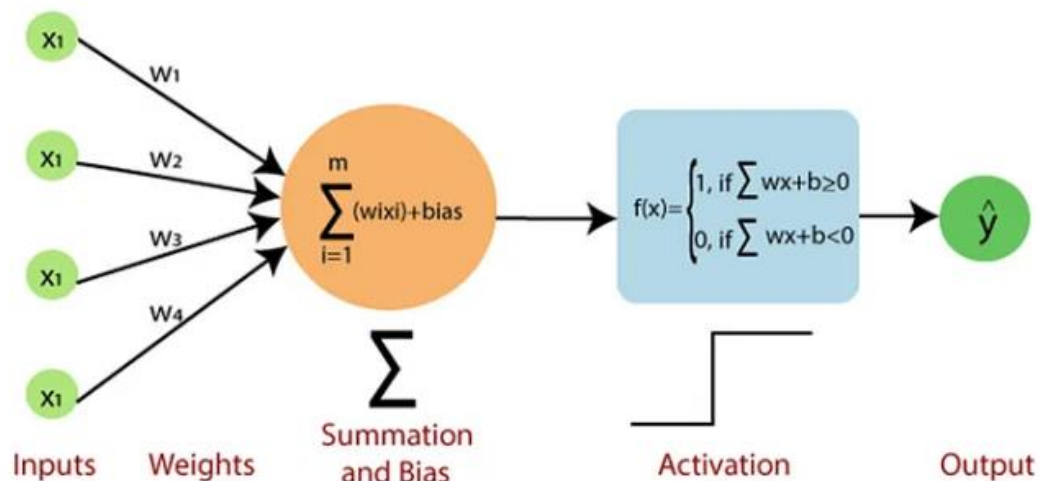
A **perceptron** is the simplest type of artificial neural network. It is a type of **supervised learning model** used for classification tasks. A perceptron works like a decision-making system that can determine whether something belongs to one category or another (for example, "spam" or "not spam" in email classification).

The perceptron was invented by **Frank Rosenblatt** in 1958 and is the foundation of modern neural networks.

4.2. Structure of a Perceptron

A perceptron is made up of the following components:

1. **Input Layer:** This consists of multiple input values (features) that provide information to the perceptron.
2. **Weights (W):** Each input is assigned a weight that determines its importance.
3. **Summation Function (Σ):** The weighted sum of all inputs is calculated.
4. **Activation Function:** The weighted sum is passed through an activation function that decides whether the neuron should "fire" (output a value) or not.
5. **Output:** The final result, which is either **0 (inactive)** or **1 (active)** based on the activation function.



Mathematical Representation:

$$y = f(\sum (w_i \cdot x_i) + b)$$

Where:

- x_i = Inputs
- w_i = Weights

- b = Bias (adjusts the decision boundary)
- f = Activation function
- y = Output

4.3. Working of a Perceptron

A perceptron makes decisions using a **step function** as an activation function.

- If the weighted sum is greater than a certain threshold, the perceptron outputs **1 (Yes/True/Positive)**.
- If it is less than the threshold, the perceptron outputs **0 (No/False/Negative)**.

Example:

Imagine a **student grading system** where a teacher decides whether a student **passes or fails** based on two factors:

1. **Assignment Marks (X_1)**
2. **Exam Marks (X_2)**

The decision is based on the formula:

$$y = f(w_1 \cdot X_1 + w_2 \cdot X_2 + b)$$

- If the final result is **above the threshold**, the student **passes (1)**.
- If the result is **below the threshold**, the student **fails (0)**.

This is how a perceptron makes binary decisions!

4.4. Real-Life Example of a Perceptron

Example 1: Spam Email Detection

A perceptron can classify emails as **Spam (1)** or **Not Spam (0)** based on different features:

Feature	Spam Email (1)	Not Spam (0)
Contains "Free Money"	Yes (1)	No (0)
Contains "Urgent"	Yes (1)	No (0)
Contains "Dear Customer"	Yes (1)	No (0)
Contains "Meeting Invitation"	No (0)	Yes (1)

- If the weighted sum of these features exceeds a threshold, the email is classified as **Spam (1)**.
- If it does not exceed the threshold, it is classified as **Not Spam (0)**.

This is how perceptrons help in real-world applications!

4.5. Limitations of a Perceptron

While perceptrons are useful, they have some limitations:

1. **Cannot Solve Complex Problems:** Perceptrons can only solve **linearly separable problems** (like distinguishing cats vs. dogs), but they **fail** at problems requiring multiple layers (like recognizing handwritten digits).
2. **Single Layer Network:** A simple perceptron cannot handle **nonlinear** problems like the XOR problem.
3. **No Hidden Layers:** More complex decisions require deeper networks with hidden layers.

To overcome these limitations, **multi-layer perceptrons (MLPs)** and **backpropagation** are used in modern neural networks.

5. Single Layer Artificial Neural Network

A **Single Layer Artificial Neural Network (ANN)** is the simplest type of neural network that consists of only one layer of neurons (also called nodes or perceptrons). This type of network is mainly used for simple classification problems where data can be separated into different categories.

Structure of a Single Layer ANN

A single-layer neural network consists of three main components:

1. **Input Layer:** Receives input data (features).
2. **Weights:** Each input is multiplied by a weight, which determines its importance.
3. **Summation Function:** Adds up all the weighted inputs.
4. **Activation Function:** Decides whether a neuron should be activated or not, based on a threshold.
5. **Output Layer:** Produces the final result (prediction).

Mathematical Representation

If we have inputs $x_1, x_2, x_3, \dots, x_n$ and corresponding weights $w_1, w_2, w_3, \dots, w_n$, the neuron processes them as:

$$y = f(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$$

where:

- y is the output,
- w are the weights,
- x are the inputs,
- b is the bias,
- f is the activation function.

Real-Life Example: Email Spam Filter

Imagine you receive an email, and you want to classify it as **Spam or Not Spam** using a single-layer neural network.

1. **Inputs:** Features of an email, such as:
 - Presence of the word “free” (x_1)
 - Presence of “lottery” (x_2)
 - Number of capital letters (x_3)
 - Presence of a link (x_4)
2. **Weights:** Each input has a weight representing its importance.
 - Example: The word “lottery” (x_2) might have a higher weight because spam emails often contain this word.
3. **Summation Function:** Computes the weighted sum of inputs.

$$z = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

4. **Activation Function:**
 - If z exceeds a certain threshold, classify the email as **Spam (1)**.
 - Otherwise, classify it as **Not Spam (0)**.

Thus, the neural network helps in classifying emails as spam or not spam based on the learned patterns.

Limitations of Single-Layer ANN

- Can only handle simple problems where data can be separated with a straight line (Linear Separation).
 - Cannot solve complex problems like recognizing handwritten digits or face recognition.
-

6. Multilayer Perceptron (MLP)

A **Multilayer Perceptron (MLP)** is a type of artificial neural network (ANN) that consists of multiple layers of neurons. It is widely used in tasks like image recognition, speech recognition, and predicting stock prices.

MLP is a **supervised learning** model, meaning it learns from labelled data, where the correct output is known during training.

Structure of a Multilayer Perceptron (MLP)

An MLP consists of three main types of layers:

1. **Input Layer** – This layer receives the input data (features). Each neuron in this layer represents one input feature.
2. **Hidden Layers** – These layers process the input data and learn patterns. An MLP can have one or more hidden layers.
3. **Output Layer** – This layer gives the final result, such as classification (e.g., identifying a cat vs. dog) or regression (e.g., predicting house prices).

Each neuron in one layer is connected to every neuron in the next layer through **weights**. These weights determine the importance of each connection.

Working of Multilayer Perceptron (MLP)

Step 1: Forward Propagation

- The input data passes through the network from the input layer to the output layer.
- Each neuron processes the input using the **weighted sum formula** and applies an **activation function** to introduce non-linearity.
- This allows MLP to learn complex patterns.

Mathematically, a neuron's output is calculated as:

$$y = f(w_1x_1 + w_2x_2 + w_3x_3 + \dots + b)$$

where:

- $x_1, x_2, x_3, \dots$ are input values
- $w_1, w_2, w_3, \dots$ are weights
- b is the bias term
- f is the activation function

Step 2: Error Calculation

- The output of the network is compared with the actual result (ground truth), and an **error (loss)** is calculated.
- The goal of training is to **reduce this error** by adjusting the weights of the network.

Step 3: Backpropagation

- The error is propagated backward through the network.
- The weights are updated using an optimization algorithm like **Gradient Descent** to reduce the error.
- This process is repeated for multiple iterations until the network learns the correct patterns.

Real-Life Example of Multilayer Perceptron (MLP)

Example 1: Handwriting Recognition (Digit Recognition)

Imagine you want to create a system that can recognize handwritten digits (0-9). Here's how an MLP can be used:

1. **Input Layer:** The system takes an image of a handwritten digit (say "7"). The image is converted into pixel values, which serve as inputs.
2. **Hidden Layers:** The network processes these pixel values and extracts important patterns, such as edges and curves.
3. **Output Layer:** The network predicts the digit as "7". If the prediction is incorrect, backpropagation adjusts the weights to improve accuracy.

Over time, the MLP learns to recognize handwritten digits correctly with high accuracy.

Example 2: Spam Email Detection

Consider an email filtering system that detects spam emails.

1. **Input Layer:** The system takes an email as input. It extracts features like the number of spam words (e.g., "win", "free", "lottery").
 2. **Hidden Layers:** The network processes these features and identifies patterns common in spam emails.
 3. **Output Layer:** The system classifies the email as **Spam (1)** or **Not Spam (0)**. If the classification is wrong, the network adjusts its weights through backpropagation to improve future predictions.
-

7. Backpropagation Algorithm

7.1. What is Backpropagation?

Backpropagation (short for *backward propagation of errors*) is an algorithm used to train artificial neural networks by adjusting the weights based on the error in predictions.

How It Works (Step-by-Step Process)

1. **Forward Propagation:** The input is passed through the network layer by layer to generate an output.
2. **Error Calculation:** The difference between the predicted and actual output is calculated (Loss Function).
3. **Backward Propagation:** The error is propagated backward to adjust the weights.
4. **Weight Update:** The weights are updated to minimize the error.

Real-Life Example:

◆ *Google Translate*

- Suppose you type "Hello" and the translation is incorrect.
- The system compares its translation to correct translations and adjusts the model.
- Over time, it improves by learning from mistakes using backpropagation.

7.2. Backpropagation Learning

How Does Backpropagation Help in Learning?

- Backpropagation helps the neural network "**learn**" from mistakes.
- It corrects the weights based on feedback.
- The learning continues until the network achieves high accuracy.

Real-Life Example:

◆ *Self-Driving Cars*

- A car must recognize stop signs and pedestrians.
- If it incorrectly identifies an object, backpropagation helps correct the mistake.
- After several corrections, the car learns to drive safely.

7.3. Training of the Network

Why is Training Needed?

Training a neural network means adjusting its weights so that it can make accurate predictions on new data.

Steps in Training:

1. Provide input data.
2. Compute the output.
3. Compare it with actual output.
4. Adjust weights using backpropagation.
5. Repeat until accuracy improves.

Real-Life Example:

◆ *Face Recognition on Smartphones*

- When setting up face unlock, the phone trains itself to recognize your face.
- It adjusts its internal network to match your unique facial features.
- The more images it sees, the better it gets at recognizing you.

7.4. Calculation of Error

Why is Error Calculation Important?

Error calculation helps the network understand how far its predictions are from the actual values. The smaller the error, the better the model.

Mathematical Formula:

$$\text{Error} = \text{Actual Output} - \text{Predicted Output}$$

Real-Life Example:

✦ *Weather Prediction Apps*

- If a weather app predicts rain but it stays sunny, there is an error in prediction.
- The app calculates this error and improves its future forecasts.

7.5. Method of Steepest Descent Algorithm

What is the Steepest Descent Method?

This method finds the fastest way to reduce the error in a neural network. It updates the weights in the direction that decreases the error the most.

Mathematical Understanding:

$$W_{\text{new}} = W_{\text{old}} - \eta \cdot \frac{\partial E}{\partial W}$$

Where:

- η = Learning rate (step size)
- E = Error
- W = Weight

Real-Life Example:

✦ *Mountain Climbing*

- Imagine you are at the top of a hill and want to reach the lowest point.
- The steepest descent method tells you to move in the direction where the slope is the steepest (fastest way down).
- Similarly, in neural networks, this method helps quickly reduce error.

7.6. Effect of Learning Rate

What is the Learning Rate?

The learning rate (η) determines how much the weights are adjusted in each step.

- **High Learning Rate** → Fast learning but may miss the optimal solution.
- **Low Learning Rate** → Slow learning but more precise adjustments.

Real-Life Example:

✦ *Learning a New Language*

- If you try to learn 100 new words in one day (high learning rate), you may forget them quickly.
- If you learn 5 new words per day (low learning rate), you will remember them better over time.

7.7. Adding Momentum Term

Why is Momentum Needed?

Momentum helps the learning process by preventing the weights from making unnecessary large changes. It ensures smooth learning.

Mathematical Formula:

$$W_{\text{new}} = W_{\text{old}} - \eta \cdot \frac{\partial E}{\partial W} + \alpha \cdot \Delta W_{\text{previous}}$$

Where:

- α = Momentum factor (how much past changes affect the current step)

Real-Life Example:

✦ *Riding a Bicycle*

- If you stop pedaling suddenly, the bike still moves forward due to momentum.
- Similarly, in neural networks, momentum prevents abrupt stops and keeps learning smooth.

UNIT III

ASSOCIATIVE MEMORY

1. Introduction

What is Associative Memory?

Associative Memory is a type of memory used in Artificial Neural Networks (ANNs) that helps store and retrieve data based on patterns rather than exact matches. Unlike traditional memory (where you search for an exact address), associative memory works like our human brain—it remembers information by association.

How Does Associative Memory Work?

Instead of retrieving data based on a specific location (like in a computer), associative memory finds the closest matching pattern. It is useful when:

1. **Exact data is missing or incomplete** (it can still recognize the closest match).
2. **You need to retrieve information based on partial input** (like recognizing a blurry image).

◆ Example:

Imagine you meet a person but forget their name. However, you remember that they have curly hair and wear glasses. Your brain uses associative memory to recall the person based on these features.

Types of Associative Memory

There are two main types:

1. **Autoassociative Memory:**
 - Stores patterns and retrieves the same pattern when given a noisy or incomplete version of it.
 - *Example:* If you see a blurred version of a famous logo (like Apple or Nike), your brain still recognizes it.
2. **Heteroassociative Memory:**
 - Stores patterns and retrieves different but related patterns.
 - *Example:* If you enter a keyword like "cat," Google automatically suggests related terms like "kitten" or "pet food."

Real-Life Examples of Associative Memory

1. Face Recognition in Smartphones

- When you unlock your phone using face recognition, the system does not compare your face pixel by pixel.
- Instead, it recognizes key features (eyes, nose, and shape) and matches them to a stored pattern, even if your face is partially covered.

2. Auto-Correct in Smartphones

- When you type a word with a spelling mistake (like "teh" instead of "the"), your phone suggests the correct word based on associative memory.

3. Image and Voice Recognition (Google Lens, Siri, Alexa)

- If you show Google Lens an incomplete or unclear image, it still finds similar objects based on pattern association.
- Virtual assistants like Siri recognize your voice even if there is background noise.

4. Medical Diagnosis

- Doctors use AI-based neural networks to recognize diseases. If a patient's test results are similar to a stored disease pattern, the system suggests a diagnosis.
-

2. Autocorrelation

Introduction to Associative Memory

Associative memory is a type of memory system in artificial neural networks that helps retrieve stored information when given a partial or related input. It mimics how the human brain recalls information based on associations rather than exact matches.

What is Autocorrelation?

Autocorrelation is a mathematical technique used in associative memory to recognize patterns in data. It measures how a signal, sequence, or pattern relates to itself at different time intervals.

◆ Key Concept:

- It checks for similarities between a signal and a shifted version of itself.
- If a pattern appears multiple times, autocorrelation helps identify it.
- It is commonly used in speech recognition, image processing, and pattern detection.

Understanding Autocorrelation with a Simple Example

Imagine you are **listening to a song**, and suddenly the music stops. However, based on the rhythm and melody you remember, you can predict the next note. This happens because your brain remembers patterns and fills in the missing part. Similarly, autocorrelation helps neural networks recognize repeated patterns in data.

Mathematical Representation of Autocorrelation

Autocorrelation is calculated using the formula:

$$R(k) = \sum_t x(t)x(t - k)$$

Where:

- $R(k)$ = Autocorrelation at time lag k
- $x(t)$ = Signal at time t
- $x(t - k)$ = Shifted version of the signal
- k = Time shift or lag

In simple terms, autocorrelation checks how similar a sequence is to itself after shifting by k steps.

Real-Life Examples of Autocorrelation

1. Weather Forecasting

- Suppose you track daily temperatures.
- If today is very hot, there's a high chance that tomorrow's temperature will also be similar.
- By analysing past temperature patterns, autocorrelation helps predict future weather conditions.

2. Stock Market Trends

- Investors use autocorrelation to analyse stock price movements.
- If a stock's price shows a repeating pattern, traders can predict future price changes based on past trends.

3. Voice Recognition (Google Assistant, Siri)

- When you say a word like "Hello," your voice has a unique wave pattern.
- Autocorrelation helps voice assistants recognize your speech by comparing it with stored wave patterns.

4. Medical Signal Analysis (ECG, EEG)

- Doctors use autocorrelation to detect heart and brain activity patterns.
 - If an ECG signal shows an irregular heartbeat, autocorrelation can help identify heart conditions.
-

3. Heteroassociative Memory (Hetero Correlation)

Introduction to Associative Memory

Before understanding **hetero correlation**, let's first understand **associative memory**.

Associative memory is a type of memory model in artificial neural networks that stores patterns and retrieves them when provided with partial or noisy input. It works similarly to how humans remember things.

For example:

- When you see a blurry image of a person, you might still recognize them based on past experiences.
- If you hear just a part of a song, you can recall the full lyrics.

Associative memories are of two types:

1. **Autoassociative Memory** – The input and output are the same. (Used for pattern completion.)
2. **Heteroassociative Memory** – The input and output are different. (Used for pattern mapping.)

What is Hetero Correlation (Heteroassociative Memory)?

Heteroassociative memory is a type of neural network memory where the input pattern is different from the output pattern. It **associates one set of data with another**.

This means that if you input a certain pattern, the network retrieves a different corresponding pattern, similar to how a dictionary maps words to meanings.

Mathematical Representation of Hetero Correlation

Heteroassociative memory is represented as a weight matrix W that links an input vector X to an output vector Y :

$$Y = W \times X$$

Where:

- X is the input vector (data fed into the network)
- Y is the output vector (retrieved data)
- W is the weight matrix (learned memory mapping input to output)

This method helps in pattern mapping, where one type of input is associated with a different type of output.

This method helps in pattern mapping, where one type of input is associated with a different type of output.

Real-Life Examples of Hetero Correlation

1. Language Translation (Google Translate)

- When you input an English word, Google Translate provides its equivalent in another language (e.g., "Hello" → "Hola" in Spanish).
- Here, the input pattern (English words) and output pattern (Spanish words) are different, showing **heteroassociative memory** in action.

2. Speech-to-Text Conversion (Voice Assistants)

- When you speak a sentence to Siri, Google Assistant, or Alexa, it converts the speech (audio signal) into written text.
- Here, **sound waves (input)** are mapped to **text (output)**, making it an example of hetero correlation.

3. Image-to-Text Recognition (OCR Technology)

- Optical Character Recognition (OCR) converts images of handwritten or printed text into editable digital text.
- The input is an **image of text**, and the output is **digital text** (e.g., PDF to Word conversion).
- This is another application of heteroassociative memory, where an image pattern is mapped to a text pattern.

4. Medical Diagnosis (AI in Healthcare)

- AI systems take patient symptoms as input and provide possible disease predictions as output.
- For example, inputting **fever, cough, and fatigue** may output **Influenza or COVID-19**.
- This shows how different input (symptoms) are mapped to different outputs (disease names).

4. Multiple Training Encoding Strategy

Introduction

In Artificial Neural Networks (ANNs), **Associative Memory** is a type of memory that allows the network to recall stored patterns based on partial or noisy input. It works like the human brain, which remembers faces, names, and experiences based on associations.

A **Multiple Training Encoding Strategy** is a method used in associative memory where a neural network is trained with multiple patterns and examples to improve its ability to recall information accurately. This strategy helps the network learn various patterns and associations more efficiently.

Understanding Multiple Training Encoding Strategy

1. What is Encoding in Neural Networks?

Encoding refers to the process of storing information in a network. In the case of associative memory, multiple patterns or data points are stored together so that when a partial or similar input is given, the correct stored pattern can be retrieved.

2. What is Multiple Training?

Instead of training a network with a single pattern, **multiple training encoding** involves training the network with various patterns multiple times to improve recall and generalization. This method ensures that the network can recognize different variations of the same data.

3. How Does Multiple Training Encoding Work?

1. **Multiple Patterns Are Stored** – The network is trained with multiple different patterns or examples.
2. **Training is Repeated** – The network goes through multiple training cycles to strengthen memory.
3. **Noise Handling Improves** – The network becomes more resistant to errors and noise in input data.
4. **Better Generalization** – The network learns to associate partial or distorted input with correct stored patterns.

Real-Life Examples of Multiple Training Encoding Strategy

Example 1: Face Recognition in Smartphones

- When you set up **Face ID** on your phone, it asks you to scan your face multiple times.
- The phone collects different angles and lighting conditions to **train the model** using multiple training encoding.
- Later, even if you wear glasses or change your hairstyle, the phone can still recognize you because it has learned variations of your face.

Example 2: Learning Handwriting Styles (OCR - Optical Character Recognition)

- Imagine an AI system that reads handwritten text (like Google Lens or bank cheque scanners).
- It is trained on multiple handwriting styles (cursive, block letters, and slanted text).
- Even if someone writes the same letter differently, the system can recognize it correctly.

Example 3: Voice Assistants (Alexa, Siri, Google Assistant)

- When you train your voice assistant, it asks you to say phrases multiple times.
- This helps the AI **learn your voice** under different conditions (soft voice, loud voice, noisy background).
- Later, it recognizes your commands accurately, even if there is some background noise.

Example 4: Self-Driving Cars - Traffic Sign Recognition

- Self-driving cars use neural networks to recognize traffic signs like "STOP" or "SPEED LIMIT."
 - The AI is trained with multiple images of the same sign under different weather conditions (rain, fog, night, bright sunlight).
 - This ensures that the car correctly identifies the sign, even if it is partially covered or unclear.
-

5. Hopfield Network

Introduction

A **Hopfield Network** is a type of **recurrent artificial neural network** that is used for **associative memory**. It can store and recall patterns, similar to how humans remember familiar faces or words. Unlike traditional feedforward networks, Hopfield networks have **bidirectional** connections, meaning that each neuron is connected to every other neuron.

Understanding Hopfield Networks

A Hopfield Network consists of:

- ✓ **Neurons (Nodes):** These represent different stored patterns.
- ✓ **Connections (Weights):** Each neuron is connected to all other neurons.
- ✓ **Activation Function:** Determines whether a neuron should be active (ON) or inactive (OFF).

How It Works:

1. **Training Phase:** The network is trained by storing patterns in the form of weight matrices.
2. **Recall Phase:** When a partial or noisy version of a stored pattern is given as input, the network recalls the closest stored pattern.
3. **Energy Minimization:** The network always moves toward the most stable state, just like how our brain recognizes faces even in low light.

Real-Life Example: Recognizing a Blurry Image

Imagine you are looking at a **blurry photo of your best friend**. Even if some details are missing, your brain recognizes the face based on memory.

◆ In a **Hopfield Network**, if you input an **incomplete pattern**, the network **recalls the closest stored pattern**, just like how your brain fills in missing details!

6. Error Performance of Hopfield Network

Introduction

The **error performance** of a Hopfield Network refers to how well it can recall stored patterns **without mistakes**, especially when the input is noisy or incomplete. The performance is affected by factors like:

- ✓ **Number of stored patterns** – Too many patterns can cause confusion.
- ✓ **Noise in input** – If the input is too distorted, the network may retrieve the wrong pattern.
- ✓ **Network Stability** – The network should converge to a correct memory state.

How Errors Affect Hopfield Networks

1. **Memory Overload:** If a Hopfield Network tries to store too many patterns, it may fail to retrieve the correct one.
2. **Spurious States:** Sometimes, the network may recall a wrong or mixed pattern (like a blend of two stored images).
3. **Convergence Issues:** If the network does not have the correct weight adjustments, it may not reach the right memory state.

Real-Life Example: Autocorrect on Smartphones

- When you type a word incorrectly, your phone suggests the correct word based on stored memory.
 - If too many similar words are stored (e.g., "hello" and "hell"), autocorrect might choose the wrong word.
 - This is similar to **error performance in Hopfield Networks**, where too many patterns can lead to wrong recalls.
-

7.Exponential Bidirectional Associative Memory (Exponential BAM)

Introduction

Exponential Bidirectional Associative Memory (Exponential BAM) is a type of associative memory used in neural networks. Associative memory refers to a system that stores and recalls patterns based on input-output associations, just like how our brain remembers related information.

BAM is an **advanced neural network model** that extends the traditional Hopfield network by introducing bidirectional connections. The **Exponential BAM** model further improves memory storage capacity and recall efficiency by utilizing **exponential weight functions** instead of linear ones.

Understanding Exponential BAM

1. What is Bidirectional Associative Memory (BAM)?

- BAM is a type of neural network that **stores pairs of related patterns**.
- If the network is trained with an input-output pair, it can recall **one when given the other** (bidirectional recall).
- BAM networks contain **two layers of neurons** that interact with each other in **both directions** (forward and backward propagation).

2. Why is it Called "Exponential BAM"?

- Traditional BAM networks use a **linear weight update rule** for learning.
- **Exponential BAM**, however, **modifies the weight update rule using an exponential function**, which improves the memory storage capacity and stability of the network.
- This means it can **store more patterns** and recall them more **accurately and efficiently**.

How Does Exponential BAM Work?

Step 1: Training the Network

- The network is trained by providing **paired patterns** (e.g., "Hello" ↔ "Hola").
- Weights are updated using an **exponential function** rather than a simple sum.

Step 2: Storing Patterns

- Patterns are stored in a way that when one part of a pair is given, the other can be recalled.
- The use of **exponential weight functions** allows more efficient and noise-resistant storage.

Step 3: Recalling Information

- If part of a stored pattern is given as input, the network retrieves the corresponding pattern using bidirectional connections.

Real-Life Examples of Exponential BAM

1. Language Translation (Google Translate)

- Suppose a neural network is trained with English-Spanish word pairs:
 - "Apple" ↔ "Manzana"
 - "Book" ↔ "Libro"
- With **Exponential BAM**, even if you input a misspelled word like "Appl," the network can still recognize it and retrieve "Manzana" accurately.
- This **improves translation accuracy** compared to traditional BAM.

2. Image Recognition (Face Unlock in Phones)

- If a smartphone stores your **front and side profile images** using BAM, it can recognize you **from different angles**.
- **Exponential BAM enhances memory stability**, making it more **accurate even under varying lighting conditions**.

3. Medical Diagnosis (Disease Symptom Matching)

- Doctors often diagnose diseases based on **symptom-disease pairs** stored in a system.
 - If a patient provides **partial symptoms**, Exponential BAM helps recall the **most likely disease** based on stored medical records.
 - This allows **faster and more accurate diagnoses**.
-

8. Associative Memory for Real-Coded Pattern Pairs

Introduction

Associative memory is a type of memory system in artificial neural networks that allows the system to remember and retrieve stored patterns based on partial or noisy input. It works similarly to how the human brain recalls information based on associations.

In **real-coded pattern pairs**, the memory is trained to store numerical (real number) patterns and retrieve the correct output when a similar input is given.

How Does Associative Memory Work?

1. **Training Phase:** The system is trained with input-output pairs (real number patterns).
2. **Storage:** The system stores these patterns in memory.
3. **Retrieval Phase:** When a new or slightly modified input is given, the system retrieves the most similar stored pattern.

◆ **Example:** *Face Recognition in Smartphones*

- Your phone stores your facial features as a pattern.
- When you try to unlock it, the system compares your current facial scan with stored patterns.
- If the match is close enough, the phone unlocks.
- Even if you wear glasses or have a slight beard, the system can still recognize you using associative memory.

Mathematical Concept (Simplified)

Associative memory uses **weight matrices** to store patterns. If we represent input-output pairs as vectors:

$$X = (x_1, x_2, x_3, \dots, x_n)$$

$$Y = (y_1, y_2, y_3, \dots, y_n)$$

The system learns a weight matrix W that maps input X to output Y :

$$Y = W \cdot X$$

Over multiple training cycles, the weight matrix is adjusted so that the correct output is retrieved even with slight variations in input.

9. Character Recognition Application

What is Character Recognition?

Character recognition is a practical application of associative memory where the system learns to recognize letters, numbers, or symbols from handwritten or printed text. It is widely used in:

- ✓ OCR (Optical Character Recognition) in scanners
 - ✓ License plate recognition in traffic systems
 - ✓ Digit recognition in postal services (sorting mail)
-

How Does It Work?

1. **Pre-processing:** The image of a character is converted into a numerical format (grayscale values or pixel intensities).
 2. **Feature Extraction:** Important features like edges, curves, and strokes are extracted.
 3. **Pattern Matching:** The system compares the extracted features with stored patterns in associative memory.
 4. **Recognition:** The closest match is selected as the correct character.
-

Real-Life Example: OCR in Bank Cheques

Banks use OCR to automatically read handwritten amounts on cheques:

- The cheque is scanned and converted into a digital image.
- The OCR system extracts numerical patterns from the cheque.
- Associative memory helps the system recognize handwritten numbers and convert them into text.
- The recognized amount is then processed in the banking system.

UNIT IV

ADAPTIVE RESONANCE THEORY

1. Introduction to Adaptive Resonance Theory (ART)

1.1. What is Adaptive Resonance Theory (ART)?

Adaptive Resonance Theory (ART) is a machine learning model used in artificial neural networks for pattern recognition and classification. It was developed by **Stephen Grossberg and Gail Carpenter** in the 1980s.

1. Why is ART Important?

- Traditional neural networks often **forget** previously learned information when trained with new data.
- ART networks are **designed to learn continuously** without forgetting old patterns.
- They are widely used in **speech recognition, handwriting recognition, medical diagnosis, and fraud detection.**

2. How Does ART Work?

ART follows two key principles:

1. **Adaptive Learning** – The network learns new information without forgetting old knowledge.
2. **Resonance Condition** – The network checks whether new data fits into an existing category or needs a new category.

3. Key Terms in ART:

- **Recognition Field:** A collection of neurons that classify input patterns.
- **Comparison Field:** Compares new input with stored patterns.
- **Reset Mechanism:** If the new input doesn't match any stored pattern, a new category is created.

4. Real-Life Example of ART: Sorting Fruits in a Supermarket

Imagine you are working in a supermarket, sorting fruits into different baskets.

- You see an **apple** and place it in the "Apple" basket.
- You get a **banana** and place it in the "Banana" basket.
- You then see a **green apple** and compare it with the existing apple basket. If it matches, you put it there.
- Now, you see a **mango**, which is a completely new fruit. Since no basket exists for mangoes, you create a new "Mango" basket.

□ **How ART Works in This Example:**

✓ If the new fruit matches a known category → **It gets stored in an existing group.**

✗ If the new fruit is different → **A new category is created.**

This is exactly how ART networks work. They compare new inputs with previous data and decide whether to classify them into an existing category or create a new one.

5. Applications of ART in Real Life

1 □ **Handwriting Recognition (Google OCR, Signature Verification)**

- ART helps computers recognize different handwriting styles without forgetting old ones.
- When a person writes a new letter style, the system either identifies it correctly or stores it as a new pattern.

2 □ **Medical Diagnosis**

- ART is used in detecting diseases like cancer by recognizing patterns in medical images.
- It classifies new patient data based on previously learned patterns but also creates new categories if a unique disease pattern appears.

3 □ **Face Recognition (Facebook, Security Systems)**

- ART is used in recognizing faces even if a person changes their hairstyle, wears glasses, or ages over time.
 - It learns new facial features while retaining older patterns.
-

2. Cluster Structure

A **cluster** is a group of similar data points grouped together based on common characteristics.

Cluster Structure in ART refers to the way in which patterns (data points) are grouped into different categories without losing previously learned knowledge. This helps in classification and pattern recognition.

2.1. How Does Clustering Work in ART?

Clustering in ART is done in three main steps:

1. **Input Processing:**
 - The system takes an input pattern (data).
 - It checks if the pattern is similar to an existing cluster (group).
2. **Matching & Resonance Check:**
 - If the input is **similar** to an existing cluster, it is added to that cluster.
 - If the input is **not similar** to any existing cluster, a **new cluster** is created.
 - The system ensures that clusters remain stable while allowing learning of new patterns.
3. **Weight Update & Learning:**
 - If a pattern is assigned to a cluster, the system **reinforces** the connection.
 - If a new cluster is created, the system learns new characteristics.

2.2. Real-Life Examples of Cluster Structure in ART

1. Email Spam Filtering

- Your email service (like Gmail) categorizes emails as **spam** or **not spam**.
- ART helps by grouping emails into clusters based on words, sender address, and patterns.
- If a new spam email comes in, ART either adds it to an existing spam cluster or creates a new one if it's a unique type of spam.

2. Medical Diagnosis (Cancer Detection)

- ART is used in **medical imaging** to detect diseases like cancer.
- It groups medical images into clusters: **healthy tissue**, **benign tumor**, and **malignant tumor**.
- If a new image is similar to a known tumor pattern, it is classified accordingly.
- If the tumor has a **new pattern**, a separate cluster is created, helping doctors identify rare conditions.

3. Fingerprint Recognition

- ART is used in **biometric security systems** to classify fingerprints.
- It creates clusters of similar fingerprint patterns.
- If your fingerprint matches an existing cluster, access is granted; if not, a new cluster may be created (for new users).

4. Customer Segmentation in E-Commerce

- Websites like **Amazon and Flipkart** use ART to group customers based on shopping behaviour.
- It creates clusters like:
 - **Frequent buyers**
 - **Bargain hunters**
 - **Luxury shoppers**
- When a new customer joins, ART checks their shopping behaviour and places them in an appropriate cluster.

3. Vector Quantization

3.1. Introduction to Vector Quantization

Vector Quantization (VQ) is a technique used in machine learning and neural networks for data compression, pattern recognition, and clustering. It is a process where data points are represented by vectors, and similar vectors are grouped into "clusters." VQ maps input data into a limited number of representative vectors, reducing the dimensionality of the data while retaining essential information.

In ART (Adaptive Resonance Theory), vector quantization helps in grouping data into categories, allowing the system to learn and recognize patterns.

3.2. How Does Vector Quantization Work?

- 1. Input Data:**
A set of data points (vectors) is fed into the system. These vectors represent different features of the data. For example, in image compression, each vector might represent pixel intensities or colours.
- 2. Codebook:**
A "codebook" is a collection of vectors that represent the learned patterns. Initially, these vectors might be random but will be refined as the system learns.
- 3. Distance Measurement:**
The algorithm calculates the distance between each input vector and the vectors in the codebook. The most similar vector (the one with the smallest distance) is chosen as the representative for that input.
- 4. Updating the Codebook:**
After identifying the closest codebook vector, the system adjusts the codebook to make the chosen vector more representative of the input data. This is done by moving the vector closer to the input vector.
- 5. Cluster Formation:**
As the system continues learning, similar input vectors are grouped together. This results in the creation of clusters, where each cluster represents a specific feature or pattern in the data.

3.3. Real-Life Example of Vector Quantization

1. Image Compression (JPEG, PNG Compression)

- **Context:** When you compress an image, the goal is to reduce its file size while keeping the quality as high as possible.
- **How it works:**
Vector quantization is used to represent regions of the image by similar pixel values. For example, in a photo with lots of blue sky, the pixels in that area might have similar color values (shades of blue). These similar pixels are grouped together in a vector. Instead of storing each pixel individually, the system stores a representative vector, which significantly reduces the amount of data needed to represent the image.

- **Real-World Use:**
 - **JPEG compression** uses a similar concept where the image is divided into blocks, and each block is represented by a codebook of values. This technique is part of the reason why images can be compressed without a significant loss of quality.

2. Speech Recognition (Voice Assistants like Siri, Alexa)

- **Context:** Voice assistants need to recognize words or phrases and respond appropriately.
- **How it works:**
Vector quantization is used in speech recognition systems to compress the voice signal into vectors that represent the sound patterns. For example, similar sounds in a word like "hello" or "hi" are grouped into a cluster. When a user speaks, the system compares the sound with pre-learned patterns and recognizes the word based on the closest match.
- **Real-World Use:**
 - **Speech-to-Text systems** like Siri or Google Assistant often use vector quantization to identify spoken words from a range of possible sounds.
 - These systems identify and categorize sounds in real-time, making it easier for voice assistants to interpret commands.

3. Image Recognition (Facial Recognition Systems)

- **Context:** In facial recognition, a system must identify faces in images or videos.
- **How it works:**
The system breaks down the features of a face (eyes, nose, mouth, etc.) into vectors. Vector quantization helps the system group similar facial features (e.g., the distance between the eyes) into a cluster. When a new image is presented, the system compares it with the learned features in the codebook to determine if the face matches an existing one.
- **Real-World Use:**
 - **Facial recognition** systems used in smartphones and security cameras use vector quantization to efficiently process and recognize faces.
 - For instance, Apple's Face ID uses vector quantization in recognizing your face to unlock your phone.

3.4. Key Steps in Vector Quantization:

1. **Initialization:** Start with a random set of vectors (codebook).
2. **Matching:** For each input vector, find the closest vector in the codebook.
3. **Update:** Adjust the codebook by moving the codebook vector closer to the input vector.
4. **Repeat:** Continue the process for multiple iterations until the codebook stabilizes and accurately represents the input data.

3.5. Applications of Vector Quantization:

- **Data Compression:** Reduces data storage and transmission requirements without significant loss of quality.
- **Pattern Recognition:** Helps in classifying and clustering similar data, especially in areas like speech recognition, image processing, and data mining.
- **Neural Networks and Machine Learning:** Used in clustering algorithms, like K-means, and in systems that need to learn from patterns.

3.6. Advantages of Vector Quantization:

- **Efficient Data Representation:** Compresses data by representing it with fewer vectors.
- **Noise Reduction:** Helps in reducing noise by grouping similar features together.
- **Real-Time Processing:** Especially useful in systems that need to process large amounts of data quickly, like real-time image or speech recognition.

3.7. Challenges:

- **Choosing the Right Codebook Size:** If the codebook is too small, it may not capture enough information, leading to poor representation. If it's too large, it can become computationally expensive.
 - **Convergence:** The system may take many iterations to stabilize, especially with large datasets.
-

4. Classical ART Network

4.1. Introduction

The **Adaptive Resonance Theory (ART)** is a type of neural network designed to learn new patterns without forgetting previously learned ones. It was developed by **Stephen Grossberg** in the 1970s. ART networks are particularly useful for tasks that require online learning, where the network needs to adapt to new information over time while preserving its previous knowledge. This theory addresses a common issue in neural networks called **catastrophic forgetting**, where the network forgets previously learned data when learning new data.

How Does the Classical ART Network Work?

The classical ART network consists of two main layers:

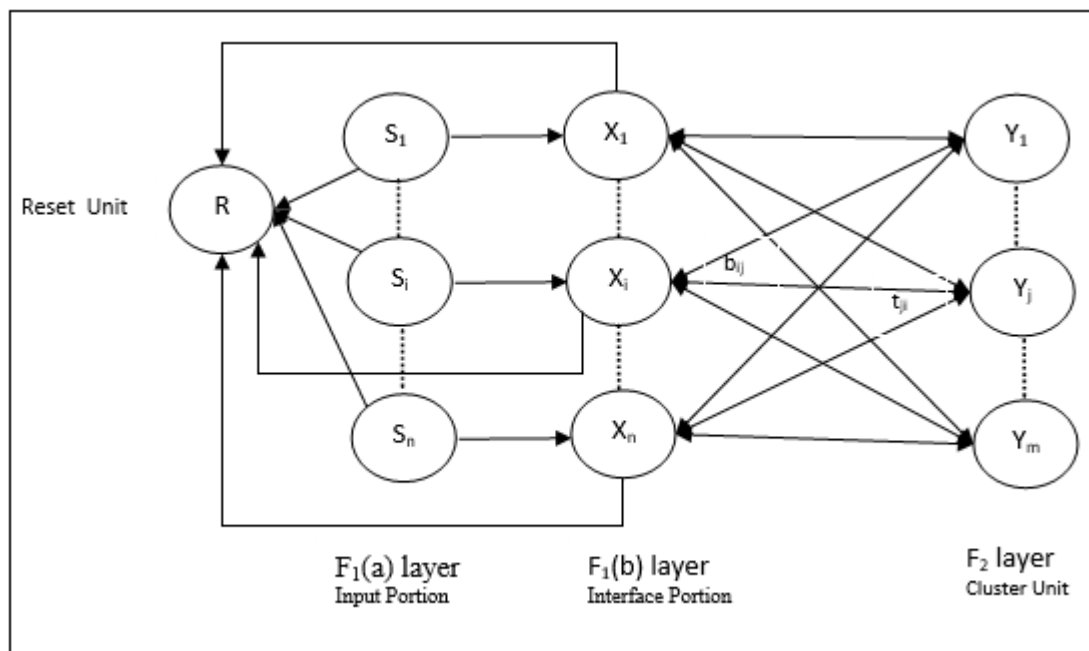
1. **F1 Layer (Input Layer)**
2. **F2 Layer (Category Layer)**

F1 Layer (Input Layer)

- This layer receives input data. It consists of neurons that represent different features or characteristics of the data.
- The input data is compared with the patterns stored in the F2 layer.

F2 Layer (Category Layer)

- This layer contains neurons that represent different learned categories or patterns.
- The neurons in F2 become activated if the input data is similar to a stored pattern.



Working Process:

1. **Pattern Recognition:** The input pattern from the F1 layer is compared with the patterns stored in the F2 layer.
2. **Resonance:** If the input pattern is similar enough to a pattern in F2 (i.e., it "resonates"), the network updates the stored category. This is called **resonance** because the network's response is based on the similarity between input and stored patterns.
3. **Learning:** If the input pattern does not resonate with any existing category, a new category is created, and learning happens without disturbing the old categories. This is achieved through a process called **vigilance**. Vigilance ensures that the network creates a new category only when the input is sufficiently different from all existing categories.

4.2. Real-Life Example of Classical ART Network:

Face Recognition System:

- Consider a facial recognition system used in smartphones.
- The system stores the patterns of faces it has seen in the past.
- When a new face appears, the system compares it with the stored faces. If the new face is similar to a stored one (i.e., it "resonates" with an existing pattern), the system recognizes it.
- If the system encounters a completely new face, it creates a new category for this face without losing the previously learned faces, which is a key feature of ART.

Advantages of Classical ART Network:

- Can learn continuously without forgetting old knowledge.
- Works well with noisy or incomplete data.
- Adaptable to dynamic environments.

Challenges:

- Can be computationally intensive.
 - Requires careful tuning of vigilance parameters to avoid excessive category creation.
-

5. Simplified ART Architecture

5.1. Introduction

The **Simplified ART Architecture** is a more efficient version of the classical ART network. It simplifies some of the mechanisms involved in learning, making it faster and easier to implement. The simplified version still maintains the core principles of **resonance** and **vigilance** but reduces the complexity of the original ART.

5.2. How Does the Simplified ART Architecture Work?

In the simplified ART architecture, the overall structure is still similar to the classical ART network, with two layers:

1. **F1 Layer (Input Layer)**
2. **F2 Layer (Category Layer)**

However, some changes are made in how the patterns are compared and learned:

Simplified Mechanisms:

1. **Pattern Matching:** The input from the F1 layer is matched with the stored patterns in F2, but the complexity of comparison and learning is reduced compared to the classical version.
2. **Vigilance Control:** The vigilance parameter is more straightforward in the simplified ART network, and the creation of new categories is done in a more direct manner.
3. **Learning Speed:** The simplified architecture improves the speed at which the network learns new patterns, making it suitable for real-time applications.

5.3. Real-Life Example of Simplified ART Architecture:

Spam Email Filtering:

- In a simplified ART network used for spam email filtering, the network continually learns new patterns of spam emails.
- Initially, the system recognizes common spam patterns, but as it encounters new types of spam, it adapts by creating new categories without forgetting the old ones.
- The simplified ART makes this process faster and more efficient, allowing the system to quickly adapt to new forms of spam.

5.4. Advantages of Simplified ART Architecture:

- Faster learning and adaptation to new data.
- Less computationally intensive compared to the classical ART.
- Can handle dynamic data efficiently.

5.5. Challenges:

- May not be as robust in handling very complex data compared to the classical ART.
 - Requires careful tuning of parameters for best performance.
-

6. ART1 Architecture

The **ART1 model** is a specific version of ART, designed to handle binary input data (i.e., data with values of 0 or 1). The architecture of ART1 consists of two main layers:

1. **Input Layer (X)**
 - This layer receives the binary input vector (data) and passes it to the next layer. The input vector typically represents some features of the data.
 - Each node in the input layer represents a particular feature of the input data.
2. **Comparison Layer (T)**
 - This layer compares the input data to stored patterns and determines which category the data belongs to. If the pattern matches, it resonates (activates).
 - Each node in the comparison layer represents a potential "category" or "class."
3. **Learning Layer**

- A feedback connection from the comparison layer to the input layer helps fine-tune the recognition process, ensuring that the system continues to update its knowledge base.

2. Fast Learning and Self-Organization

ART networks are self-organizing, meaning they automatically categorize input data into clusters or groups based on similarity. They can learn quickly without the need for supervision, and they adjust their internal structure as more data is received.

3. Bottom-Up and Top-Down Processing

ART networks use a combination of **bottom-up processing** (from input to comparison) and **top-down processing** (feedback from the comparison layer). This dual-processing approach helps ART networks adapt to new data while retaining previously learned patterns.

4. Category Formation

When the system receives new data that does not match any existing category, it creates a new category. This allows ART networks to handle continuously changing data and form new categories dynamically.

6.2. ART1 Algorithm

The **ART1 learning algorithm** involves the following steps:

1. **Input Presentation:** The input pattern (binary vector) is presented to the input layer.
2. **Comparison with Stored Patterns:**
 - The input is compared with each stored pattern in the comparison layer.
 - The network checks if the input pattern resonates with any existing category (stored pattern). This is done using a **vigilance criterion** (a threshold value), which determines how closely the input must match a category to "resonate."
3. **Resonance or Search for a New Category:**
 - If the input pattern resonates with a stored pattern (i.e., the match is strong enough), the corresponding category is activated, and learning takes place.
 - If no resonance is found, a new category is created, and the network adjusts its weights to learn the new pattern.
4. **Weight Adjustment:**
 - After resonance, the weights between the input layer and the comparison layer are updated to reinforce the match. The learning is done based on the new pattern.
5. **Iteration:**
 - The process continues iteratively. As new data is presented, the ART network either adjusts existing categories or creates new ones, always ensuring stable learning and avoiding catastrophic forgetting.

6.3. Real-Life Example of ART1 in Action

Example 1: Customer Segmentation in Marketing

Imagine a marketing company using ART1 to segment customers based on their purchasing behaviour.

- **Step 1:** Customers' purchasing data is input into the ART1 network (e.g., frequency of purchases, types of products bought).
- **Step 2:** The ART1 network compares this data with existing customer categories (e.g., "frequent buyers," "occasional buyers").
- **Step 3:** If the customer's behaviour resonates with an existing category, they are assigned to that category. If not, a new category is created, and the network adjusts the weights to include the new customer type.
- **Outcome:** The network can continuously update customer segments as new purchasing data comes in, while still retaining the old categories.

Example 2: Anomaly Detection in Security Systems

ART1 can also be used in anomaly detection, such as identifying fraudulent activities in banking transactions.

- **Step 1:** A transaction's features (e.g., amount, location, time) are fed into the ART1 network.
 - **Step 2:** The system compares this transaction to previous patterns of normal transactions.
 - **Step 3:** If the transaction is unusual but still resembles a known pattern (e.g., a high-value transaction from a frequent customer), it is flagged as normal. If the transaction is significantly different, a new category for potential fraud is created.
 - **Outcome:** The system can adapt to new types of fraud without forgetting previously learned, legitimate transaction patterns.
-

7. ART2 Architecture

7.1. Introduction to ART2

Adaptive Resonance Theory (ART) is a type of neural network designed to model the process of learning and memory in the human brain. ART2 is a specific model of ART that is used primarily for processing continuous and real-valued input patterns. ART2 is particularly well-known for its ability to handle non-stationary data and avoid catastrophic forgetting, which is a common issue in many learning models.

7.2. Key Features of ART2

1. **Stable Learning:**
ART2 ensures that the network learns stably without forgetting previously learned information, even as new data is introduced.

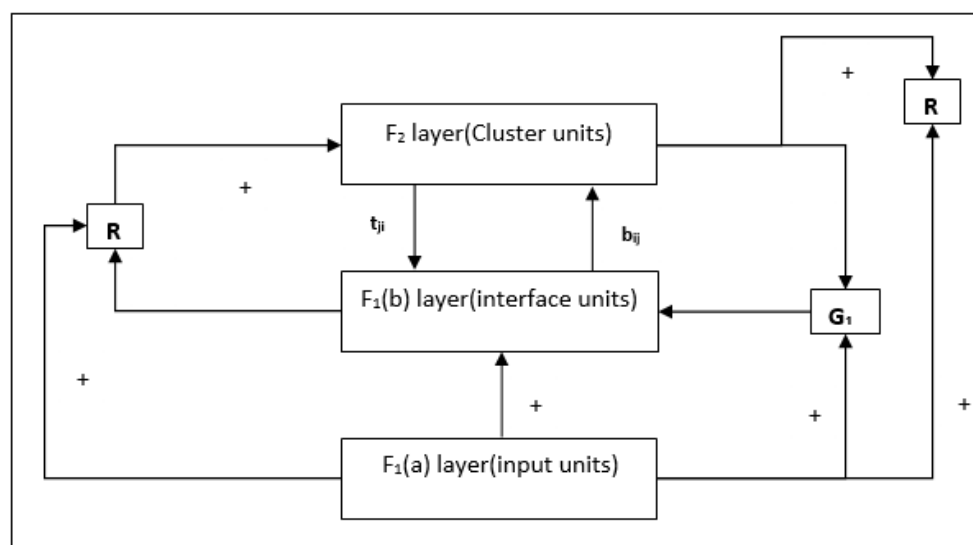
2. **Cluster Formation:**
ART2 clusters data based on similarity, so similar patterns are grouped together. It is very effective at detecting patterns in large datasets.
3. **Top-down and Bottom-up Connections:**
The ART2 network uses feedback loops between the input layer and the output layer to refine the learning process.
4. **Competitive Learning:**
Neurons in the ART2 network compete to represent the input data. This competition ensures that only the most appropriate neurons are activated based on the input.

7.3. Architecture of ART2

The architecture of ART2 consists of two main layers: the **input layer (F1)** and the **category layer (F2)**, with several additional components for controlling the learning process.

Key Components in ART2 Architecture:

1. **F1 Layer (Input Layer):**
The input layer is where data is fed into the network. It processes continuous values (real-valued inputs) and sends them to the category layer. Each input neuron represents a feature of the data.
2. **F2 Layer (Category Layer):**
The category layer is responsible for representing different categories or clusters of data. Each neuron in this layer is activated based on the similarity between the input data and the weights of the neurons in the F2 layer.
3. **Bottom-up and Top-down Pathways:**
 - The **bottom-up pathway** sends the input signal from the F1 layer to the F2 layer.
 - The **top-down pathway** sends feedback from the F2 layer back to the F1 layer. This feedback helps refine the categorization and learning process.
4. **Vigilance Parameter:**
This is a key feature of ART2 that controls the degree of similarity required for an input pattern to be assigned to a specific category. If the input is too different from an existing category, a new category is created.



7.4. Special Features of ART2

1. **Vigilance Mechanism:**

- The vigilance parameter plays a crucial role in ART2 by adjusting the sensitivity of the network to new inputs.
- **High vigilance:** The network will be stricter, and only patterns very similar to the existing categories will be grouped together.
- **Low vigilance:** The network is more lenient, and a broader range of patterns can be assigned to the same category.

2. **Dynamic Learning:**

- ART2 allows the network to adapt dynamically without catastrophic forgetting. It can incorporate new patterns without affecting previously learned patterns.

3. **Incremental Learning:**

- ART2 is capable of incremental learning, which means that the network can continuously learn new information without the need to retrain on all previous data.

7.5. ART2 Algorithm

The ART2 algorithm works in two main phases: the **Recognition Phase** and the **Learning Phase**.

1. *Recognition Phase (Matching Process):*

- The input pattern is compared with the stored categories in the F2 layer.
- If the input pattern is sufficiently similar to an existing category (based on the vigilance parameter), it is assigned to that category.
- If the input pattern is not similar enough to any category, a new category is created.

2. *Learning Phase (Updating Weights):*

- If the input pattern is assigned to an existing category, the weights of the corresponding F2 neuron are adjusted to reinforce the representation of that pattern.
- The network updates the weights of the neurons in both the F1 and F2 layers to improve the match for future inputs.

The weights are updated using a learning rule that adjusts the weights based on the input and the output neurons that were activated.

7.6. Real-Life Examples of ART2

1. *Image Compression (Pattern Recognition):*

Imagine a system designed to compress images by clustering similar pixel patterns together.

- The ART2 model would learn to group similar pixel patterns into categories, making it possible to compress the image data efficiently.

- For instance, images of skies and oceans could be clustered into similar categories based on color patterns, reducing the data needed for storage.

2. Speech Recognition Systems:

In speech recognition, ART2 can be used to recognize patterns in speech data.

- As a user speaks, the ART2 system categorizes different speech patterns (e.g., vowels, consonants, words) based on input features like pitch, tone, and frequency.
- The vigilance mechanism helps the system differentiate between similar-sounding words, ensuring accurate recognition even in noisy environments.

3. Anomaly Detection in Cyber security:

ART2 is effective in detecting unusual patterns in network traffic.

- If new, malicious activity (like a cyber-attack) doesn't match any existing patterns in the system, ART2 can create a new category for that type of behaviour, alerting security teams to a potential threat.
- This system helps in real-time monitoring and detection of anomalies in large datasets, improving cyber security.

8. How ART1 Works for Character Recognition

In **character recognition**, ART1 is used to identify patterns like **letters or numbers** from an input image. The key steps involved in recognizing characters using ART1 are as follows:

1. Input Encoding:

The input image is converted into a binary vector. This vector represents the features of the character (for example, lines, curves, and shapes). The image might look something like a **handwritten letter** which is scanned or digitized. Each letter or character is represented as a set of binary values that reflect its features.

2. Input Pattern Comparison:

The input binary vector is fed into the network. In ART1, the comparison layer compares the input pattern with the stored prototypes (previously learned character patterns). The network searches for the closest match using similarity measures.

3. Resonance and Activation:

When a match is found between the input vector and an existing prototype, the recognition layer becomes activated. This process is called **resonance**. The network "remembers" the matched character and recognizes it.

4. Category Formation:

If no match is found, a new prototype is created, and the network learns a new category (new character). This is one of the key strengths of ART1: it can **learn new characters without forgetting previous ones**.

5. **Weight Adjustment:**

Once the match or new pattern is recognized, the weights in the network are adjusted to strengthen the pattern recognition ability for future inputs.

Real-Life Example: Character Recognition in Postal Systems

Problem:

Postal services need to automatically read handwritten addresses on envelopes to sort them efficiently. Handwritten characters may vary greatly in style and size, and new styles can appear daily.

Solution with ART1:

- **Input:** Each handwritten address is converted into a binary pattern (based on features of the letters, like curves, straight lines, etc.).
- **Processing:** ART1 compares these patterns to stored prototypes of known characters (e.g., "A", "B", "C", etc.).
- **Resonance:** If the pattern closely matches one of the stored character prototypes, the letter is recognized correctly.
- **Learning:** If a new character is encountered (say a new style of "A"), ART1 will learn the new character and add it to the prototypes without forgetting the previously learned ones.

This system allows the postal service to sort letters and parcels without manual intervention, even as new handwriting styles emerge.

Advantages of ART1 in Character Recognition

1. **Incremental Learning:** ART1 can learn new characters without erasing old ones, which is essential for real-time applications where new inputs constantly appear.
2. **Robustness to Noise:** ART1 can recognize characters even in noisy or distorted images (e.g., faint handwriting or partially obscured letters).
3. **Unsupervised Learning:** ART1 does not require labelled data for training, making it suitable for dynamic environments where patterns may change.

Challenges

1. **Input Representation:** The input data needs to be carefully encoded into binary vectors, which can be complex for certain types of data.
 2. **Network Tuning:** The network needs to be carefully tuned to handle specific types of character variations effectively.
-

UNIT V

FUZZY SET THEORY

Introduction to Fuzzy Set Theory

What is Fuzzy Set Theory?

Fuzzy Set Theory is a mathematical framework used to handle uncertainty and imprecision in decision-making and problem-solving. Unlike traditional "crisp" logic, where things are either **true or false** (black or white), fuzzy logic allows for a **range of possibilities** (shades of gray).

It was introduced by **Lotfi A. Zadeh** in 1965 to model situations where human reasoning is imprecise, such as understanding "**tall**," "**cold**," "**expensive**," or "**fast**."

Why Do We Need Fuzzy Set Theory?

In real life, many things are not just **yes or no**, but somewhere in between.
For example:

- A room temperature of **25°C** may feel "warm" to one person and "cool" to another.
- A student scoring **68%** in a test might be considered "average" by one teacher and "good" by another.

Classical Set Theory (Crisp Logic) says:

✓ If you score **above 50%**, you "Pass."

✗ If you score **below 50%**, you "Fail."

But in reality, a student scoring **49%** is not much different from someone scoring **50%**.
Fuzzy logic handles this **imprecision** by allowing partial membership in sets.

Key Concepts in Fuzzy Set Theory

1. Crisp Sets vs. Fuzzy Sets

- **Crisp Set:** Either an element belongs to a set or it doesn't (strict boundary).
 - Example:
 - If we define a "Tall Person" as **above 6 feet**, then:
 - **6.1 feet** → Tall (✓ Yes)
 - **5.9 feet** → Not Tall (✗ No)
- **Fuzzy Set:** Membership is **gradual**; a person can be "somewhat tall" or "very tall."

2. Membership Function

A **membership function (μ)** represents the degree to which an element belongs to a fuzzy set. It assigns a value between **0 and 1** to each element.

Example: "**Tallness**" of a person

Height (ft)	Membership (Tallness)
5.5	0.2
5.8	0.5
6.0	0.8
6.3	1.0

This means:

- A person **5.5 feet tall** is **20% tall**.
- A person **6.3 feet tall** is **100% tall**.

Unlike crisp logic, where a sharp cutoff exists at **6 feet**, fuzzy logic provides a **smooth transition**.

Real-Life Examples of Fuzzy Set Theory

1. Automatic Air Conditioners

- Traditional thermostats work as:
 - If **temperature > 25°C**, turn AC **ON**.
 - If **temperature < 22°C**, turn AC **OFF**.
 - Fuzzy logic controls AC more efficiently:
 - If **temperature is 23°C** → Run at **30% power**.
 - If **temperature is 27°C** → Run at **80% power**.
 - If **temperature is 30°C** → Run at **100% power**.
 - This results in **smoother cooling** instead of abrupt ON/OFF cycles.
-

2. Washing Machines

- Modern washing machines use fuzzy logic to **adjust wash cycles**.
 - Instead of fixed settings (5 minutes wash, 10 minutes rinse), fuzzy logic considers:
 - **Dirt level**: "Very Dirty," "Moderate," "Slightly Dirty"
 - **Cloth Type**: "Delicate," "Heavy," "Normal"
 - Based on fuzzy input, it adjusts **water level, detergent amount, and spin speed**.
-

3. Automatic Car Braking System

- A traditional system would apply **full brakes** if an object is detected.
 - Fuzzy logic considers:
 - **Speed of car**
 - **Distance from obstacle**
 - **Road conditions (wet/dry)**
 - It **gradually applies brakes** instead of stopping suddenly.
-

Fuzzy vs Crisp

Introduction

In mathematics and logic, sets are used to classify objects based on certain characteristics. Traditional set theory, known as **crisp set theory**, follows strict boundaries—an element either belongs to a set or it does not. However, real-world problems often involve uncertainty and imprecise boundaries. This is where **fuzzy set theory** comes in, allowing for degrees of membership.

Let’s explore the differences between **fuzzy sets** and **crisp sets** in detail.

1. What is a Crisp Set?

A **crisp set** (or classical set) follows a **binary** approach, meaning an element is either in the set (**1 - Yes**) or not in the set (**0 - No**). There is no in-between.

Example of Crisp Set:

Consider a **classroom** where students are classified based on passing or failing an exam.

- If a student scores **above 40 marks**, they are in the **“Pass”** category.
- If a student scores **below 40 marks**, they are in the **“Fail”** category.

Student	Marks Scored	Classification (Crisp Set)
Alice	45	Pass (1)
Bob	38	Fail (0)
Charlie	50	Pass (1)

Problem with Crisp Sets:

- What if a student scores **39.9 marks**? Should they really be classified as **Fail (0)** when someone with 40 marks is **Pass (1)**?
 - Crisp sets do not handle borderline cases well.
-

2. What is a Fuzzy Set?

A **fuzzy set** allows elements to have **partial membership** in a set. Instead of just **0 (No)** or **1 (Yes)**, elements can have a degree of membership between **0 and 1**.

Example of Fuzzy Set:

Revisiting the classroom example, instead of dividing students into just **Pass (1)** or **Fail (0)**, we can define a fuzzy membership function based on their marks.

Fuzzy Membership Function:

- **Marks < 30** → Fail (0)
- **Marks between 30-50** → Partial Membership (e.g., 0.3, 0.5, 0.7)
- **Marks > 50** → Pass (1)

Student	Marks Scored	Membership in "Pass" Set (Fuzzy)
Alice	45	0.8
Bob	38	0.4
Charlie	50	1.0

Advantage of Fuzzy Sets:

- Instead of abruptly saying a student "failed" at 39 marks, we can say they are **40% passed** and **60% failed**.
- This **better represents real-world uncertainty**.

Real-Life Examples of Fuzzy vs. Crisp Logic

Example 1: Weather Classification

- **Crisp Logic:** If the temperature is **above 30°C**, it is **"Hot"**; otherwise, it is **"Cold"**.
- **Fuzzy Logic:** Instead of just "Hot" or "Cold," we can classify temperatures as **"Slightly Hot" (0.6)**, **"Warm" (0.8)**, or **"Very Hot" (1.0)**.

Example:

- $25^{\circ}\text{C} \rightarrow 0.3 \text{ Hot}$
 - $30^{\circ}\text{C} \rightarrow 0.7 \text{ Hot}$
 - $35^{\circ}\text{C} \rightarrow 1.0 \text{ Hot}$
-

Example 2: Air Conditioning System

- **Crisp Control:** If the room temperature is **above 25°C** , turn **ON AC**; otherwise, keep it OFF.
 - **Fuzzy Control:** Adjust the AC **gradually** based on temperature (e.g., increase cooling **proportionally** instead of just ON/OFF).
-

Example 3: Medical Diagnosis

- **Crisp Diagnosis:** A person with **blood sugar $> 140 \text{ mg/dL}$** is classified as **Diabetic (1)**, and those below **140 mg/dL** are **Non-Diabetic (0)**.
- **Fuzzy Diagnosis:** A blood sugar level of **135 mg/dL** might be **0.7 Diabetic** and **0.3 Non-Diabetic**, showing a **gradual transition** rather than an abrupt classification.

Operations and Properties of Crisp Sets

What are Crisp Sets?

A **crisp set** is a collection of objects where each element either **belongs** to the set or **does not belong** to the set. There is no uncertainty or partial membership in crisp sets.

Example of a Crisp Set

Consider a group of students who passed a math exam. If we define a set **P** as "students who passed the exam," a student is either in **P** (if they passed) or not in **P** (if they failed). There is no middle ground.

Operations on Crisp Sets

Just like in regular mathematics, crisp sets follow standard operations. Let's take two sets:

- $A = \{1, 2, 3, 4\}$
- $B = \{3, 4, 5, 6\}$

1. Union ($A \cup B$)

- The union of two sets contains all elements that are in either **A** or **B** or both.
- **Example:** $A \cup B = \{1, 2, 3, 4, 5, 6\}$

✦ Real-life Example:

- If **A** is the set of students who like football and **B** is the set of students who like basketball, then $A \cup B$ represents students who like either football or basketball or both.

2. Intersection ($A \cap B$)

- The intersection of two sets contains only elements that are in **both A** and **B**.
- **Example:** $A \cap B = \{3, 4\}$

✦ Real-life Example:

- If **A** is students who like football and **B** is students who like basketball, then $A \cap B$ represents students who like both sports.

3. Complement (A')

- The complement of a set **A** contains all elements **not** in **A** (within the universal set).
- If the universal set $U = \{1, 2, 3, 4, 5, 6\}$, then $A' = \{5, 6\}$

✦ Real-life Example:

- If **A** is the set of students who passed an exam, **A'** represents students who failed the exam.

4. Difference ($A - B$)

- The difference $A - B$ contains elements that are in **A** but not in **B**.
- **Example:** $A - B = \{1, 2\}$

✦ Real-life Example:

- If **A** is the set of students enrolled in a Science course and **B** is the set of students enrolled in a Math course, then $A - B$ represents students taking Science but not Math.

Properties of Crisp Sets

1. **Idempotent Law:** $A \cup A = A$ and $A \cap A = A$
2. **Commutative Law:** $A \cup B = B \cup A$ and $A \cap B = B \cap A$
3. **Associative Law:** $(A \cup B) \cup C = A \cup (B \cup C)$
4. **Distributive Law:** $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
5. **De Morgan's Law:** $(A \cup B)' = A' \cap B'$ and $(A \cap B)' = A' \cup B'$

Operations and Properties of Fuzzy Sets

Introduction to Fuzzy Sets

In classical set theory (crisp sets), an element either belongs to a set or does not. However, in **fuzzy set theory**, an element can partially belong to a set with a certain degree of membership (ranging from 0 to 1).

For example, in a **crisp set**, if we define a set of "Tall People," a person of height **6 feet** might be included, while someone **5 feet 4 inches** might not. However, in **fuzzy sets**, a person **5 feet 8 inches** might have a membership value of **0.6** in the "Tall People" set, meaning they are somewhat tall.

1. Operations on Fuzzy Sets

Just like classical set theory has operations such as **Union, Intersection, and Complement**, fuzzy sets also have these operations but use different mathematical formulas.

1.1 Fuzzy Union (OR Operation)

- The union of two fuzzy sets **A** and **B** is defined as: $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$ where **max** takes the highest membership value between the two sets.

Example:

Let's define two fuzzy sets representing two concepts:

- Set A: Young People**
 - A 10-year-old has a membership of **1.0**
 - A 20-year-old has a membership of **0.7**
 - A 30-year-old has a membership of **0.4**
- Set B: Middle-Aged People**
 - A 30-year-old has a membership of **0.6**
 - A 40-year-old has a membership of **0.8**
 - A 50-year-old has a membership of **1.0**

The union of these sets (Young OR Middle-Aged) is calculated by taking the **maximum membership** for each age:

- 10-year-old: **$\max(1.0, 0) = 1.0$**
- 20-year-old: **$\max(0.7, 0) = 0.7$**
- 30-year-old: **$\max(0.4, 0.6) = 0.6$**
- 40-year-old: **$\max(0, 0.8) = 0.8$**
- 50-year-old: **$\max(0, 1.0) = 1.0$**

So, this operation combines both sets to form a broader classification.

1.2 Fuzzy Intersection (AND Operation)

- The intersection of two fuzzy sets **A** and **B** is defined as: $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$ where **min** takes the lowest membership value between the two sets.

Example:

Using the same "Young" and "Middle-Aged" sets from above:

- 10-year-old: $\min(1.0, 0) = 0$
- 20-year-old: $\min(0.7, 0) = 0$
- 30-year-old: $\min(0.4, 0.6) = 0.4$
- 40-year-old: $\min(0, 0.8) = 0$
- 50-year-old: $\min(0, 1.0) = 0$

The result shows that **only 30-year-olds have partial membership in both sets.**

1.3 Fuzzy Complement (NOT Operation)

- The complement (or negation) of a fuzzy set **A** is defined as: $\mu_{\neg A}(x) = 1 - \mu_A(x)$ This means the membership value of an element in the complement is the **opposite** of its original membership.

Example:

If **Set A = Young People**:

- 10-year-old: $1 - 1.0 = 0.0$ (Not old at all)
- 20-year-old: $1 - 0.7 = 0.3$ (Still quite young)
- 30-year-old: $1 - 0.4 = 0.6$ (Somewhat young)

The complement set represents "**Not Young People**," meaning people with lower membership in the original set will have higher membership in its complement.

2. Properties of Fuzzy Sets

Fuzzy sets have several important mathematical properties. Here are some key ones:

2.1 Idempotency

- $A \cup A = A$ and $A \cap A = A$
- This means that applying the union or intersection of a set with itself does not change the result.

Example:

If we take the "Young People" fuzzy set and compute **Young OR Young**, it remains the same because nothing new is added.

2.2 Commutativity

- $A \cup B = B \cup A$
- $A \cap B = B \cap A$
- This means that the order of combining two fuzzy sets does not matter.

Example:

Whether we combine "Young People" with "Middle-Aged People" first or the other way around, the result remains the same.

2.3 Associativity

- $(A \cup B) \cup C = A \cup (B \cup C)$
- $(A \cap B) \cap C = A \cap (B \cap C)$
- This means that if we combine three fuzzy sets, it doesn't matter which two we combine first.

Example:

If we have sets for "Young People," "Middle-Aged People," and "Old People," the way we group them for classification doesn't change the final result.

2.4 Distributivity

- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
- This means that **intersection and union** distribute over each other.

Example:

If we have "Tall People" (Set A), "Athletic People" (Set B), and "Healthy People" (Set C), applying fuzzy logic to classify people based on height and fitness would give consistent results no matter the order of operations.

2.5 De Morgan's Laws in Fuzzy Sets

De Morgan's laws for fuzzy sets state:

- $\text{NOT}(A \cup B) = \text{NOT}(A) \cap \text{NOT}(B)$
- $\text{NOT}(A \cap B) = \text{NOT}(A) \cup \text{NOT}(B)$

Example:

If "Young" and "Middle-Aged" are two fuzzy sets, then:

- The **negation of the union** of "Young OR Middle-Aged" is the same as "NOT Young AND NOT Middle-Aged" (i.e., old people).

- The **negation of the intersection** of "Young AND Middle-Aged" is the same as "NOT Young OR NOT Middle-Aged."
-

Real-Life Example: Weather Forecasting

In weather prediction, fuzzy logic is often used to categorize **temperature** into fuzzy sets like "Cold," "Moderate," and "Hot."

- If **temperature = 25°C**, it may belong **50% to Moderate and 20% to Hot**.
- Using fuzzy operations, we can define "Warm Weather" as **Moderate OR Hot**, and "Very Warm Weather" as **Moderate AND Hot**.

This helps meteorologists **predict weather** in a way that considers uncertainty and gradual transitions.

Membership Functions

Introduction to Membership Functions

In classical (crisp) set theory, an element either belongs to a set or it does not. This is known as binary classification. However, in real-life scenarios, many situations cannot be classified as simply "**Yes**" or "**No**". Instead, they exist in a **range** or **degree of belongingness**.

Example:

- A person with a height of **6 feet** is "tall."
- A person with a height of **5 feet** is "short."
- But what about someone who is **5.6 feet**? Are they tall or short?
- Instead of a strict classification, we assign a **degree of membership** to express how tall the person is.

This is where **Fuzzy Set Theory** comes in, and **Membership Functions (MFs)** are used to define how much an element belongs to a fuzzy set.

What is a Membership Function?

A **membership function** (MF) is a curve that defines how each input value is mapped to a membership degree between **0** and **1** in a fuzzy set.

$$\mu_A(x) : X \rightarrow [0, 1]$$

where:

- $\mu_A(x)$ represents the membership function of set A .
 - X is the universe of discourse (all possible values).
 - The output is always between **0** and **1**.
 - $\mu_A(x) = 1 \rightarrow$ Fully belongs to the set.
 - $\mu_A(x) = 0 \rightarrow$ Does not belong at all.
 - $0 < \mu_A(x) < 1 \rightarrow$ Partially belongs to the set.
-

Types of Membership Functions

Different types of membership functions are used depending on the application:

1. Triangular Membership Function

It is the simplest MF, defined by a triangle shape. It is represented as:

$$\mu_A(x) = \begin{cases} 0, & x < a \text{ or } x > c \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \end{cases}$$

◆ Real-Life Example: Air Conditioning System

- Suppose an air conditioner adjusts cooling based on temperature.
- We define a fuzzy set "**Comfortable Temperature**".
- If temperature = **22°C**, membership degree = **1** (fully comfortable).
- If **18°C** or **26°C**, membership = **0** (not comfortable).
- If **20°C** or **24°C**, membership = **0.5** (partially comfortable).

2. Trapezoidal Membership Function

Similar to the triangular function but with a flat top. It allows a range of values to have full membership.

◆ Real-Life Example: *Speed Control in Vehicles*

- Define a fuzzy set "**Safe Speed**" for driving.
- Speed between **40–60 km/h** is fully safe (membership = 1).
- Speed **30 km/h** or **70 km/h** is partly safe (membership = 0.5).
- Speed below **20 km/h** or above **80 km/h** is unsafe (membership = 0).

3. Gaussian Membership Function

It has a smooth bell-shaped curve, often used in precise applications like medical diagnosis.

◆ Real-Life Example: *Medical Diagnosis of Fever*

- Normal body temperature is **37°C** (membership = 1).
- If **38°C**, membership = **0.8** (mild fever).
- If **39°C**, membership = **0.5** (moderate fever).
- If **40°C**, membership = **0.1** (high fever).

4. Sigmoid Membership Function

It has an S-shape and is useful for applications requiring a gradual transition.

◆ Real-Life Example: *Brightness Control in Smart Lights*

- A room with **low ambient light** increases brightness.
- If light intensity = **0%**, membership = **1** (fully dark).
- If **50%**, membership = **0.5** (partially dark).
- If **100%**, membership = **0** (fully bright).

Basic Fuzzy Set Operations

Introduction to Fuzzy Sets

In classical set theory (also called **crisp sets**), an element either belongs to a set or does not. For example, in a set of tall people, a person who is 180 cm tall is included, but a person who is 160 cm tall is excluded. However, real life is not always so clear-cut. A person who is 170 cm tall is **somewhat tall** but not completely tall.

To handle such uncertainty, **fuzzy sets** were introduced by Lotfi Zadeh in 1965. In a fuzzy set, each element has a **membership value** between 0 and 1, indicating how much it belongs to the set.

Basic Fuzzy Set Operations

Fuzzy set operations are similar to classical set operations but work with membership values. The key fuzzy set operations are:

1. **Union (OR Operation)**
2. **Intersection (AND Operation)**
3. **Complement (NOT Operation)**

Each operation modifies the membership function of the fuzzy set.

1. Union (OR Operation)

The union of two fuzzy sets A and B represents elements that belong to either set. The membership value of an element in the union is given by:

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$

Real-Life Example: Weather Classification

Imagine a weather forecasting system where:

- Set A represents "warm weather" (with membership values based on temperature).
- Set B represents "humid weather" (with membership values based on humidity).

A day with 30°C temperature (moderately warm) and 60% humidity (moderately humid) will belong to both sets to some extent. The union of these sets helps determine whether the day is **suitable for outdoor activities**, considering both warmth and humidity.

2. Intersection (AND Operation)

The intersection of two fuzzy sets A and B represents elements that belong to both sets. The membership value is given by:

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$$

Real-Life Example: Employee Performance

A company evaluates employees based on two factors:

- Set A represents "hardworking employees" (based on hours worked).
- Set B represents "efficient employees" (based on tasks completed).

If an employee works 40 hours a week (medium hardworking, $\mu_A=0.7$ and completes 30 tasks (high efficiency, $\mu_B=0.9$ then their **overall performance** (intersection) will be determined by the minimum value:

$$\mu_{A \cap B} = \min(0.7, 0.9) = 0.7$$

This means the employee is considered **moderately good overall**.

3. Complement (NOT Operation)

The complement of a fuzzy set AAA consists of elements that are **not in the set**. The membership function is given by:

$$\mu_{-A}(x) = 1 - \mu_A(x)$$

Real-Life Example: Traffic Congestion

A city defines a fuzzy set AAA for "high traffic congestion" based on vehicle density. If a particular road has a congestion level of $\mu_A=0.8$ (very high traffic), then its complement represents the **free-flowing traffic** level:

$$\mu_{-A} = 1 - 0.8 = 0.2$$

This means the road has only a **20% chance of being uncongested**.

Summary of Basic Fuzzy Set Operations

Operation	Formula	Meaning	Example
Union (OR)	$\max(\mu_A, \mu_B)$	Belongs to either set	Suitable weather based on warmth and humidity
Intersection (AND)	$\min(\mu_A, \mu_B)$	Belongs to both sets	Employee performance based on hard work and efficiency
Complement (NOT)	$1 - \mu_A$	Opposite of set	Traffic congestion vs. free-flowing traffic

Properties of Fuzzy Sets

1. Normality

A fuzzy set A is **normal** if at least one element in the set has a membership value of 1.

$$\exists x \in A \text{ such that } \mu_A(x) = 1$$

✓ Real-Life Example:

In a fuzzy set of "**Very Hot Days**," if one day reaches 45°C, it has a membership value of 1, making the set **normal**.

2. Convexity

A fuzzy set A is convex if for any two elements x_1 and x_2 , the membership of their average is at least as high as the lower of the two.

$$\mu_A\left(\frac{x_1 + x_2}{2}\right) \geq \min(\mu_A(x_1), \mu_A(x_2))$$

✓ Real-Life Example:

In a fuzzy set of "**Medium-Sized Companies**," if a company with 50 employees has a membership of 0.8 and another with 100 employees has a membership of 0.6, then a company with 75 employees should have a membership **greater than or equal to** the minimum of both (0.6).

3. Associativity

For fuzzy union and intersection, the order of grouping does not matter.

$$(A \cup B) \cup C = A \cup (B \cup C)$$

$$(A \cap B) \cap C = A \cap (B \cap C)$$

✓ Real-Life Example:

Grouping in a fuzzy set of "**Popular Movies**" remains the same whether we first combine "Action & Comedy" and then add "Drama," or vice versa.

4. Distributivity

A fuzzy set operation distributes over another, just like in algebra.

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

✓ Real-Life

In "**Favorite Foods**", if someone likes (Pizza OR Pasta) AND Ice Cream, it is the same as liking (Pizza AND Ice Cream) OR (Pasta AND Ice Cream).

Example:

5. Idempotency of Union & Intersection

Applying union or intersection multiple times does not change the result.

$$A \cup A = A$$

$$A \cap A = A$$

✓ Real-Life Example:

If we define a fuzzy set of "**Popular Tourist Destinations**", adding "Paris" multiple times to the list does not increase its membership—it remains a popular destination.

Cartesian Product

Introduction to Cartesian Product

The Cartesian product is a fundamental concept in set theory and is used in both classical (crisp) and fuzzy set theory. It helps in forming ordered pairs or tuples from two or more sets.

In simple terms, the Cartesian product of two sets results in a new set containing all possible ordered pairs formed by taking one element from each set.

Definition

Given two sets **A** and **B**, the Cartesian product of **A** and **B**, denoted as $A \times B$, is defined as:

$$A \times B = \{(a, b) \mid a \in A, b \in B\}$$

This means that every element in **A** is paired with every element in **B** to create ordered pairs.

Example in Classical (Crisp) Set Theory

Example 1: Cartesian Product of Two Sets

Let's say we have two sets:

- $A = \{1, 2\}$
- $B = \{a, b\}$

The Cartesian product $A \times B$ is:

$$A \times B = \{(1, a), (1, b), (2, a), (2, b)\}$$

Here, every element of **A** is combined with every element of **B** to form ordered pairs.

Real-Life Example of Cartesian Product

Example 2: Menu Combinations in a Restaurant

Imagine a fast-food restaurant where customers can choose a meal from two categories:

- Main Course (A):** {Burger, Pizza}
- Drinks (B):** {Coke, Juice}

The Cartesian product of these sets will be:

$$A \times B = \{(\text{Burger, Coke}), (\text{Burger, Juice}), (\text{Pizza, Coke}), (\text{Pizza, Juice})\}$$

So, a customer has four possible meal combinations to choose from.

Cartesian Product in Fuzzy Set Theory

In **fuzzy set theory**, elements have degrees of membership. This means that instead of just having elements in a set (as in crisp sets), each element has a **membership value** between **0** and **1** representing how much it belongs to the set.

Fuzzy Cartesian Product Formula

If A and B are fuzzy sets, their Cartesian product is represented as:

$$(A \times B)(a, b) = \min(\mu_A(a), \mu_B(b))$$

Where:

- $\mu_A(a)$ is the membership value of element a in set A
 - $\mu_B(b)$ is the membership value of element b in set B
 - The function takes the **minimum** of these two membership values
-

Example 3: Fuzzy Cartesian Product in Weather Analysis

Let's consider two fuzzy sets:

- **Temperature (A)**: {Cold (0.8), Warm (0.5)}
- **Humidity (B)**: {High (0.6), Low (0.7)}

Now, the Cartesian product $A \times B$ is computed as follows:

$$(A \times B) = \{(\text{Cold, High, } 0.6), (\text{Cold, Low, } 0.7), (\text{Warm, High, } 0.5), (\text{Warm, Low, } 0.5)\}$$

Here, the membership value for each pair is the **minimum** of the two values.

◆ Interpretation:

- The combination **(Cold, High Humidity)** has a membership value of **0.6**, meaning it is somewhat likely.
- The combination **(Warm, Low Humidity)** has a membership value of **0.5**, meaning it is moderately likely.

Fuzzy Relations

Introduction to Fuzzy Relations

In classical (crisp) set theory, relations describe the connection between elements of two sets. For example, in a set of students and a set of grades, a relation might define which student received which grade. However, in real-world situations, relationships are often not strictly defined. Instead, they have varying degrees of truth. This is where **fuzzy relations** come into play.

Fuzzy relations allow us to **represent relationships that are not just "true" or "false" but rather exist on a spectrum**. This makes them useful for dealing with uncertainty, imprecise data, and human-like reasoning.

Representation of Fuzzy Relations

A **fuzzy relation** is usually represented as a **fuzzy matrix**. Each element in the matrix represents the degree of relationship between two elements from two sets.

Consider two sets:

- **Set A (Students):** {John, Alex, Sara}
- **Set B (Grades):** {A, B, C}

A fuzzy relation can be represented as a matrix:

$$R = \begin{bmatrix} 0.9 & 0.4 & 0.2 \\ 0.7 & 0.8 & 0.3 \\ 0.6 & 0.5 & 0.9 \end{bmatrix}$$

Each value in the matrix represents the **degree of association** between a student and a grade. For example, in the first row:

- **John has a 90% (0.9) chance of getting an A.**
- **John has a 40% (0.4) chance of getting a B.**
- **John has a 20% (0.2) chance of getting a C.**

Instead of making strict classifications, **fuzzy relations allow for partial associations**, which are more realistic.

Types of Fuzzy Relations

1. **Reflexive Fuzzy Relation**
 - A relation R is reflexive if every element is at least somewhat related to itself.
 - Example: A person's similarity to themselves is always **1 (100%)**.
2. **Symmetric Fuzzy Relation**
 - A relation R is symmetric if the degree of relation is the same in both directions.
 - Example: If "John is 70% similar to Alex," then "Alex is also 70% similar to John."

3. Transitive Fuzzy Relation

- A relation RRR is transitive if an indirect relationship exists through a third element.
- Example: If "John is 80% similar to Alex" and "Alex is 60% similar to Sara," then "John should have some level of similarity with Sara."

Real-Life Examples of Fuzzy Relations

Example 1: Online Shopping Recommendations (E-commerce)

- Websites like **Amazon, Flipkart, and eBay** use fuzzy relations to suggest products.
- Suppose you buy a smartphone. The recommendation system checks **fuzzy relations between users and products** to suggest accessories like cases or headphones.
- A fuzzy relation matrix might represent how **strongly related** different products are:

$$R = \begin{bmatrix} 1 & 0.8 & 0.4 \\ 0.8 & 1 & 0.5 \\ 0.4 & 0.5 & 1 \end{bmatrix}$$

Here,

- A phone and a charger have a **strong relation (0.8)**.
- A phone and a case have a **medium relation (0.4)**.
- A charger and a case have a **weak relation (0.5)**.

This allows websites to suggest **related items based on previous purchases** rather than making strict binary decisions.

Example 2: Weather Forecasting

- In traditional weather prediction, temperature and humidity are classified into **fixed categories** (e.g., "hot" or "cold").
- However, real weather conditions exist **on a spectrum**. A temperature of **28°C** might feel "warm" to one person but "hot" to another.
- A fuzzy relation matrix can describe how different weather factors are related:

$$R = \begin{bmatrix} 1 & 0.7 & 0.3 \\ 0.7 & 1 & 0.5 \\ 0.3 & 0.5 & 1 \end{bmatrix}$$

Here,

- Temperature and humidity have a **strong relation (0.7)** because higher humidity often makes temperatures feel hotter.
- Wind speed and temperature have a **weaker relation (0.3)** because strong winds might not always affect temperature significantly.

Fuzzy Cartesian Product

Introduction

In classical (crisp) set theory, the **Cartesian product** of two sets AAA and BBB is the set of all possible ordered pairs (a,b) where a belongs to A and b belongs to B. However, in fuzzy set theory, elements have degrees of membership, so the Cartesian product is modified to account for these membership values.

Definition

The **Fuzzy Cartesian Product** of two fuzzy sets AAA and BBB, represented by their membership functions

$\mu_A(x)$ and $\mu_B(y)$, is given as:

$$\mu_{A \times B}(x, y) = \min(\mu_A(x), \mu_B(y))$$

This means that the membership value of a pair (x, y) in the fuzzy Cartesian product is determined by taking the **minimum** membership value of x in A and y in B .

Real-Life Example: Fuzzy Cartesian Product in Student Performance Evaluation

Let's consider a university that evaluates students based on two fuzzy sets:

1. **Fuzzy set A (Student's Attendance Level)**
 - Low (0.2), Medium (0.6), High (1.0)
2. **Fuzzy set B (Student's Exam Performance)**
 - Poor (0.3), Average (0.7), Excellent (1.0)

Now, we form a fuzzy Cartesian product where each student's performance depends on both attendance and exam scores. The Cartesian product will take the minimum membership value from attendance and performance.

Attendance →	Exam ↓	Low (0.2)	Medium (0.6)	High (1.0)
Poor (0.3)	$\min(0.2, 0.3) = 0.2$	$\min(0.6, 0.3) = 0.3$	$\min(1.0, 0.3) = 0.3$	
Average (0.7)	$\min(0.2, 0.7) = 0.2$	$\min(0.6, 0.7) = 0.6$	$\min(1.0, 0.7) = 0.7$	
Excellent (1.0)	$\min(0.2, 1.0) = 0.2$	$\min(0.6, 1.0) = 0.6$	$\min(1.0, 1.0) = 1.0$	

Thus, fuzzy Cartesian products help in decision-making systems where multiple criteria affect the outcome.

Topic 2: Operations of Fuzzy Relations

Introduction

A **fuzzy relation** is an extension of a classical relation in fuzzy set theory, where relationships between elements are defined with degrees of membership. Fuzzy relations are often represented as matrices with values between 0 and 1, indicating the strength of the relationship.

Basic Operations on Fuzzy Relations

1. Union (OR operation)

The union of two fuzzy relations R_1 and R_2 is given by:

$$\mu_R(x, y) = \max(\mu_{R_1}(x, y), \mu_{R_2}(x, y))$$

1.

- The highest membership value is chosen for each pair.

2. Intersection (AND operation)

The intersection of two fuzzy relations is given by:

$$\mu_R(x, y) = \min(\mu_{R_1}(x, y), \mu_{R_2}(x, y))$$

The lowest membership value is chosen.

3. Composition of Fuzzy Relations

If we have two fuzzy relations, R_1 between sets A and B, and R_2 between sets B and C, then their composition is:

$$\mu_R(a, c) = \max_{b \in B} (\min(\mu_{R_1}(a, b), \mu_{R_2}(b, c)))$$

- This is used in fuzzy logic systems to combine different relationships.

Real-Life Example: Fuzzy Relations in Traffic Control Systems

Consider a **traffic control system** where:

- **Relation R1 (Weather Condition vs. Road Safety Level):**
 - Sunny $\rightarrow$ Safe (1.0), Moderate (0.8), Dangerous (0.2)
 - Rainy $\rightarrow$ Safe (0.3), Moderate (0.7), Dangerous (0.9)
- **Relation R2 (Road Safety Level vs. Speed Limit):**
 - Safe $\rightarrow$ High Speed (1.0), Medium Speed (0.8), Low Speed (0.2)
 - Dangerous $\rightarrow$ High Speed (0.2), Medium Speed (0.5), Low Speed (0.9)

Using **fuzzy composition**, we determine the speed limit for a given weather condition. For example, on a rainy day, since the road is moderately safe (0.7), we combine this with the road safety-to-speed limit relation to suggest a reasonable speed limit.

Thus, **fuzzy relations help in automated decision-making systems like adaptive traffic control.**

UNIT VI

FUZZY SYSTEMS

Introduction to Fuzzy Systems

What is a Fuzzy System?

A **Fuzzy System** is a decision-making system that deals with uncertain, imprecise, or vague information. It is based on **Fuzzy Logic**, which allows computers to make human-like decisions in situations where precise answers are difficult to define.

Unlike traditional binary logic (which uses only 0s and 1s), fuzzy logic allows values between 0 and 1, representing degrees of truth. This makes fuzzy systems highly effective in real-world applications where uncertainty exists.

Why Do We Need Fuzzy Systems?

In many real-life situations, exact or precise values are not always available. Consider the following examples:

1. **Weather Prediction** – Instead of saying "It is exactly 27°C," we often say "It is warm today."
2. **Driving a Car** – Instead of sharp speed limits, drivers often adjust their speed based on conditions like "a little fast" or "too slow."
3. **Washing Machines** – Instead of fixed wash times, modern machines adjust based on how "dirty" the clothes are.

In all these cases, **Fuzzy Systems help computers handle uncertainty** just like humans do.

Basic Components of a Fuzzy System

A fuzzy system consists of four main components:

1. **Fuzzification** – Converts crisp (exact) input data into fuzzy values.
 2. **Fuzzy Rule Base** – A set of "IF-THEN" rules that define how decisions are made.
 3. **Inference Engine** – Processes fuzzy rules to determine the output.
 4. **Defuzzification** – Converts fuzzy output back into a crisp value.
-

Real-Life Example: Fuzzy Logic in Air Conditioners

Modern air conditioners use fuzzy logic to automatically adjust temperature settings based on room conditions.

Scenario:

Imagine you have an AC that adjusts cooling based on how "hot" the room feels rather than a fixed temperature.

1. **Fuzzification:**
 - If the temperature is **28°C**, it is classified as "**warm**" (**0.6**) and "**hot**" (**0.3**) rather than just "hot."
2. **Fuzzy Rule Base:**
 - IF **room is warm**, THEN **set fan speed to medium**.
 - IF **room is hot**, THEN **increase cooling**.
3. **Inference Engine:**
 - Based on the fuzzy rules, the AC decides to **increase cooling slightly**.
4. **Defuzzification:**
 - The fuzzy output (cooling adjustment) is converted into an exact fan speed (e.g., **level 3** instead of just "medium").

This ensures **smooth cooling adjustments** instead of sudden changes, making the AC more energy-efficient and comfortable.

Laws of Propositional Logic

Introduction

Propositional logic is a branch of logic that deals with propositions (statements) that can be either **true** or **false**. In classical logic, these statements follow strict rules. However, in **fuzzy logic**, truth values are not just 0 (false) or 1 (true), but can take any value between 0 and 1.

The **laws of propositional logic** provide a foundation for reasoning in both classical and fuzzy logic. These laws define how logical statements interact and are used in various fields like **artificial intelligence, decision-making systems, and automation**.

Basic Laws of Propositional Logic

1. Law of Identity

- **Statement:** A proposition is always equal to itself.

$$P \equiv P$$

- **Example:**
 - If "It is raining" is **true**, then the same statement remains **true**.

2. Law of Contradiction

- **Statement:** A proposition **cannot** be both true and false at the same time.

$$P \wedge \neg P = 0$$

- **Example:**
 - The statement "The light is ON and OFF at the same time" is contradictory.

3. Law of Excluded Middle

- **Statement:** A proposition is either true or false, with no middle ground in classical logic.

$$P \vee \neg P = 1$$

- **Example:**
 - The statement "The door is open" must be either **true** or **false** (in classical logic).

However, in **fuzzy logic**, the truth of a statement can be **partially true**, such as "The door is **half-open** (0.5 truth value)."

Operations in Propositional Logic

1. Commutative Law

- **Statement:** The order of operations does not affect the result.

$$P \vee Q = Q \vee P$$

$$P \wedge Q = Q \wedge P$$

- **Example:**
 - "It is raining OR it is cold" is the same as "It is cold OR it is raining."

2. Associative Law

- **Statement:** Grouping of statements does not affect the result.

$$(P \vee Q) \vee R = P \vee (Q \vee R)$$

$$(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$$

- **Example:**
 - "Today is Monday OR (it is raining OR it is cold)" is the same as "(Today is Monday OR it is raining) OR it is cold."

3. Distributive Law

- **Statement:** Logical operations distribute over each other, similar to algebra.

$$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$$

$$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$$

- **Example:**
 - "It is sunny AND (hot OR humid)" is the same as "(It is sunny AND hot) OR (It is sunny AND humid)."

4. Idempotent Law

- **Statement:** Repeating a proposition does not change its truth value.

$$P \vee P = P$$

$$P \wedge P = P$$

- **Example:**
 - "It is raining OR it is raining" is just "It is raining."

5. De Morgan's Theorems

- **Statement:** Logical negations distribute over AND/OR operations.

$$\neg(P \wedge Q) = \neg P \vee \neg Q$$

$$\neg(P \vee Q) = \neg P \wedge \neg Q$$

- **Example:**
 - "It is NOT (hot AND humid)" is the same as "It is NOT hot OR it is NOT humid."

Real-Life Example: Smart Home Automation using Propositional Logic

Consider a **smart home system** that automates appliances based on certain conditions.

- **P:** "The room is hot."
- **Q:** "The fan is ON."
- **R:** "The window is open."

Using **propositional logic laws**, we can write rules like:

1. **If the room is hot, the fan must be ON (Implication Law):**

$$P \Rightarrow Q \quad \neg Q \Rightarrow \neg P$$

- If "The room is hot" is **true**, then "The fan is ON" should also be **true**.
2. **If the fan is ON and the window is open, cooling is efficient (Associative Law):**

$$(Q \wedge R) \Rightarrow \text{Cooling Efficient} \quad (Q \wedge (R \wedge S)) \Rightarrow \text{Cooling Efficient}$$

- The system can use this logic to adjust energy usage.
3. **If the fan is ON, the room is NOT hot (De Morgan's Theorem):**

$$\neg(Q \vee P) \Rightarrow P \neg Q \vee \neg P$$

- If the fan is running, the room should **not** remain hot.

Thus, **propositional logic** helps in designing automated systems that take actions based on logical conditions.

Inference in Propositional Logic

Introduction to Propositional Logic

Propositional logic is a branch of logic that deals with statements (propositions) that can either be **true (1)** or **false (0)**. These statements are combined using logical operators such as **AND ($\wedge$)**, **OR ($\vee$)**, **NOT ($\neg$)**, **IF-THEN ($\rightarrow$)**, and **IFF ($\leftrightarrow$)** to form logical expressions.

In classical logic, everything is either completely **true or false** (binary logic). However, in real life, we deal with uncertainties. **Fuzzy logic** extends classical propositional logic by allowing statements to have degrees of truth, ranging from 0 to 1. This makes **fuzzy inference** useful for reasoning in situations where uncertainty is involved.

What is Inference in Propositional Logic?

Inference means drawing conclusions based on given facts (premises). In propositional logic, inference allows us to determine the truth of a statement based on the truth of other related statements.

In **fuzzy logic**, inference is based on **degrees of truth** rather than absolute truth or falsehood. This allows fuzzy systems to handle situations where truth values are not strictly 0 or 1 but can be any value between them.

Basic Logical Operators in Propositional Logic

In both classical and fuzzy logic, inference is made using **logical operators**:

Logical Operation	Classical Logic	Fuzzy Logic
NOT ($\neg A$)	If $A = 1$, then NOT $A = 0$. If $A = 0$, then NOT $A = 1$.	$\mu_{\neg A} = 1 - \mu_A$
AND ($A \wedge B$)	$A \wedge B$ is true only if both A and B are true.	$\mu_{A \wedge B} = \min(\mu_A, \mu_B)$
OR ($A \vee B$)	$A \vee B$ is true if at least one of A or B is true.	$\mu_{A \vee B} = \max(\mu_A, \mu_B)$
IF-THEN ($A \rightarrow B$)	If A is true, then B must be true.	$\mu_{A \rightarrow B} = \max(1 - \mu_A, \mu_B)$

Fuzzy Inference Process

In fuzzy logic, inference is performed using **fuzzy rules** of the form:
"IF condition THEN conclusion"

For example,

- **IF temperature is high THEN fan speed is fast.**
- **IF road is wet THEN reduce speed.**

The fuzzy inference process follows these steps:

1. **Fuzzification**: Convert input values into fuzzy sets.
2. **Rule Evaluation**: Apply fuzzy logic rules.
3. **Aggregation**: Combine results from different rules.
4. **Defuzzification**: Convert fuzzy output into a crisp value.

Real-Life Example: Fuzzy Inference in Weather Prediction

Let's consider a weather forecasting system using fuzzy logic.

1. Defining the Fuzzy Variables

- **Temperature (T)**: Cold (0.1), Warm (0.6), Hot (1.0)
- **Humidity (H)**: Low (0.2), Medium (0.7), High (1.0)
- **Rain Prediction (R)**: No Rain (0.3), Light Rain (0.7), Heavy Rain (1.0)

2. Fuzzy Rules for Weather Inference

- Rule 1: IF temperature is hot AND humidity is high THEN rain is unlikely.
 - $\mu_{\text{Hot}} = 1.0, \mu_{\text{High}} = 1.0$
 - $\mu_{\text{Rain}} = \min(1.0, 1.0) = 1.0 \rightarrow \text{No Rain}$
- Rule 2: IF temperature is warm AND humidity is medium THEN light rain is expected.
 - $\mu_{\text{Warm}} = 0.6, \mu_{\text{Medium}} = 0.7$
 - $\mu_{\text{Rain}} = \min(0.6, 0.7) = 0.6 \rightarrow \text{Light Rain}$
- Rule 3: IF temperature is cold AND humidity is high THEN heavy rain is expected.
 - $\mu_{\text{Cold}} = 0.1, \mu_{\text{High}} = 1.0$
 - $\mu_{\text{Rain}} = \min(0.1, 1.0) = 0.1 \rightarrow \text{No Rain}$

3. Decision Making

The fuzzy system processes the rules and decides the **likelihood of rain** based on the given conditions.

Applications of Fuzzy Inference in Propositional Logic

1. **Medical Diagnosis:**
 - IF symptoms are **high fever** AND **severe cough** THEN **probability of flu is high**.
 - IF symptoms are **mild cough** AND **no fever** THEN **probability of flu is low**.
2. **Automated Driving Systems:**
 - IF road is **slippery** AND **speed is high** THEN **apply brakes gently**.
 - IF road is **dry** AND **speed is normal** THEN **maintain speed**.
3. **Smart Home Systems:**
 - IF room temperature is **high** THEN **increase AC cooling**.
 - IF brightness is **low** THEN **turn on lights**.

Predicate Logic

Introduction

Predicate logic is an extension of propositional logic that allows us to express more complex statements about objects and their relationships. While **propositional logic** deals with simple true/false statements, **predicate logic** introduces **quantifiers and variables**, making it more expressive.

Predicate logic is widely used in **artificial intelligence, database systems, and fuzzy logic applications**, where reasoning with uncertain or imprecise information is necessary.

Components of Predicate Logic

Predicate logic consists of:

1. **Predicates:** Functions that describe a property or a relationship (e.g., "isTall(x)" means "x is tall").
2. **Variables:** Represent objects (e.g., x, y, z).
3. **Quantifiers:** Define the scope of variables:
 - **Universal Quantifier ($\forall$):** Means "for all" (e.g., $\forall x, P(x)$ means "P is true for all x").
 - **Existential Quantifier ($\exists$):** Means "there exists" (e.g., $\exists x, P(x)$ means "There exists an x for which P is true").
4. **Logical Connectives:**
 - **AND ($\wedge$), OR ($\vee$), NOT ($\neg$), IMPLIES ($\rightarrow$)****
 - Used to form compound statements.

Example of Predicate Logic in Real Life: Medical Diagnosis

Let's say we want to represent medical conditions using predicate logic.

- **Predicate:** "HasFever(x)" means "x has a fever."
- **Universal Statement:** $\forall x, HasFever(x) \rightarrow TakeMedicine(x)$
 - Meaning: If any person x has a fever, they should take medicine.
- **Existential Statement:** $\exists x, HasFever(x)$
 - Meaning: There is at least one person who has a fever.

This helps in AI-based **expert systems** used in hospitals, where computers analyze symptoms and suggest treatments.

Fuzzy Predicate Logic

In real life, predicates are not always **crisp (true/false)**. For example:

- "John is tall" might be **somewhat true** rather than fully true or false.
- "The weather is hot" depends on perception.

Fuzzy predicate logic introduces **degrees of truth (between 0 and 1)** rather than just **true/false** values.

Example: Weather Forecasting

Example: Weather Forecasting

Instead of saying:

- $Hot(35^{\circ}C) = 1$ (Completely True)
- $Hot(20^{\circ}C) = 0$ (Completely False)

Fuzzy predicate logic allows:

- $Hot(30^{\circ}C) = 0.8$ (Mostly True)
- $Hot(25^{\circ}C) = 0.5$ (Partially True)

This makes reasoning systems smarter, allowing AI-based **virtual assistants (Alexa, Siri, Google Assistant)** to understand human language better.

Interpretations of Predicate in Fuzzy Logic

Introduction

In classical logic, a **predicate** is a statement that can be either **true** or **false**. However, in **fuzzy logic**, predicates are not strictly true or false; they have degrees of truth. This means that instead of assigning only **0 (false)** or **1 (true)**, fuzzy logic assigns values between **0 and 1** to indicate the **degree of truth**.

What is a Predicate?

A **predicate** is a function or statement that describes some property or condition. In classical logic, it follows the form:

$$P(x) = \text{True or False}$$

where x is a variable.

In **fuzzy logic**, the predicate is interpreted using a **membership function**:

$$\mu_P(x) \in [0, 1]$$

This means that the predicate **P(x)** does not just return true or false but can take any value between **0 and 1**, representing the degree to which x satisfies the condition.

Real-Life Example: Weather Interpretation (Hot or Not?)

Classical Predicate Logic Interpretation:

- Suppose we define the predicate "**The weather is hot**" as:

$$P(\text{Temperature}) = \begin{cases} 1, & \text{if Temperature} \geq 30^{\circ}\text{C} \\ 0, & \text{if Temperature} < 30^{\circ}\text{C} \end{cases}$$

- If it's **30°C or above**, the statement is **true (1)**.
- If it's below **30°C**, the statement is **false (0)**.

Fuzzy Predicate Logic Interpretation:

- In fuzzy logic, the interpretation of **hot weather** is **gradual**:

$$\mu_H(T) = \begin{cases} 0, & \text{if } T \leq 20^{\circ}\text{C} \\ \frac{T-20}{10}, & \text{if } 20^{\circ}\text{C} < T < 30^{\circ}\text{C} \\ 1, & \text{if } T \geq 30^{\circ}\text{C} \end{cases}$$

- **20°C → 0 (Not Hot at all)**
- **25°C → 0.5 (Somewhat Hot)**
- **30°C → 1 (Completely Hot)**

Thus, **fuzzy logic allows for a gradual transition**, making it more realistic.

Fuzzy Interpretation of Predicates in Real Life

1. Grading System (Student Performance)

- Suppose we define a predicate "**Student is intelligent**":
 - **Classical Approach**: If a student scores above 80%, they are intelligent (1); otherwise, they are not (0).
 - **Fuzzy Approach**: A student with 70% may be **partially intelligent (0.7)**, and a student with 50% may be **less intelligent (0.3)**.

2. Speed Control in Cars

- Predicate: "**The car is moving fast.**"
 - **Classical Approach**: If speed > 100 km/h → Fast (1); Otherwise → Not Fast (0).
 - **Fuzzy Approach**: If speed = 70 km/h, it may be **moderately fast (0.5)**.

3. Medical Diagnosis (Blood Sugar Levels)

- Predicate: "**Person is diabetic.**"
 - **Classical Approach**: If blood sugar > 140 mg/dL, the person is diabetic (1); otherwise, they are not (0).
 - **Fuzzy Approach**: A person with **130 mg/dL might have a 0.8 chance of being diabetic**, while **110 mg/dL might be 0.4**.

Logic Formula

Introduction to Logic

Logic is a system of reasoning used to determine truth or falsehood. It is the foundation of artificial intelligence, computer science, and mathematics. Logic is broadly classified into:

- **Classical (Crisp) Logic** (0 or 1, True or False)
- **Fuzzy Logic** (Values between 0 and 1, Partial Truth)

A **logic formula** consists of logical expressions using variables, operators, and rules to define conditions.

Logical Operators in Classical Logic

1. **AND ($\wedge$)** – Both conditions must be true.
 - Example: *You can go outside if it's sunny AND you have permission.*
2. **OR ($\vee$)** – At least one condition must be true.
 - Example: *You will get a reward if you score above 90 OR win the competition.*
3. **NOT ($\neg$)** – Reverses the truth value.
 - Example: *If it's NOT raining, then we will go for a picnic.*

Real-Life Example: Logic in Decision Making

Consider a **school grading system**:

- If a student scores **above 90**, they get an '**A**' grade.
- If a student scores **between 70 and 90**, they get a '**B**' grade.
- If a student scores **below 70**, they get a '**C**' grade.

Using classical logic, we define:

$$\text{Grade} = \begin{cases} A, & \text{if marks} \geq 90 \\ B, & \text{if } 70 \leq \text{marks} < 90 \\ C, & \text{if marks} < 70 \end{cases}$$

However, in **fuzzy logic**, we can assign **partial grades** instead of fixed categories (explained later).

Inference in Predicate Logic

What is Predicate Logic?

Predicate Logic extends basic logic by allowing **variables and quantifiers** to describe more complex statements.

- **Predicate**: A function that expresses a property of an object.
- **Quantifiers**:
 - **Universal ($\forall$)**: True for all cases (e.g., $\forall x, x > 0$ means all x are greater than 0).

- **Existential ($\exists$):** True for at least one case (e.g., $\exists x, x > 10$ means some x is greater than 10).

Inference in Predicate Logic

Inference is the process of deriving new information from existing facts using logical rules. The two main inference methods are:

1. **Modus Ponens (If $A \rightarrow B$ and A is true, then B is true)**
 - Example:
 - If it rains (A), the ground gets wet (B).
 - It rains (A), therefore, the ground gets wet (B).
2. **Modus Tollens (If $A \rightarrow B$ and B is false, then A is false)**
 - Example:
 - If a student studies (A), they pass the exam (B).
 - The student did not pass ($\neg B$), so they did not study ($\neg A$).

Real-Life Example: Medical Diagnosis

A doctor diagnoses a disease using predicate logic:

- If a patient has a fever (P) and cough (Q), then the patient may have the flu (R).
- Given P and Q are true, we infer that R is likely true (flu detected).
- If a new symptom S (loss of taste) is added, inference adjusts to diagnose **COVID-19 instead of flu**.

Predicate logic is widely used in **AI chatbots, medical diagnosis, and search engines**.

Fuzzy Logic

What is Fuzzy Logic?

Fuzzy Logic is an extension of Boolean logic that deals with **partial truth**. Instead of binary values (0 or 1), fuzzy logic allows truth values between **0 and 1**.

Comparison: Classical Logic vs. Fuzzy Logic

Condition	Classical Logic (Binary)	Fuzzy Logic (Partial Truth)
Temperature = Hot	True (1) or False (0)	0.8 (Mostly Hot)
Speed = Fast	True (1) or False (0)	0.6 (Moderately Fast)
Water = Cold	True (1) or False (0)	0.3 (Slightly Cold)

How Fuzzy Logic Works

1. **Fuzzification** – Convert real-world input into fuzzy values (e.g., "temperature is warm" instead of 30°C).
2. **Inference** – Apply fuzzy rules (e.g., "If the room is warm, set AC to medium cooling").
3. **Defuzzification** – Convert fuzzy output back into crisp values (e.g., AC temperature set to 24°C).

Real-Life Example: Automatic Washing Machine

A washing machine adjusts its washing mode based on dirt level and fabric type.

- If clothes are **very dirty**, the machine runs at **high speed (1.0)**.
- If clothes are **slightly dirty**, the machine runs at **medium speed (0.5)**.
- If clothes are **clean**, the machine runs at **low speed (0.2)**.

A fuzzy rule could be:

If dirt level is high (0.8) and fabric is delicate (0.4), then washing speed is medium (0.5))

Fuzzy Quantifiers and Inference

What are Fuzzy Quantifiers?

In classical logic, quantifiers like "**all**," "**some**," or "**none**" are used to describe sets precisely. However, in real life, we often use **vague** quantifiers like:

- "Most students passed the exam."
- "A few cars are on the road."
- "Many people like ice cream."

Fuzzy quantifiers help us **express uncertainty** in a way that a fuzzy system can understand.

Types of Fuzzy Quantifiers

1. **Absolute Quantifiers** (specific amounts)
 - "About 10 people"
 - "Nearly 50%"
2. **Relative Quantifiers** (proportions)
 - "Most," "Few," "Several"

Real-Life Example: Fuzzy Quantifiers in Weather Forecasting

- "There is a **high chance** of rain tomorrow."
- "It will be **slightly cold** today."

In a fuzzy system, "high chance" or "slightly cold" is converted into numerical values using fuzzy logic.

What is Fuzzy Inference?

Fuzzy inference is the process of **drawing conclusions** from fuzzy rules. It is based on **IF-THEN** rules.

Steps in Fuzzy Inference

1. **Fuzzification:** Convert crisp inputs into fuzzy values.
2. **Rule Evaluation:** Apply fuzzy rules (e.g., "If temperature is high, then fan speed is high").
3. **Aggregation:** Combine results of multiple rules.
4. **Defuzzification:** Convert fuzzy results into a crisp value.

Fuzzy Rule-Based System

What is a Fuzzy Rule-Based System?

A **fuzzy rule-based system (FRBS)** is a system that makes decisions based on fuzzy rules.

Structure of an FRBS

1. **Knowledge Base:** Contains fuzzy rules.
2. **Fuzzification Module:** Converts inputs into fuzzy values.
3. **Inference Engine:** Applies fuzzy rules to make decisions.
4. **Defuzzification Module:** Converts fuzzy output into crisp values.

Example: Fuzzy Rule-Based System in Washing Machines

Modern washing machines use fuzzy logic to **adjust water level and washing time**.

- **Rule 1:** If clothes are **very dirty**, then **use more water and more time**.
- **Rule 2:** If clothes are **slightly dirty**, then **use less water and less time**.

By applying fuzzy inference, the washing machine decides how much water and time are needed.

Defuzzification Methods

Defuzzification is the process of converting a **fuzzy output** back into a **crisp (numerical) value**.

Common Defuzzification Methods

1. **Centroid Method:** Calculates the center of gravity of the fuzzy set.
2. **Maximum Membership Method:** Chooses the value with the highest membership.
3. **Average Method:** Takes the average of all possible values.

Example: Defuzzification in Air Conditioners

An air conditioner adjusts cooling based on fuzzy inputs (temperature and humidity).

- Input: "It is **warm** and **slightly humid**."
- Fuzzy Output: "Cooling should be **medium**."
- Defuzzification converts "medium" into a specific temperature (e.g., **24°C**).

Applications of Fuzzy Systems

Fuzzy systems are widely used in real-world applications, including:

1. Industrial Automation

- Used in control systems for **temperature, speed, and pressure regulation**.
- Example: **Cement factories** use fuzzy logic to optimize kiln temperature.

2. Healthcare

- Fuzzy systems help in **diagnosis and treatment**.
- Example: A fuzzy system can decide **insulin dosage** for diabetic patients based on sugar levels.

3. Automotive Systems

- Fuzzy logic is used in **anti-lock braking systems (ABS)** and **cruise control**.
- Example: Cars adjust braking force based on road conditions using fuzzy logic.

4. AI and Robotics

- Fuzzy logic helps **robots navigate and make decisions**.
- Example: A **self-driving car** uses fuzzy rules to determine speed based on traffic conditions.

Cryptography

Unit 1 : Attacks on Computer And Computer Security

1.1] Introduction

- Computer security refers to protecting and securing computers and their related data, networks, software, hardware from unauthorized access, misuse, theft, information loss, and other security issues. The Internet has made our lives easier and has provided us with lots of advantages but it has also put our system's security at risk of being infected by a virus, of being hacked, information theft, damage to the system, and much more.
- Technology is growing day by day and the entire world is in its grasp. We cannot imagine even a day without electronic devices around us. With the use of this growing technology, invaders, hackers and thieves are trying to harm our computer's security for monetary gains, recognition purposes, ransom demands, bullying others, invading into other businesses, organizations, etc. In order to protect our system from all these risks, computer security is important.

Types of computer security

Computer security can be classified into four types:

- 1. Cyber Security:** Cyber security means securing our computers, electronic devices, networks, programs, systems from cyber attacks. Cyber attacks are those attacks that happen when our system is connected to the Internet.
- 2. Information Security:** Information security means protecting our system's information from theft, illegal use and piracy from unauthorized use. Information security has mainly three objectives: confidentiality, integrity, and availability of information.
- 3. Application Security:** Application security means securing our applications and data so that they don't get hacked and also the databases of the applications remain safe and private to the owner itself so that user's data remains confidential.
- 4. Network Security:** Network security means securing a network and protecting the user's information about who is connected through that network. Over the network hackers steal, the packets of data through sniffing and spoofing attacks, man in the middle attack, war driving, etc, and misuse the data for their benefits.

1.2] Need For Security :

- Network security encompasses all the steps taken to protect the integrity of a computer network and the data within it. Network security is important because

it keeps sensitive data safe from cyber attacks and ensures the network is usable and trustworthy.

- Network Security is vital in protecting client data and information, it keeps shared data secure, protects from viruses and helps with network performance by reducing overhead expenses and costly losses from data breaches, and since there will be less downtime from malicious users or viruses, it can save businesses ...
- Security measures, such as surveillance systems, access control, and trained security personnel, provide a sense of safety and personal protection. They act as a deterrent against criminal activities, making individuals feel more secure in public spaces, workplaces, and residential areas.
- Network security encompasses all the steps taken to protect the integrity of a computer network and the data within it. Network security is important because it keeps sensitive data safe from cyber attacks and ensures the network is usable and trustworthy. Successful network security strategies employ multiple security solutions to protect users and organizations from malware and cyber attacks, like distributed denial of service.
- A network is composed of interconnected devices, such as computers, servers and wireless networks. Many of these devices are susceptible to potential attackers. Network security involves the use of a variety of software and hardware tools on a network or as software as a service. Security becomes more important as networks grow more complex and enterprises rely more on their networks and data to conduct business. Security methods must evolve as [threat actors](#) create new attack methods on these increasingly complex networks.

1.3] Security Approaches :

The three approaches to computer security

If we looked at the computer systems and how they try to provide security, I think we could categorize those attempts into three broad categories:

- 1) Security by Correctness
- 2) Security by Isolation
- 3) Security by Obscurity

Let's discuss those categories in more detail below.

1] Security by Correctness

The assumption here is obvious: if we can produce software that doesn't have bugs (nor any maliciously behaving code), then we don't have security problems at all. The only problem is that we don't have any tools to make sure that a given code is correct (in terms of implementation, design and ethical behavior). But if we look at various efforts in computer science, we will notice a lot of effort has been made to achieve Security by Correctness: "safe" languages, code verifiers (although not sound ones, just heuristic based), developer's education, manual code audit, etc. Microsoft's famed Secure Development Life-cycle is all about Security by Correctness. The only problem is: all those approaches sometimes work and sometimes do not, sometimes they miss some bug and also there are problems that I simply don't believe can be addressed by automatic code verifiers or even safe languages, like e.g. logic/design bugs or deciding on whether a given code behaves maliciously or not (after all this is an ethical problem in many cases, not a computer science problem).

To sum it: I think that in some more or less distant future (some people think about a timeframe of 50 years or so), we would get rid of all the implementation bugs, thanks to safe languages and/or sound code verifiers. But I don't believe we could assure correctness of software on any higher level of abstraction than implementation level.

2] Security by Isolation

Because of the problems with effectively implementing Security by Correctness approach, people, from the very beginning, has also taken another approach, which is based on isolation. The idea is to split a computer system into smaller pieces and make sure that each piece is separated from the other ones, so that if it gets compromised/malfunctions, then it cannot affect the other entities in the system. Early UNIX's user accounts and separate process address spaces, things that are now present in every modern OS, are examples of Security by Isolation.

Simple as it sounds, in practice the isolation approach turned out to be very tricky to implement. One problem is how to partition the system into meaningful pieces and how to set permissions for each piece. The other problem is implementation - e.g. if we take a contemporary consumer OS, like Vista, Linux or Mac OSX, all of them have monolithic kernels, meaning that a simple bug in any of the kernel components (think: hundreds of 3rd party drivers running there), allows to bypass the isolation mechanisms provided by the kernel to the rest of the system (process separation, ACLs, etc).

Obviously the problem is because the kernels are monolithic. Why not implement Security by Isolation on a kernel level then? Well, I would personally love that approach, but the industry simply took another course and decided that monolithic kernels are better than micro-kernels, because it's easier to write the code for them and (arguably) they offer better performance.

Many believe, including myself, that this landscape can be changed by the virtualization technology. Thin bare-metal hypervisor, like e.g. Xen, can act like a micro kernel and enforce isolation between other components in the system - e.g. we can move drivers into a separate

domain and isolate them from the rest of the system. But again there are challenges here on both the design- as well as the implementation-level. For example, we should not put all the drivers into the same domain, as this would provide little improvement in security. Also, how to make sure that the hypervisor itself is not buggy?

3] Security by Obscurity (or Security by Randomization)

Finally we have the Security by Obscurity approach that is based on the assumption that we cannot get rid of all the bugs (like in Security by Isolation approach), but at least we can make exploitation of those bugs very hard. So, it's all about making our system unfriendly to the attacker.

Examples of this approach include Address Space Layout Randomization (ASLR, present in all newer OSes, like Linux, Vista, OSX), StackGuard-like protections (again used by most contemporary OSes), pointer encryption (Windows and Linux) and probably some other mechanisms that I can't remember at the moment. Probably the most extreme example of Security by Obscurity would be to use a compiler that generates heavily obfuscated binaries from the source code and creates a unique (on a binary level) instances of the same system. Alex did his PhD on this topic and his an expert on compilers and obfuscators.

The obvious disadvantage of this approach is that it doesn't prevent the bugs from being exploited - it only make the meaningful exploitation very hard or even impossible. But if one is concerned also about e.g. DoS attacks, then Security by Obscurity will not prevent them in most cases. The other problem with obfuscating the code is the performance (compiler cannot optimize the code for speed) and maintenance (if we got a crash dump on an "obfuscated" Windows box, we couldn't count on help from the technical support). Finally there is a problem of proving that the whole scheme is correct and that our obfuscator (or e.g. ASLR engine) doesn't introduce bugs to the generated code and that we will not get random crashes later (that we would be most likely unable to debug, as the code will be obfuscated).

1.4] Principle Of Security :

1. Confidentiality

- The confidentiality principle of security states that only their intended sender and receiver should be able to access messages, if an unauthorized person gets access to this message then the confidentiality gets compromised.
- For example, suppose user X wants to send a message to user Y, and X does not want some else to get access to this message, or if it gets access, he/she does not come to

know about the details. But if user Z somehow gets access to this secret message, which is not desired, then the purpose of this confidentiality gets fail. This leads to the interception. i.e. if user Z access the secret message or email sent by user X to Y without permission of X and Y, then it is called an interception. Interception causes loss of message confidentiality.

2. Authentication

- The authentication principle of security establishes proof of identity, it ensures that the origin of a document or electronic message is correctly identified.
- For example suppose user Z sends a message to user Y, however, the trouble is that user Z posed as user X while sending a message to user Y. How would user Y know that message comes from Z, not X. This leads to the fabrication attack. For example
- The attacker can act as user X and sends fund transfer request(from X' account to attacker account) to a bank, and the bank will transfer the amount as requested from X's account to attacker, as banks think fund transfer request comes from user X. Fabrication is possible in absence of proper authentication mechanism.

3. Integrity

- The integrity principle of security states that the message should not be altered. In other words, we can say that, when the content of the message changes after the sender sends it, but before it reaches the intended receiver, we can say that integrity of the message is lost.
- For example, suppose user X sends a message to User Y, and attacker Z somehow gets access to this message during transmission and changes the content of the

message and then sends it to user Y. User Y and User X does not have any knowledge that the content of the message was changed after user X send it to Y. This leads to a modification. Modification causes loss of message integrity.

4. Non-repudiation

- Non-repudiation principle of security does not allow the sender of a message to refute the claim of not sending that message. There are some situations where the user sends a message and later on refuses that he/she had sent that message.
- For example, user X sends requests to the bank for fund transfer over the internet. After the bank performs fund transfer based on user X request, User X cannot claim that he/she never sent the fund transfer request to the bank. This principle of security defeats such possibilities of denying something after having done it.

5. Access control

- Access control principles of security determine who should be able to access what. i.e. we can specify that what users can access which functions,
- for example, we can specify that user X can view the database record but cannot update them, but user Y can access both, can view record, and can update them. This principle is broadly related to two areas – role management and rule management where role management concentrates on the user side. i.e. which user can do what and rule management concentrate on the resources side i.e. which resource is available. Based on this matrix is prepared, which lists the user against q list of items they can access. The access control list is a subset of the access control matrix.

6. Availability

- The availability principle of security states that resources should be available to the authorized person at all times.
- For example, because of the intentional action of another unauthorized user Z, an authorized user x may not be able to contact server Y, this leads to an interruption attack, interruption puts the availability of resources in danger. A real-life example of this could be, suppose attacker or unauthorized person Z tries to access the FB Account of user X, as User Z does not know the password of user X, he/she tries to log in to the X's account using a random password. after attempting maxim limit for the password, if it is not correct then X's account will be blocked, therefore because of unauthorized person Z, user X could not access his account.

7. Ethical and legal issues

Ethical issues in the security system are classified into the following categories

- **Privacy:** It deals with the individual's right to access the personal information
- **Accuracy:** It deals with the responsibility of authentication, fidelity, and accuracy of information
- **Property:** It deals with the owner of the information
- **Accessibility:** It deals with what information does an organization has the right to collect.

while dealing with legal issues, we must remember that there is a hierarchy of regulatory bodies that govern the legality of information security, it can be classified into the following categories

- International
- Federal

- State
- Organization

1.5] Type Of Attacks

Attacks are typically categorized based on the action performed by the attacker. An attack, thus, can be **passive** or **active**.

A] Passive Attacks

The main goal of a passive attack is to obtain **unauthorized access to the information**. For example, actions such as intercepting and eavesdropping on the communication channel can be regarded as passive attack.

These actions are passive in nature, as they neither affect information nor disrupt the communication channel. A passive attack is often seen as *stealing* information. The only difference in stealing physical goods and stealing information is that theft of data still leaves the owner in possession of that data. Passive information attack is thus more dangerous than stealing of goods, as information theft may go unnoticed by the owner.

B] Active Attacks

An active attack involves changing the information in some way by conducting some process on the information. For example,

- Modifying the information in an unauthorized manner.
 - Initiating unintended or unauthorized transmission of information.
 - Alteration of authentication data such as originator name or timestamp associated with information
 - Unauthorized deletion of data.
 - Denial of access to information for legitimate users (denial of service).
-
- Cryptography provides many tools and techniques for implementing cryptosystems capable of preventing most of the attacks described above.

1. A] Cryptography

A.1] Concept And Techniques Introduction :

- *The art and science of concealing the messages to introduce secrecy in information security is recognized as cryptography.*
 - The word 'cryptography' was coined by combining two Greek words, 'Krypto' meaning hidden and 'graphene' meaning writing.
 - **Cryptography** is the study and practice of techniques for secure communication in the presence of third parties called adversaries. It deals with developing and analyzing protocols that prevents malicious third parties from retrieving information being shared between two entities thereby following the various aspects of information security.
 - Secure Communication refers to the scenario where the message or data shared between two parties can't be accessed by an adversary. In Cryptography, an Adversary is a malicious entity, which aims to retrieve precious information or data thereby undermining the principles of information security.
 - Data Confidentiality, Data Integrity, Authentication and Non-repudiation are core principles of modern-day cryptography.
1. **Confidentiality** refers to certain rules and guidelines usually executed under confidentiality agreements which ensure that the information is restricted to certain people or places.
 2. **Data integrity** refers to maintaining and making sure that the data stays accurate and consistent over its entire life cycle.
 3. **Authentication** is the process of making sure that the piece of data being claimed by the user belongs to it.
 4. **Non-repudiation** refers to the ability to make sure that a person or a party associated with a contract or a communication cannot deny the authenticity of their signature over their document or the sending of a message.

1.A.2] Plain Text And Cypher Text

❖ What is Plaintext?

In cryptography, Plaintext is usually plain readable text before it is encrypted into ciphertext or readable text after it is decrypted. Any message, document, file, or the like that is not meant to be encrypted is referred to as plaintext.

- The input to a cryptosystem is plaintext, and the output is ciphertext. Algorithms in cryptography convert plaintext to ciphertext and ciphertext to plaintext.

- Plaintext saved in a computer file must be protected since its contents are completely disclosed and hence potentially actionable if it is stolen, disclosed, or sent without permission. If data is to be saved, the storage medium, device, components, and backups must all be protected.

Examples of Plaintext

Plaintext is recommended in most applications. Plaintext should be shown in a browser, word processor, or email client, for example. However, early Internet network protocols occasionally transmitted user ID and password combinations in plaintext. This is a bad security practice since it exposes user credentials for systems that are accessible remotely over a public network such as the Internet.

- Although the Password Authentication System provided a two-way handshake authentication exchange protocol, it did not include a mechanism for encrypting credentials.
- Those credentials are exposed if plaintext passwords are written in application configuration files. Developers that keep plaintext passwords in their source code are less likely to expose their credentials.
- Password protection in PowerShell scripts to avoid revealing those passwords in their scripts, developers must take care.

Plaintext Attack :

- The **known-plaintext attack (KPA)** is a cryptanalysis attack paradigm in which the attacker has both the plaintext (also known as a crib) and its encrypted form (ciphertext). These can be used to uncover more secrets, such as secret keys and codebooks.
- Known-plaintext attacks are common against traditional ciphers. A Caesar cipher, for example, can be decrypted entirely using a single letter of corresponding plaintext and ciphertext. If there are less than 26 unique pairs, a universal monoalphabetic substitution cipher requires multiple character pairs and some guessing.

❖ Defining Ciphertext

What is ciphertext in cryptography?

Ciphertext is encrypted text transformed from plaintext using an encryption algorithm. Ciphertext can't be read until it has been converted into plaintext (decrypted) with a key. The decryption cipher is an algorithm that transforms the ciphertext back into plaintext.

- The result of employing encryption methods, often referred to as cyphers, is called ciphertext. When data cannot be understood by individuals or devices lacking the appropriate cypher, it is considered encrypted. To interpret the data, the cypher is necessary. Algorithms transform plaintext into ciphertext, and vice versa, to convert ciphertext back into plaintext. These processes are known as encryption and decryption.
- Ciphertext, represents a cryptographic approach in which an algorithm utilizes substitutions instead of original plaintext elements. Substitution ciphers replace individual letters, letter pairs, letter triplets, or various combinations of these while preserving the initial sequence. Single-letter substitutions are utilized in simple substitution cyphers, while polygraphed cyphers involve larger letter groupings.
- In simpler terms, letters are substituted for other letters. In the past, recording corresponding characters to decipher a message was feasible.

For example, "GOOD DOG" can be encrypted as "PLLX XLP" where "L" substitutes for "O", "P" for "G", and "X" for "D" in the message. Transposition of the letters "GOOD DOG" can result in "DGOGD00". These simple ciphers and examples are easy to crack, even without plaintext-ciphertext pairs.

What is the formula for ciphertext?

The conversion formula is of the form $c \equiv p + a \pmod{26}$. We know that when $p = 5$ (plaintext E), we have $c = 10$ (ciphertext J). Thus, $10 \equiv 5 + a \pmod{26}$. So $a \equiv 5 \pmod{26}$, and the encryption formula is $c \equiv p + 5 \pmod{26}$.

1.A.3] Substitution And Transposition Technique

What is Substitution Cipher Technique?

In the Substitution Cipher technique, each character is substituted with other character/number or other symbol. This technique changes the identity of a character but not its position in the string.

A substitution cipher encrypts a text sequence by replacing letters or units of text with other letters or units of text. Substitution ciphers are an early form of cryptography that predates the development of computers and are now largely out-dated.

Like B or C, a letter is converted into another letter in a substitution cipher, encrypting the sequence for a human reader. The problem is that basic substitution ciphers do not encrypt adequately in terms of computer assessment.

Substitution ciphers became relatively easy for computers to crack with the emergence of the personal computer. However, some of the substitution cipher's concepts have persisted, for

example, some versions of current encryption may encrypt data using an extensive text set and a very clever substitution.

What is Transposition Cipher Technique?

In the Transposition Cipher Technique, each character's position is shifted to a different position. A transposition cipher is an encryption method in which plaintext units (usually letters or groups of characters) are shifted in a predictable way resulting in the ciphertext being a permutation of the plaintext. That is, the units' order is altered (the plaintext is reordered).

To encrypt, a objective function is applied to the locations of the characters, and to decrypt, an inverse function is used.

The Rail Fence encryption is a type of transposition cipher named after how it is encoded. The plaintext is written on successive "rails" of an imaginary fence in the rail fence, downwards and diagonally, then moved up when we reach the bottom. The message is then read aloud in a series of rows.

The Rail Fence Cipher follows a scytale-like pattern, an old Greek mechanical device for constructing a transposition cipher. A cylinder and a ribbon wrapped around the cylinder comprised the system. The coiled ribbon was used to write the encrypted message. When the ribbon was uncoiled from the cylinder, the letters of the original message were rearranged. The message was decoded when the ribbon recoiled on a cylinder with the same diameter as the encrypting cylinder.

Difference between Substitution Cipher and Transposition Cipher

The following table highlights the important differences between Substitution Cipher and Transposition Cipher.

Key	Substitution Cipher Technique	Transposition Cipher Technique
Algorithm	Each character is replaced with a different character, integer, or symbol.	Each character has been repositioned from its original place.

Key	Substitution Cipher Technique	Transposition Cipher Technique
Forms	It comes in two forms: Mono-Alphabetic Substitution Cipher and Poly-Alphabetic Substitution Cipher.	It has two forms: Key-less transposition cipher and keyed transposition cipher.
Change	Character identity is changed but position remains same.	Character position is changed but identity remains same.
Detection	A letter less frequently used can be easily traced.	A letter near to original position can be easily traced.
Example	Caesar Cipher is an example of Substitution Cipher.	Reil Fence Cipher is an example

1.A.4] Encryption And Decryption

- The process of turning plain text into an encrypted form (cipher text) is known as encryption. Sensitive data is transmitted in an encrypted form so that it can be protected and a strong encryption mechanism ensures that the data is not misused even if a hacker gets hold of it. Decryption is the reverse mechanism where the encrypted cipher text is converted back into its original form.

What is Encryption?

Encryption is the procedure of encoding data in cryptography. This procedure turns plaintext, or the original representation of the data, into cipher text or an alternate representation of the data. Only authorized parties should be able to decrypt a cipher text back to plaintext and access the original data. Encryption does not prevent interference in and of itself, but it does hinder a would-be interceptor from understanding the information.

An encryption system generally employs a pseudo-random encryption key produced by an algorithm for technical reasons. Although it is possible to decode a message without knowing the

key, a well-designed encryption system requires significant computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users.

- Encrypted data typically appears as a long string of random letters and numbers.
- Once the information has been encrypted, the only way to decrypt it and make it readable again is to use the correct encryption key. Encryption is necessary for the secure transmission and storage of sensitive data.
- Stream ciphers encrypt data one bit or byte at a time, making them ideal for real-time communications. Before encrypting data, block ciphers divide it into larger chunks, usually 64 bits.

What is Decryption?

Decryption is the procedure of restoring encrypted data to its original state. In most cases, it's the reverse mechanism of the encryption process. Because decryption needs a secret key or password, it decodes the encrypted information so that only an authorized user may decrypt the data.

As information flows across the Internet, it becomes vulnerable to unauthorized persons or groups scrutinizing and accessing it. As a consequence, data is encrypted in order to prevent data theft and loss. Email communications, text files, pictures, user data, and directories are just a few of the objects that are commonly encrypted. The person in charge of decryption receives a popup or window asking for a password to access the encrypted data.

Difference between Encryption and Decryption

The following table highlights the major differences between encryption and decryption –

Key	Encryption	Decryption
Definition	Encryption is a process of converting a plain text into an encrypted or cipher text.	Decryption is a process of converting the encrypted or cipher text into plain text.
Place of Occurrence	Encryption takes place at the sender's end.	Decryption is done at the receiver's end.
Process	A secret key or a public key must be used to encrypt	Any communication that requires the use of a secret

Key	Encryption	Decryption
	every message.	key or private key to decode.
Actor	After encrypting the data with a secret key or a public key, the sender delivers it to the recipient.	The receiver gets the encrypted data and uses the secret key or private key to decode it.

1.A.5] Symmetric Key Cryptography

In symmetric key cryptography, an individual key is used for both encryption and decryption. The sender needs the key to encrypt the plaintext and sends the cipher document to the receiver. The receiver used the similar key (or rule set) to decrypt the message and recover the plaintext. Because an individual key is used for both functions, symmetric key cryptography is also known as symmetric encryption.

Symmetric key cryptography schemes are usually categorized such as stream ciphers or block ciphers. Stream ciphers works on a single bit (byte or computer word) at a time and execute some form of feedback structure so that the key is constantly changing.

❖ Asymmetric cryptography

Asymmetric cryptography uses two keys for encryption and decryption. It depends on the technique of public and private keys. A public key, which is interchanged between higher than one user. Data is decrypted by a private key, which is not transformed. It is slower but more secure. The public key used in this encryption technique is applicable to everyone, but the private key used in it is not revealed.

In asymmetric encryption, a message that is encrypted utilizing a public key can be decrypted by a private key, while if the message is encrypted by a private key can be decrypted by utilizing the public key. Asymmetric encryption is broadly used in dayto- day communication channels, particularly on the internet.

Symmetric Key Cryptography

There is only one key (symmetric key) is used, and the similar key can be used to encrypt and decrypt the message.

It is effective as this technique is recommended for high amounts of text.

Symmetric encryption is generally used to transmit bulk information.

Symmetric key cryptography is also known as secret-key cryptography or private key cryptography.

Symmetric key cryptography uses fewer resources as compared to asymmetric key cryptography.

The length of the keys used is frequently 128 or 256 bits, based on the security need.

Asymmetric Key Cryptography

There are two different cryptographic keys (asymmetric keys), known as the public and the private keys, are used for encryption and decryption.

It is inefficient as this approach is used only for short messages.

It is generally used in smaller transactions. It is used for making a secure connection channel before transferring the actual information.

Asymmetric key cryptography is also known as public-key cryptography or a conventional cryptographic system.

Asymmetric key cryptography uses more resources as compared to symmetric key cryptography.

The length of the keys is much higher, such as the recommended RSA key size is 2048 bits or higher.

1.A.6] Steganography

Steganography is the art and science of embedding secret messages in a cover message in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message

The diagram below depicts a basic steganography model.

As the image depicts, both cover file(X) and secret message(M) are fed into steganographic encoder as input. Steganographic Encoder function, $f(X,M,K)$ embeds the secret message into a cover file. Resulting Stego Object looks very similar to your cover file, with no visible changes. This completes encoding. To retrieve the secret message, Stego Object is fed into Steganographic Decoder.

Steganography: Historical Background

Steganography is the practice of concealing a secret message behind a normal message. It stems from two Greek words, which are *steganos*, means covered and *graphia*, means writing. Steganography is an ancient practice, being practiced in various forms for thousands of years to keep communications private. For Example:

- The first use of steganography can be traced back to 440 BC when ancient Greece, people wrote messages on wood and covered it with wax, that acted as a covering medium
- Romans used various forms of Invisible Inks, to decipher those hidden messages light or heat were used
- During World War II the Germans introduced microdots, which were complete documents, pictures, and plans reduced in size to the size of a dot and were attached to normal paperwork
- Null Ciphers were also used to hide unencrypted secret messages in an innocent looking normal message

How is Steganography different from Cryptography?

[Cryptography](#) changes the information to ciphertext which cannot be understood without a decryption key. So, if someone were to intercept this encrypted message, they could easily see that some form of [encryption](#) had been applied. On the other hand, steganography does not change the format of the information but it conceals the existence of the message.

	STEGANOGRAPHY	CRYPTOGRAPHY
Definition	It is a technique to hide the existence of communication	It's a technique to convert data into an incomprehensible form
Purpose	Keep communication secure	Provide data protection
Data Visibility	Never	Always
Data Structure	Doesn't alter the overall structure of data	Alters the overall structure of data
Key	Optional, but offers more security if used	Necessary requirement
Failure	Once the presence of a secret message is discovered, anyone can use the secret data	If you possess the decryption key, then you can figure out original message the ciphertext

1.A.7] Key Range And Key Size

Key Size :

- In cryptography, key size or key length is the size (measured in bits or bytes) of the key used in a cryptographic algorithm (such as a cipher). Typical key sizes in modern symmetric ciphers are 128, 192, and 256 bits. Older symmetric ciphers used only 40, 56, or 64 bits, which can be broken by brute force.
- The longer the key, the more secure data can be viewed with it. The key length is an encryption security standard and varies by the protocol used and the algorithm.

Key Range :

Key range" refers to the set of all possible keys that can be used in cryptography. The range is determined by the key size and the underlying algorithm, and it affects the security of the encryption.

KEY RANGE AND KEY SIZE

- The encryption/decryption algorithm
- The encrypted message
- Knowledge about the key size (e.g. the value of the key is a number between 0 and 100 billion).

For example consider the brute force attack here, which works on the principle of trying every possible key in the key range, until you get the right key. Brute force attack

Understanding key range With every incremental bit, the attacker has to perform double the

number of operations as compared to the previous key size. It is found that for a 56-bit key, 41 it takes 1 second to search 1 percent of the key range. Taking this argument further, it takes about 1 minute to search about half of the key range (which is what is required, on an average, to crack a key). Using this as the basis, let us have a look at the similar values (time required for a search of 1 percent and 50 percent of the key space) for various key sizes. This is shown in Table We can represent the possible values in the key range using hexadecimal notation and see visually how an increase in the key size increases the key range and therefore, the complexity for an attacker. Key sizes and ranges

1.A.8] POSSIBLE TYPES OF ATTACKS

When the sender of a message encrypts a plain text message into its corresponding cipher text, there are five possibilities for an attack on this message.

- **Cipher text only attack:** In this type of attack, the attacker does not have any clue about the plaintext and has some or all of the cipher text. The attacker analyzes the cipher text at leisure to try and figure out the original plain text.
- **Known plain text attack:** In this case, the attacker knows about some pairs of plain text and corresponding cipher text for those pairs. Using this information, the attacker tries to find other pairs and therefore, know more and more of the plain text. Examples of such known plain texts are company banners, file headers, etc. which are found commonly in all the documents of a particular company.
- **Chosen plain text attack:** Here, the attacker selects a plain text block and tries to look for the encryption of the same in the cipher text. Here, the attacker is able to choose the messages to encrypt. Based on this, the attacker intentionally Security, picks patterns of cipher text that result in obtaining more information about the key.
- **Chosen cipher text attack:** In the chosen cipher text attack, the attacker knows the cipher text to be decrypted, the encryption algorithm that was used to produce this cipher text and the

corresponding plain text block. The attacker's job is to discover the key used for encryption.

Chosen text attack: The chosen text attack is essentially a combination of chosen plain text attack and chosen cipher text attack

Cryptography

Unit 1 : Attacks on Computer And Computer Security

1.1] Introduction

- Computer security refers to protecting and securing computers and their related data, networks, software, hardware from unauthorized access, misuse, theft, information loss, and other security issues. The Internet has made our lives easier and has provided us with lots of advantages but it has also put our system's security at risk of being infected by a virus, of being hacked, information theft, damage to the system, and much more.
- Technology is growing day by day and the entire world is in its grasp. We cannot imagine even a day without electronic devices around us. With the use of this growing technology, invaders, hackers and thieves are trying to harm our computer's security for monetary gains, recognition purposes, ransom demands, bullying others, invading into other businesses, organizations, etc. In order to protect our system from all these risks, computer security is important.

Types of computer security

Computer security can be classified into four types:

- 1. Cyber Security:** Cyber security means securing our computers, electronic devices, networks, programs, systems from cyber attacks. Cyber attacks are those attacks that happen when our system is connected to the Internet.
- 2. Information Security:** Information security means protecting our system's information from theft, illegal use and piracy from unauthorized use. Information security has mainly three objectives: confidentiality, integrity, and availability of information.
- 3. Application Security:** Application security means securing our applications and data so that they don't get hacked and also the databases of the applications remain safe and private to the owner itself so that user's data remains confidential.
- 4. Network Security:** Network security means securing a network and protecting the user's information about who is connected through that network. Over the network hackers steal, the packets of data through sniffing and spoofing attacks, man in the middle attack, war driving, etc, and misuse the data for their benefits.

1.2] Need For Security :

- Network security encompasses all the steps taken to protect the integrity of a computer network and the data within it. Network security is important because

it keeps sensitive data safe from cyber attacks and ensures the network is usable and trustworthy.

- Network Security is vital in protecting client data and information, it keeps shared data secure, protects from viruses and helps with network performance by reducing overhead expenses and costly losses from data breaches, and since there will be less downtime from malicious users or viruses, it can save businesses ...
- Security measures, such as surveillance systems, access control, and trained security personnel, provide a sense of safety and personal protection. They act as a deterrent against criminal activities, making individuals feel more secure in public spaces, workplaces, and residential areas.
- Network security encompasses all the steps taken to protect the integrity of a computer network and the data within it. Network security is important because it keeps sensitive data safe from cyber attacks and ensures the network is usable and trustworthy. Successful network security strategies employ multiple security solutions to protect users and organizations from malware and cyber attacks, like distributed denial of service.
- A network is composed of interconnected devices, such as computers, servers and wireless networks. Many of these devices are susceptible to potential attackers. Network security involves the use of a variety of software and hardware tools on a network or as software as a service. Security becomes more important as networks grow more complex and enterprises rely more on their networks and data to conduct business. Security methods must evolve as [threat actors](#) create new attack methods on these increasingly complex networks.

1.3] Security Approaches :

The three approaches to computer security

If we looked at the computer systems and how they try to provide security, I think we could categorize those attempts into three broad categories:

- 1) Security by Correctness
- 2) Security by Isolation
- 3) Security by Obscurity

Let's discuss those categories in more detail below.

1] Security by Correctness

The assumption here is obvious: if we can produce software that doesn't have bugs (nor any maliciously behaving code), then we don't have security problems at all. The only problem is that we don't have any tools to make sure that a given code is correct (in terms of implementation, design and ethical behavior). But if we look at various efforts in computer science, we will notice a lot of effort has been made to achieve Security by Correctness: "safe" languages, code verifiers (although not sound ones, just heuristic based), developer's education, manual code audit, etc. Microsoft's famed Secure Development Life-cycle is all about Security by Correctness. The only problem is: all those approaches sometimes work and sometimes do not, sometimes they miss some bug and also there are problems that I simply don't believe can be addressed by automatic code verifiers or even safe languages, like e.g. logic/design bugs or deciding on whether a given code behaves maliciously or not (after all this is an ethical problem in many cases, not a computer science problem).

To sum it: I think that in some more or less distant future (some people think about a timeframe of 50 years or so), we would get rid of all the implementation bugs, thanks to safe languages and/or sound code verifiers. But I don't believe we could assure correctness of software on any higher level of abstraction than implementation level.

2] Security by Isolation

Because of the problems with effectively implementing Security by Correctness approach, people, from the very beginning, has also taken another approach, which is based on isolation. The idea is to split a computer system into smaller pieces and make sure that each piece is separated from the other ones, so that if it gets compromised/malfunctions, then it cannot affect the other entities in the system. Early UNIX's user accounts and separate process address spaces, things that are now present in every modern OS, are examples of Security by Isolation.

Simple as it sounds, in practice the isolation approach turned out to be very tricky to implement. One problem is how to partition the system into meaningful pieces and how to set permissions for each piece. The other problem is implementation - e.g. if we take a contemporary consumer OS, like Vista, Linux or Mac OSX, all of them have monolithic kernels, meaning that a simple bug in any of the kernel components (think: hundreds of 3rd party drivers running there), allows to bypass of the isolation mechanisms provided by the kernel to the rest of the system (process separation, ACLs, etc).

Obviously the problem is because the kernels are monolithic. Why not implement Security by Isolation on a kernel level then? Well, I would personally love that approach, but the industry simply took another course and decided that monolithic kernels are better than micro-kernels, because it's easier to write the code for them and (arguably) they offer better performance.

Many believe, including myself, that this landscape can be changed by the virtualization technology. Thin bare-metal hypervisor, like e.g. Xen, can act like a micro kernel and enforce isolation between other components in the system - e.g. we can move drivers into a separate

domain and isolate them from the rest of the system. But again there are challenges here on both the design- as well as the implementation-level. For example, we should not put all the drivers into the same domain, as this would provide little improvement in security. Also, how to make sure that the hypervisor itself is not buggy?

3] Security by Obscurity (or Security by Randomization)

Finally we have the Security by Obscurity approach that is based on the assumption that we cannot get rid of all the bugs (like in Security by Isolation approach), but at least we can make exploitation of those bugs very hard. So, it's all about making our system unfriendly to the attacker.

Examples of this approach include Address Space Layout Randomization (ASLR, present in all newer OSes, like Linux, Vista, OSX), StackGuard-like protections (again used by most contemporary OSes), pointer encryption (Windows and Linux) and probably some other mechanisms that I can't remember at the moment. Probably the most extreme example of Security by Obscurity would be to use a compiler that generates heavily obfuscated binaries from the source code and creates a unique (on a binary level) instances of the same system. Alex did his PhD on this topic and is an expert on compilers and obfuscators.

The obvious disadvantage of this approach is that it doesn't prevent the bugs from being exploited - it only make the meaningful exploitation very hard or even impossible. But if one is concerned also about e.g. DoS attacks, then Security by Obscurity will not prevent them in most cases. The other problem with obfuscating the code is the performance (compiler cannot optimize the code for speed) and maintenance (if we got a crash dump on an "obfuscated" Windows box, we couldn't count on help from the technical support). Finally there is a problem of proving that the whole scheme is correct and that our obfuscator (or e.g. ASLR engine) doesn't introduce bugs to the generated code and that we will not get random crashes later (that we would be most likely unable to debug, as the code will be obfuscated).

1.4] Principle Of Security :

1. Confidentiality

- The confidentiality principle of security states that only their intended sender and receiver should be able to access messages, if an unauthorized person gets access to this message then the confidentiality gets compromised.
- For example, suppose user X wants to send a message to user Y, and X does not want some else to get access to this message, or if it gets access, he/she does not come to

know about the details. But if user Z somehow gets access to this secret message, which is not desired, then the purpose of this confidentiality gets fail. This leads to the interception. i.e. if user Z access the secret message or email sent by user X to Y without permission of X and Y, then it is called an interception. Interception causes loss of message confidentiality.

2. Authentication

- The authentication principle of security establishes proof of identity, it ensures that the origin of a document or electronic message is correctly identified.
- For example suppose user Z sends a message to user Y, however, the trouble is that user Z posed as user X while sending a message to user Y. How would user Y know that message comes from Z, not X. This leads to the fabrication attack. For example
- The attacker can act as user X and sends fund transfer request(from X' account to attacker account) to a bank, and the bank will transfer the amount as requested from X's account to attacker, as banks think fund transfer request comes from user X. Fabrication is possible in absence of proper authentication mechanism.

3. Integrity

- The integrity principle of security states that the message should not be altered. In other words, we can say that, when the content of the message changes after the sender sends it, but before it reaches the intended receiver, we can say that integrity of the message is lost.
- For example, suppose user X sends a message to User Y, and attacker Z somehow gets access to this message during transmission and changes the content of the

message and then sends it to user Y. User Y and User X does not have any knowledge that the content of the message was changed after user X send it to Y. This leads to a modification. Modification causes loss of message integrity.

4. Non-repudiation

- Non-repudiation principle of security does not allow the sender of a message to refute the claim of not sending that message. There are some situations where the user sends a message and later on refuses that he/she had sent that message.
- For example, user X sends requests to the bank for fund transfer over the internet. After the bank performs fund transfer based on user X request, User X cannot claim that he/she never sent the fund transfer request to the bank. This principle of security defeats such possibilities of denying something after having done it.

5. Access control

- Access control principles of security determine who should be able to access what. i.e. we can specify that what users can access which functions,
- for example, we can specify that user X can view the database record but cannot update them, but user Y can access both, can view record, and can update them. This principle is broadly related to two areas – role management and rule management where role management concentrates on the user side. i.e. which user can do what and rule management concentrate on the resources side i.e. which resource is available. Based on this matrix is prepared, which lists the user against q list of items they can access. The access control list is a subset of the access control matrix.

6. Availability

- The availability principle of security states that resources should be available to the authorized person at all times.
- For example, because of the intentional action of another unauthorized user Z, an authorized user x may not be able to contact server Y, this leads to an interruption attack, interruption puts the availability of resources in danger. A real-life example of this could be, suppose attacker or unauthorized person Z tries to access the FB Account of user X, as User Z does not know the password of user X, he/she tries to log in to the X's account using a random password. after attempting maxim limit for the password, if it is not correct then X's account will be blocked, therefore because of unauthorized person Z, user X could not access his account.

7. Ethical and legal issues

Ethical issues in the security system are classified into the following categories

- **Privacy:** It deals with the individual's right to access the personal information
- **Accuracy:** It deals with the responsibility of authentication, fidelity, and accuracy of information
- **Property:** It deals with the owner of the information
- **Accessibility:** It deals with what information does an organization has the right to collect.

while dealing with legal issues, we must remember that there is a hierarchy of regulatory bodies that govern the legality of information security, it can be classified into the following categories

- International
- Federal

- State
- Organization

1.5] Type Of Attacks

Attacks are typically categorized based on the action performed by the attacker. An attack, thus, can be **passive** or **active**.

A] Passive Attacks

The main goal of a passive attack is to obtain **unauthorized access to the information**. For example, actions such as intercepting and eavesdropping on the communication channel can be regarded as passive attack.

These actions are passive in nature, as they neither affect information nor disrupt the communication channel. A passive attack is often seen as *stealing* information. The only difference in stealing physical goods and stealing information is that theft of data still leaves the owner in possession of that data. Passive information attack is thus more dangerous than stealing of goods, as information theft may go unnoticed by the owner.

B] Active Attacks

An active attack involves changing the information in some way by conducting some process on the information. For example,

- Modifying the information in an unauthorized manner.
 - Initiating unintended or unauthorized transmission of information.
 - Alteration of authentication data such as originator name or timestamp associated with information
 - Unauthorized deletion of data.
 - Denial of access to information for legitimate users (denial of service).
-
- Cryptography provides many tools and techniques for implementing cryptosystems capable of preventing most of the attacks described above.

1. A] Cryptography

A.1] Concept And Techniques Introduction :

- *The art and science of concealing the messages to introduce secrecy in information security is recognized as cryptography.*
 - The word 'cryptography' was coined by combining two Greek words, 'Krypto' meaning hidden and 'graphene' meaning writing.
 - **Cryptography** is the study and practice of techniques for secure communication in the presence of third parties called adversaries. It deals with developing and analyzing protocols that prevents malicious third parties from retrieving information being shared between two entities thereby following the various aspects of information security.
 - Secure Communication refers to the scenario where the message or data shared between two parties can't be accessed by an adversary. In Cryptography, an Adversary is a malicious entity, which aims to retrieve precious information or data thereby undermining the principles of information security.
 - Data Confidentiality, Data Integrity, Authentication and Non-repudiation are core principles of modern-day cryptography.
1. **Confidentiality** refers to certain rules and guidelines usually executed under confidentiality agreements which ensure that the information is restricted to certain people or places.
 2. **Data integrity** refers to maintaining and making sure that the data stays accurate and consistent over its entire life cycle.
 3. **Authentication** is the process of making sure that the piece of data being claimed by the user belongs to it.
 4. **Non-repudiation** refers to the ability to make sure that a person or a party associated with a contract or a communication cannot deny the authenticity of their signature over their document or the sending of a message.

1.A.2] Plain Text And Cypher Text

❖ What is Plaintext?

In cryptography, Plaintext is usually plain readable text before it is encrypted into ciphertext or readable text after it is decrypted. Any message, document, file, or the like that is not meant to be encrypted is referred to as plaintext.

- The input to a cryptosystem is plaintext, and the output is ciphertext. Algorithms in cryptography convert plaintext to ciphertext and ciphertext to plaintext.

- Plaintext saved in a computer file must be protected since its contents are completely disclosed and hence potentially actionable if it is stolen, disclosed, or sent without permission. If data is to be saved, the storage medium, device, components, and backups must all be protected.

Examples of Plaintext

Plaintext is recommended in most applications. Plaintext should be shown in a browser, word processor, or email client, for example. However, early Internet network protocols occasionally transmitted user ID and password combinations in plaintext. This is a bad security practice since it exposes user credentials for systems that are accessible remotely over a public network such as the Internet.

- Although the Password Authentication System provided a two-way handshake authentication exchange protocol, it did not include a mechanism for encrypting credentials.
- Those credentials are exposed if plaintext passwords are written in application configuration files. Developers that keep plaintext passwords in their source code are less likely to expose their credentials.
- Password protection in PowerShell scripts to avoid revealing those passwords in their scripts, developers must take care.

Plaintext Attack :

- The **known-plaintext attack (KPA)** is a cryptanalysis attack paradigm in which the attacker has both the plaintext (also known as a crib) and its encrypted form (ciphertext). These can be used to uncover more secrets, such as secret keys and codebooks.
- Known-plaintext attacks are common against traditional ciphers. A Caesar cipher, for example, can be decrypted entirely using a single letter of corresponding plaintext and ciphertext. If there are less than 26 unique pairs, a universal monoalphabetic substitution cipher requires multiple character pairs and some guessing.

❖ Defining Ciphertext

What is ciphertext in cryptography?

Ciphertext is encrypted text transformed from plaintext using an encryption algorithm. Ciphertext can't be read until it has been converted into plaintext (decrypted) with a key. The decryption cipher is an algorithm that transforms the ciphertext back into plaintext.

- The result of employing encryption methods, often referred to as cyphers, is called ciphertext. When data cannot be understood by individuals or devices lacking the appropriate cypher, it is considered encrypted. To interpret the data, the cypher is necessary. Algorithms transform plaintext into ciphertext, and vice versa, to convert ciphertext back into plaintext. These processes are known as encryption and decryption.
- Ciphertext, represents a cryptographic approach in which an algorithm utilizes substitutions instead of original plaintext elements. Substitution ciphers replace individual letters, letter pairs, letter triplets, or various combinations of these while preserving the initial sequence. Single-letter substitutions are utilized in simple substitution cyphers, while polygraphed cyphers involve larger letter groupings.
- In simpler terms, letters are substituted for other letters. In the past, recording corresponding characters to decipher a message was feasible.

For example, "GOOD DOG" can be encrypted as "PLLX XLP" where "L" substitutes for "O", "P" for "G", and "X" for "D" in the message. Transposition of the letters "GOOD DOG" can result in "DGOOGDOO". These simple ciphers and examples are easy to crack, even without plaintext-ciphertext pairs.

What is the formula for ciphertext?

The conversion formula is of the form $c \equiv p + a \pmod{26}$. We know that when $p = 5$ (plaintext E), we have $c = 10$ (ciphertext J). Thus, $10 \equiv 5 + a \pmod{26}$. So $a \equiv 5 \pmod{26}$, and the encryption formula is $c \equiv p + 5 \pmod{26}$.

1.A.3] Substitution And Transposition Technique

What is Substitution Cipher Technique?

In the Substitution Cipher technique, each character is substituted with other character/number or other symbol. This technique changes the identity of a character but not its position in the string.

A substitution cipher encrypts a text sequence by replacing letters or units of text with other letters or units of text. Substitution ciphers are an early form of cryptography that predates the development of computers and are now largely out-dated.

Like B or C, a letter is converted into another letter in a substitution cipher, encrypting the sequence for a human reader. The problem is that basic substitution ciphers do not encrypt adequately in terms of computer assessment.

Substitution ciphers became relatively easy for computers to crack with the emergence of the personal computer. However, some of the substitution cipher's concepts have persisted, for

example, some versions of current encryption may encrypt data using an extensive text set and a very clever substitution.

What is Transposition Cipher Technique?

In the Transposition Cipher Technique, each character's position is shifted to a different position. A transposition cipher is an encryption method in which plaintext units (usually letters or groups of characters) are shifted in a predictable way resulting in the ciphertext being a permutation of the plaintext. That is, the units' order is altered (the plaintext is reordered).

To encrypt, a objective function is applied to the locations of the characters, and to decrypt, an inverse function is used.

The Rail Fence encryption is a type of transposition cipher named after how it is encoded. The plaintext is written on successive "rails" of an imaginary fence in the rail fence, downwards and diagonally, then moved up when we reach the bottom. The message is then read aloud in a series of rows.

The Rail Fence Cipher follows a scytale-like pattern, an old Greek mechanical device for constructing a transposition cipher. A cylinder and a ribbon wrapped around the cylinder comprised the system. The coiled ribbon was used to write the encrypted message. When the ribbon was uncoiled from the cylinder, the letters of the original message were rearranged. The message was decoded when the ribbon recoiled on a cylinder with the same diameter as the encrypting cylinder.

Difference between Substitution Cipher and Transposition Cipher

The following table highlights the important differences between Substitution Cipher and Transposition Cipher.

Key	Substitution Cipher Technique	Transposition Cipher Technique
Algorithm	Each character is replaced with a different character, integer, or symbol.	Each character has been repositioned from its original place.

Key	Substitution Cipher Technique	Transposition Cipher Technique
Forms	It comes in two forms: Mono-Alphabetic Substitution Cipher and Poly-Alphabetic Substitution Cipher.	It has two forms: Key-less transposition cipher and keyed transposition cipher.
Change	Character identity is changed but position remains same.	Character position is changed but identity remains same.
Detection	A letter less frequently used can be easily traced.	A letter near to original position can be easily traced.
Example	Caesar Cipher is an example of Substitution Cipher.	Reil Fence Cipher is an example

1.A.4] Encryption And Decryption

- The process of turning plain text into an encrypted form (cipher text) is known as encryption. Sensitive data is transmitted in an encrypted form so that it can be protected and a strong encryption mechanism ensures that the data is not misused even if a hacker gets hold of it. Decryption is the reverse mechanism where the encrypted cipher text is converted back into its original form.

What is Encryption?

Encryption is the procedure of encoding data in cryptography. This procedure turns plaintext, or the original representation of the data, into cipher text or an alternate representation of the data. Only authorized parties should be able to decrypt a cipher text back to plaintext and access the original data. Encryption does not prevent interference in and of itself, but it does hinder a would-be interceptor from understanding the information.

An encryption system generally employs a pseudo-random encryption key produced by an algorithm for technical reasons. Although it is possible to decode a message without knowing the

key, a well-designed encryption system requires significant computational resources and skills are required. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients but not to unauthorized users.

- Encrypted data typically appears as a long string of random letters and numbers.
- Once the information has been encrypted, the only way to decrypt it and make it readable again is to use the correct encryption key. Encryption is necessary for the secure transmission and storage of sensitive data.
- Stream ciphers encrypt data one bit or byte at a time, making them ideal for real-time communications. Before encrypting data, block ciphers divide it into larger chunks, usually 64 bits.

What is Decryption?

Decryption is the procedure of restoring encrypted data to its original state. In most cases, it's the reverse mechanism of the encryption process. Because decryption needs a secret key or password, it decodes the encrypted information so that only an authorized user may decrypt the data.

As information flows across the Internet, it becomes vulnerable to unauthorized persons or groups scrutinizing and accessing it. As a consequence, data is encrypted in order to prevent data theft and loss. Email communications, text files, pictures, user data, and directories are just a few of the objects that are commonly encrypted. The person in charge of decryption receives a popup or window asking for a password to access the encrypted data.

Difference between Encryption and Decryption

The following table highlights the major differences between encryption and decryption –

Key	Encryption	Decryption
Definition	Encryption is a process of converting a plain text into an encrypted or cipher text.	Decryption is a process of converting the encrypted or cipher text into plain text.
Place of Occurrence	Encryption takes place at the sender's end.	Decryption is done at the receiver's end.
Process	A secret key or a public key must be used to encrypt	Any communication that requires the use of a secret

Key	Encryption	Decryption
	every message.	key or private key to decode.
Actor	After encrypting the data with a secret key or a public key, the sender delivers it to the recipient.	The receiver gets the encrypted data and uses the secret key or private key to decode it.

1.A.5] Symmetric Key Cryptography

In symmetric key cryptography, an individual key is used for both encryption and decryption. The sender needs the key to encrypt the plaintext and sends the cipher document to the receiver. The receiver used the similar key (or rule set) to decrypt the message and recover the plaintext. Because an individual key is used for both functions, symmetric key cryptography is also known as symmetric encryption.

Symmetric key cryptography schemes are usually categorized such as stream ciphers or block ciphers. Stream ciphers works on a single bit (byte or computer word) at a time and execute some form of feedback structure so that the key is constantly changing.

❖ Asymmetric cryptography

Asymmetric cryptography uses two keys for encryption and decryption. It depends on the technique of public and private keys. A public key, which is interchanged between higher than one user. Data is decrypted by a private key, which is not transformed. It is slower but more secure. The public key used in this encryption technique is applicable to everyone, but the private key used in it is not revealed.

In asymmetric encryption, a message that is encrypted utilizing a public key can be decrypted by a private key, while if the message is encrypted by a private key can be decrypted by utilizing the public key. Asymmetric encryption is broadly used in dayto- day communication channels, particularly on the internet.

Symmetric Key Cryptography

There is only one key (symmetric key) is used, and the similar key can be used to encrypt and decrypt the message.

It is effective as this technique is recommended for high amounts of text.

Symmetric encryption is generally used to transmit bulk information.

Symmetric key cryptography is also known as secret-key cryptography or private key cryptography.

Symmetric key cryptography uses fewer resources as compared to asymmetric key cryptography.

The length of the keys used is frequently 128 or 256 bits, based on the security need.

Asymmetric Key Cryptography

There are two different cryptographic keys (asymmetric keys), known as the public and the private keys, are used for encryption and decryption.

It is inefficient as this approach is used only for short messages.

It is generally used in smaller transactions. It is used for making a secure connection channel before transferring the actual information.

Asymmetric key cryptography is also known as public-key cryptography or a conventional cryptographic system.

Asymmetric key cryptography uses more resources as compared to symmetric key cryptography.

The length of the keys is much higher, such as the recommended RSA key size is 2048 bits or higher.

1.A.6] Steganography

Steganography is the art and science of embedding secret messages in a cover message in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message

The diagram below depicts a basic steganography model.

As the image depicts, both cover file(X) and secret message(M) are fed into steganographic encoder as input. Steganographic Encoder function, $f(X,M,K)$ embeds the secret message into a cover file. Resulting Stego Object looks very similar to your cover file, with no visible changes. This completes encoding. To retrieve the secret message, Stego Object is fed into Steganographic Decoder.

Steganography: Historical Background

Steganography is the practice of concealing a secret message behind a normal message. It stems from two Greek words, which are *steganos*, means covered and *graphia*, means writing. Steganography is an ancient practice, being practiced in various forms for thousands of years to keep communications private. For Example:

- The first use of steganography can be traced back to 440 BC when ancient Greece, people wrote messages on wood and covered it with wax, that acted as a covering medium
- Romans used various forms of Invisible Inks, to decipher those hidden messages light or heat were used
- During World War II the Germans introduced microdots, which were complete documents, pictures, and plans reduced in size to the size of a dot and were attached to normal paperwork
- Null Ciphers were also used to hide unencrypted secret messages in an innocent looking normal message

How is Steganography different from Cryptography?

[Cryptography](#) changes the information to ciphertext which cannot be understood without a decryption key. So, if someone were to intercept this encrypted message, they could easily see that some form of [encryption](#) had been applied. On the other hand, steganography does not change the format of the information but it conceals the existence of the message.

	STEGANOGRAPHY	CRYPTOGRAPHY
Definition	It is a technique to hide the existence of communication	It's a technique to convert data into an incomprehensible form
Purpose	Keep communication secure	Provide data protection
Data Visibility	Never	Always
Data Structure	Doesn't alter the overall structure of data	Alters the overall structure of data
Key	Optional, but offers more security if used	Necessary requirement
Failure	Once the presence of a secret message is discovered, anyone can use the secret data	If you possess the decryption key, then you can figure out original message the ciphertext

1.A.7] Key Range And Key Size

Key Size :

- In cryptography, key size or key length is the size (measured in bits or bytes) of the key used in a cryptographic algorithm (such as a cipher). Typical key sizes in modern symmetric ciphers are 128, 192, and 256 bits. Older symmetric ciphers used only 40, 56, or 64 bits, which can be broken by brute force.
- The longer the key, the more secure data can be viewed with it. The key length is an encryption security standard and varies by the protocol used and the algorithm.

Key Range :

Key range" refers to the set of all possible keys that can be used in cryptography. The range is determined by the key size and the underlying algorithm, and it affects the security of the encryption.

KEY RANGE AND KEY SIZE

- The encryption/decryption algorithm
- The encrypted message
- Knowledge about the key size (e.g. the value of the key is a number between 0 and 100 billion).

For example consider the brute force attack here, which works on the principle of trying every possible key in the key range, until you get the right key. Brute force attack

Understanding key range With every incremental bit, the attacker has to perform double the

number of operations as compared to the previous key size. It is found that for a 56-bit key, 41 it takes 1 second to search 1 percent of the key range. Taking this argument further, it takes about 1 minute to search about half of the key range (which is what is required, on an average, to crack a key). Using this as the basis, let us have a look at the similar values (time required for a search of 1 percent and 50 percent of the key space) for various key sizes. This is shown in Table We can represent the possible values in the key range using hexadecimal notation and see visually how an increase in the key size increases the key range and therefore, the complexity for an attacker. Key sizes and ranges

1.A.8] POSSIBLE TYPES OF ATTACKS

When the sender of a message encrypts a plain text message into its corresponding cipher text, there are five possibilities for an attack on this message.

- **Cipher text only attack:** In this type of attack, the attacker does not have any clue about the plaintext and has some or all of the cipher text. The attacker analyzes the cipher text at leisure to try and figure out the original plain text.
- **Known plain text attack:** In this case, the attacker knows about some pairs of plain text and corresponding cipher text for those pairs. Using this information, the attacker tries to find other pairs and therefore, know more and more of the plain text. Examples of such known plain texts are company banners, file headers, etc. which are found commonly in all the documents of a particular company.
- **Chosen plain text attack:** Here, the attacker selects a plain text block and tries to look for the encryption of the same in the cipher text. Here, the attacker is able to choose the messages to encrypt. Based on this, the attacker intentionally Security, picks patterns of cipher text that result in obtaining more information about the key.
- **Chosen cipher text attack:** In the chosen cipher text attack, the attacker knows the cipher text to be decrypted, the encryption algorithm that was used to produce this cipher text and the

corresponding plain text block. The attacker's job is to discover the key used for encryption.

Chosen text attack: The chosen text attack is essentially a combination of chosen plain text attack and chosen cipher text attack

Cryptography

Unit 2 : Symmetric Key Algorithms And AES

2.1] Introduction

What is Symmetric Key Algorithms?

Symmetric key algorithms are a type of cryptographic technique that uses a shared secret key for both encryption and decryption. This means that the same key is used to encode and decode the message. Symmetric key algorithms are generally faster and more efficient than asymmetric key algorithms, but they require that the sender and receiver of a message share a secret key.

Here are some of the basic principles of symmetric key algorithms –

- **The same key is used for both encryption and decryption** – In symmetric key algorithms, the same key is used to both encrypt and decrypt the message. This means that the sender and receiver of a message must share the same secret key in order to communicate securely.
- **Symmetric key algorithms are faster and more efficient than asymmetric key algorithms** – Symmetric key algorithms are generally faster and more efficient than asymmetric key algorithms, as they do not require the use of complex mathematical operations such as exponentiation. This makes them well-suited for applications that require fast encryption and decryption, such as securing communication over the internet.
- **Symmetric key algorithms are less secure than asymmetric key algorithms** – While symmetric key algorithms are generally faster and more efficient than asymmetric key algorithms, they are also less secure. This is because the same key is used for both encryption and decryption, which means that if the key is compromised, the security of the entire system is compromised.

Overall, symmetric key algorithms are an important type of cryptographic technique that are used to secure communication and protect data. While they are generally faster and more efficient than asymmetric key algorithms, they are also less secure and require that the sender and receiver of a message share a secret key.

Cryptographic Strength of Symmetric Algorithms

The cryptographic strength of a symmetric key algorithm refers to its ability to resist attacks and protect the confidentiality of the information it is used to encrypt. The cryptographic strength of a symmetric key algorithm is determined by a variety of factors, including –

- **Key size** – The size of the key used in a symmetric key algorithm is a major determinant of its cryptographic strength. In general, the larger the key size, the stronger the algorithm.
- **Block size** – The block size of a symmetric key algorithm refers to the size of the blocks of data that are encrypted and decrypted using the algorithm. A larger block size can increase the cryptographic strength of the algorithm.
- **Number of rounds** – The number of rounds in a symmetric key algorithm refers to the number of times that the encryption and decryption process is repeated. A larger number of rounds can increase the cryptographic strength of the algorithm.
- **Resistance to attacks** – The resistance of a symmetric key algorithm to attacks, such as brute-force attacks or differential cryptanalysis, is another factor that determines its cryptographic strength. Algorithms that are resistant to these types of attacks are generally considered to be stronger.

Overall, the cryptographic strength of a symmetric key algorithm is determined by a combination of these and other factors. Stronger algorithms are generally more resistant to attacks and more effective at protecting the confidentiality of the information they are used to encrypt.

2.2] Types of Symmetric Key Algorithms

There are several different types of symmetric key algorithms, including –

- **Block ciphers** – Block ciphers are symmetric key algorithms that operate on fixed-size blocks of data and use a secret key to encrypt and decrypt the data. Examples of block ciphers include the Advanced Encryption Standard (AES) and Blowfish.
- **Stream ciphers** – Stream ciphers are symmetric key algorithms that operate on a stream of data and use a secret key to encrypt and decrypt the data. Stream ciphers are generally faster and more efficient than block ciphers, but they are also generally considered to be less secure.
- **Feistel ciphers** – Feistel ciphers are a type of block cipher that are based on a structure known as a Feistel network. They are widely used in symmetric key algorithms and are known for their efficiency and ease of implementation.
- **Substitution-permutation ciphers** – Substitution-permutation ciphers are a type of block cipher that use both substitution and permutation operations to encrypt and decrypt data. They are known for their strong cryptographic properties and are used in many modern symmetric key algorithms.

Overall, there are many different types of symmetric key algorithms, each with its own unique characteristics and strengths. Symmetric key algorithms are an important type of cryptographic technique that are used to secure communication and protect data.

2.3] An Overview Of Symmetric Key Cryptography :

Symmetric key cryptography is a type of encryption scheme in which the similar key is used both to encrypt and decrypt messages. Such an approach of encoding data has been largely used in the previous decades to facilitate secret communication between governments and militaries.

Symmetric-key cryptography is called a shared-key, secret-key, single-key, one-key and eventually private-key cryptography. With this form of cryptography, it is clear that the key should be known to both the sender and the receiver that the shared. The complexity with this approach is the distribution of the key.

Symmetric key cryptography schemes are usually categorized such as stream ciphers or block ciphers. Stream ciphers work on a single bit (byte or computer word) at a time and execute some form of feedback structure so that the key is repeatedly changing.

A block cipher is so-called because the scheme encrypts one block of information at a time utilizing the same key on each block. In general, the same plaintext block will continually encrypt to the same ciphertext when using the similar key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher.

Block ciphers can operate in one of several modes which are as follows –

- Electronic Codebook (ECB) mode is the simplest application and the shared key can be used to encrypt the plaintext block to form a ciphertext block. There are two identical plaintext blocks will always create the same ciphertext block. Although this is the most common mode of block ciphers, it is affected to multiple brute-force attacks.
- Cipher Block Chaining (CBC) mode insert a feedback structure to the encryption scheme. In CBC, the plaintext is exclusively-ORed (XORed) with the prior ciphertext block prior to encryption. In this mode, there are two identical blocks of plaintext not encrypt to the similar cipher text.
- Cipher Feedback (CFB) mode is a block cipher implementation as a self synchronizing stream cipher. CFB mode enable data to be encrypted in units lower than the block size, which can be beneficial in some applications including encrypting interactive terminal input. If it is using 1-byte CFB mode.

Each incoming character is located into a shift register the similar size as the block, encrypted, and the block transmitted. At the receiving side, the ciphertext is decrypted and the more bits in the block are discarded.

- Output Feedback (OFB) mode is a block cipher implementation conceptually same to a synchronous stream cipher. OFB avoids the similar plaintext block from making the same cipher text block by using an internal feedback structure that is independent of both the plaintext and cipher text bit streams.

2.4] DES (Data Encryption Standard)

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration –

Since DES is based on the Feistel Cipher, all that is required to specify DES is –

- Round function
- Key schedule
- Any additional processing – Initial and final permutation

Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows –

Round Function

The heart of this cipher is the DES function, f . The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

- **Expansion Permutation Box** – Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration –

- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown –

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

- **XOR (Whitener).** – After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** – The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration –
- The S-box rule is illustrated below –
- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.
- **Straight Permutation** – The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration –

The logic for Parity drop, shifting, and Compression P-box is given in the DES description.

DES Analysis

The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

- **Avalanche effect** – A small change in plaintext results in the very great change in the cipher text.
- **Completeness** – Each bit of cipher text depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.

DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

2.5] IDEA (International Data Encryption Algorithm)

IDEA stands for International Data Encryption Algorithm. IDEA is a block cipher invented by James Massey and Xuejia Lai and was first defined in 1991. It uses 128 bit key length which works on 64 bit blocks.

It includes a series of eight identical transformations depend upon bitwise exclusive or, addition and multiplication modules. It is based upon symmetric cipher and has very weak key design approach therefore security level of the algorithm is very underprivileged as compared to the DES. IDEA not becomes so much famous because of its complex structure.

It is an IDEA, unlike the other block cipher algorithms is considered by the Swiss firm of As com. However, be unique in allow with permission with free non-commercial use of their algorithm which the result that IDEA is popular as the block cipher algorithm used within the famous encryption for the message character.

There are eight rounds in IDEA. Each round includes a sequence of operations on the four data blocks using six keys. For the first round, it can have key K_1 to K_6 , for second round it can have keys K_7 to K_{12} and finally the last round. The final step includes an output transformation, which needs four sub keys (K_{49} to K_{52}).

The final output is the output developed by the output transformation step. The blocks C_1 to C_4 are linked to form the final output. Each round includes 14 steps are as follows –

Details of one round in IDEA

Step1 : Multiply * P_1 and K_1

Step2 : Add * P_2 and K_2

Step3 : Add * P_3 and K_3

Step4 : Multiply * P_4 and K_4

Step5 : XOR the results of step1 and step3

Step6 : XOR the results of step2 and step4

Step7 : Multiply * the results of step5 with K_5

Step8 : Add * the results of step6 and step7

Step9 : Multiply * the results of step8 with K_6

Step10 : Add * the results of step7 and step9

Step11 : XOR the results of step1 and step9

Step12 : XOR the results of step3 and step9

Step13 : XOR the results of step2 and step10

Step14 : XOR the results of step4 and step10

The Add * and Multiply * in the following step of each round are not elementary addition and multiplication but they are addition module 2^{16} i.e., 65536 and Multiplication Module $2^{16} + 1$ i. e. , 65537.

Output Transformation – It is a one-time operation. It takes place endmost of the 8th round. Therefore 64-bit value is divided into four sub-blocks (say R_1 to R_4) and four sub keys are used here.

Decryption – The Decryption implements like encryption, but the order of the round keys is produced, and the sub keys for the odd rounds are inversed into this. Thus, the values of sub keys K_1 - K_4 are restored by the inverse of K_{49} - K_{52} for the respective set operation in the cryptography, K_5 , and K_6 of each group should be restored by K_{47} and K_{48} for decryption in this operation in the cryptography.

How does IDEA Encryption work?

IDEA is a block cipher and it operates on 64bit plaintext and 128 bit key. IDEA is reversible like DES that is, the equivalent algorithm can be used for encryption and decryption. IDEA needs both diffusion and confusion for encryption.

The 64-bit plaintext is divided into four portions of 16bit plaintext (P_1 to P_4). These are input to the first round. There are eight such rounds. The key includes 128 bits. In each round, six sub keys are produced from the original key, each of these sub-key includes 16 bits.

For the first round it can have key K_1 to K_6 , for second round it can have keys K_7 to K_{12} and finally the last round. The final step includes an output transformation, which needs four sub keys (K_{49} to K_{52}).

The final output is the output created by the output transformation step. The blocks C_1 to C_4 are linked to form the final output.

Rounds – There are eight round in IDEA. Each round contains a sequence of operations on the four data blocks, utilizing six keys. The Add * and Multiply * in the following step of each round are not easy addition and multiplication but they are addition module 2^{16} i.e., 65536 and Multiplication Module $2^{16} + 1$ i. e. , 65537.

Sub-key generation for a round –

- In the first round bit position 1-96 of the key are utilized. Bits 97-128 remain unutilized. They are given to round 2.
- In round second bits 97-128 are first used, thus circular left shift of 25 bit appears and new bits from 26-89 are used. Bits position 90-128 and 1-25 remains unused.
- In round third unused bit 90-128 and 1-25 are first used again, circular left shift of 25 bit appears and bit position 51-82 are used. Bit position 83-128 and 1-50 remains unused.
- In the fourth round bit 83-128 and 1-50 are used.
- In the fifth round circular left-shift of 25 bit appears and bit position 76-128 and 1-43 are used. Bit position 44-75 remains unused.
- In the sixth round unused bit of fifth round is used and left circular shift of 25 bit occur thus making bit position 37-100 unused.
- In the seventh round unused bit of sixth round i.e., 37-100 are first used and circular left shift of 25 bit appear bit position 126-128 and 1-29 are used. Bit 30-125 remain unused.
- Unused bit position 30-125 from round seven is used and key is disabled.

Output Transformation – It is a one-time operation. It takes place at the end of the 8th round. Therefore 64-bit value is divided into four sub-blocks (say R_1 to R_4) and four sub keys are used here

IDEA Decryption – The decryption procedure is the equivalent as the encryption process. There are some alteration in the generation and design of sub keys. The decryption subkeys are opposite of the encryption sub keys.

2.6] RC4

What is RC4?

RC4 stands for Rivest Cipher 4. Ron Rivest invented RC4 in 1987, and it is a stream cipher. Because RC4 is a stream cipher, it encrypts data bytes by bits. Because of its speed and simplicity, RC4 is the most extensively used stream cipher of all the stream ciphers.

While RC4 is known for its ease of use and speed in software, it has been found to have several weaknesses, making it insecure. When the beginning of the output key stream isn't destroyed, or when non-random or linked keys are utilized, it's highly vulnerable. The usage of RC4, in particular, has resulted in relatively insecure protocols such as WEP.

As of 2015, several state cryptologic agencies were suspected of being able to break RC4 when it was employed in the TLS protocol. RFC 7465, published by the Internet Engineering Task Force, prohibits the use of RC4 in TLS, and Mozilla and Microsoft have issued similar recommendations.

Working of RC4

RC4 creates a pseudo-random bit stream (a key stream). These, like any other stream cipher, can be used for encryption by utilizing bit-wise exclusive or to combine it with the plaintext. The same procedure is used for decryption (since exclusive-OR is a symmetric operation).

The cipher uses a secret internal state that is divided into two sections to generate the key stream –

- Each of the 256 available bytes is permuted.
- Two index pointers (8 bits each).

The key-scheduling algorithm is known to initialize the permutation using a variable-length key, typically between 40 and 256 bits (KSA). A pseudo-random generating technique then generates the stream of bits.

For encryption –

- The user enters the Plaintext and a secret key.

- For the secret key entered, the encryption engine creates the key stream using the KSA and PRGA algorithms.
- Plaintext is XORed with the generated key stream. Because RC4 is a stream cipher, byte-by-byte XORing is used to generate the encrypted text.
- This encrypted text is now sent in encrypted form to the intended recipient.

For Decryption –

- The same byte-wise X-OR technique is used on the cipher text to decrypt it.

Usage of RC4

Over the years, RC4 has grown in popularity and has become a standard in commercial applications. It has a reputation for being a simple, quick, and inexpensive encryption technology.

The key benefits of RC4 are its ease of implementation and use, as well as its speed of operation and deployment. It enables efficient and quick processing of large data streams. In terms of memory usage, RC4 stream ciphers are also efficient.

However, due to proof of flaws and cyber attacks in recent years, there have been calls to stop using RC4 encryption algorithms. Other drawbacks were identified, such as the inability to operate with small data streams and the need for additional investigation prior to implementing new systems.

The Internet Engineering Task Force (IETF) banned the usage of RC4 in TLS protocols in 2015. Because of threat vulnerabilities, Microsoft and Mozilla have also issued recommendations to limit the use of RC4. There are many RC4 based ecosystems such as WEP, WPA, Bit Torrent protocol encryption, Microsoft Point-to-Point Encryption, etc.

RC4A is a more powerful variation of RC4. **RC4A+** is a modified version of RC4 with a more complex 3-phase key schedule that is 1.7 times longer than the basic RC4.

Advantages and Disadvantages of Using RC4 Encryption

Following are the advantages of using RC4 Encryption –

- It is easy to use RC4 stream ciphers.
- In comparison to other ciphers, RC4 has a quick operation speed.
- RC4 stream ciphers have a high coding strength and are simple to construct.
- RC4 stream ciphers do not require additional memory.

Following are the disadvantages of using RC4 Encryption –

- Encryption is vulnerable to a bit-flipping attack if RC4 is not used with a robust MAC.
- Authentication is not possible using RC4 stream ciphers.
- Before adding new systems to the RC4 algorithm, more research is needed.
- RC4 stream ciphers can't be used with tiny data streams.

❖ RC4 Algorithm :

•

RC4 is a stream cipher and variable-length key algorithm. This algorithm encrypts one byte at a time (or larger units at a time). A key input is a pseudorandom bit generator that produces a stream 8-bit number that is unpredictable without knowledge of input key, The output of the generator is called key-stream, is combined one byte at a time with the plaintext stream cipher using X-OR operation.

Example:

RC4 Encryption

10011000 ? 01010000 = 11001000

RC4 Decryption

11001000 ? 01010000 = 10011000

Key-Generation Algorithm – A variable-length key from 1 to 256 bytes is used to initialize a 256-byte state vector S, with elements S[0] to S[255]. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion, then the entries in S are permuted again.

Key-Scheduling Algorithm: Initialization: The entries of S are set equal to the values from 0 to 255 in ascending order, a temporary vector T, is created. If the length of the key k is 256 bytes, then k is assigned to T. Otherwise, for a key with length(k-len) bytes, the first k-len elements of T as copied from K, and then K is repeated as many times as necessary to fill T. The idea is illustrated as follow:

- Java

for

i = 0 to 255 do S[i] = i;

T[i] = K[i mod k - len];

we use T to produce the initial permutation of S. Starting with S[0] to S[255], and for each S[i] algorithm swap it with another byte in S according to a scheme dictated by T[i], but S will still contain values from 0 to 255 :

- C++
- Java
- Python3
- C#
- Javascript

```
int j = 0;

for (int i = 0; i <= 255; i++) {

    j = (j + S[i] + T[i]) % 256;

    swap(S[i], S[j]); // Swap S[i] and S[j]

}
```

Pseudo random generation algorithm (Stream Generation): Once the vector S is initialized, the input key will not be used. In this step, for each S[i] algorithm swap it with another byte in S according to a scheme dictated by the current configuration of S. After reaching S[255] the process continues, starting from S[0] again

- Java

```
i, j = 0;

while (true)

    i = (i + 1) mod 256;

    j = (j + S[i]) mod 256;

    Swap(S[i], S[j]);

    t = (S[i] + S[j]) mod 256;

    k = S[t];
```

Encrypt using X-Or():

News: In September 2015, Microsoft announced the end of using RC4 in Microsoft edge and internet explorer 11. This video gives a clear example of RC4 algorithm.
<https://www.youtube.com/watch?v=NKnZARFAhMk>

Features of the RC4 encryption algorithm:

1. **Symmetric key algorithm:** RC4 is a symmetric key encryption algorithm, which means that the same key is used for encryption and decryption.
2. **Stream cipher algorithm:** RC4 is a stream cipher algorithm, which means that it encrypts and decrypts data one byte at a time. It generates a key stream of pseudorandom bits that are XORed with the plaintext to produce the ciphertext.
3. **Variable key size:** RC4 supports variable key sizes, from 40 bits to 2048 bits, making it flexible for different security requirements.
4. **Fast and efficient:** RC4 is a fast and efficient encryption algorithm that is suitable for low-power devices and applications that require high-speed data transmission.
5. **Widely used:** RC4 has been widely used in various applications, including wireless networks, secure sockets layer (SSL), virtual private networks (VPN), and file encryption.
6. **Vulnerabilities:** RC4 has several vulnerabilities, including a bias in the first few bytes of the key stream, which can be exploited to recover the key. As a result, RC4 is no longer recommended for use in new applications.

2.7] RC5

RC5 is a symmetric key block encryption algorithm designed by Ron Rivest in 1994. It is notable for being simple, fast (on account of using only primitive computer operations like XOR, shift, etc.) and consumes less memory.

Example:

Key : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Plain Text : 00000000 00000000

Cipher Text : EEDBA521 6D8F4B15

RC5 is a block cipher and addresses two word blocks at a time. Depending on input plain text block size, number of rounds and key size, various instances of RC5 can be defined and each instance is denoted as RC5-w/r/b where w=word size in bits, r=number of rounds and b=key size in bytes. Allowed values are:

Parameter	Possible Value
block/word size (bits)	16, 32, 64
Number of Rounds	0 – 255
Key Size (bytes)	0 – 255

Note – Since at a time, RC5 uses 2 word blocks, the plain text block size can be 32, 64 or 128 bits. Notation used in the algorithm:

Symbol	Operation
$x \lll y$	Cyclic left shift of x by y bits
+	Two's complement addition of words where addition is modulo
$\wedge$	Bit wise Exclusive-OR

Step-1: Initialization of constants P and Q. RC5 makes use of 2 magic constants P and Q whose value is defined by the word size w.

Word Size (bits)	P (Hexadecimal)	Q (Hexadecimal)
16	b7e1	9e37
32	b7e15163	9e3779b9
64	b7e151628aed2a6b	9e3779b97f4a7c15

For any other word size, P and Q can be determined as:

$$P = \text{Odd}((e-2)^{-1}) \quad Q = \text{Odd}((\phi^{-2})^{-1})$$

Here, $\text{Odd}(x)$ is the odd integer nearest to x , e is the base of natural logarithms and ϕ is the golden ratio. **Step-2:** Converting secret key K from bytes to words. Secret key K of size b bytes is used to initialize array L consisting of c words where $c = b/u$, $u = w/8$ and $w = \text{word size}$ used for that particular instance of RC5. For example, if we choose $w=32$ bits and Key k is of size 96 bytes then, $u=32/8=4$, $c=b/u=96/4=24$. L is pre initialized to 0 value before adding secret key K to it. for $i=b-1$ to 0

$$L[i/u] = (L[i/u] \lll 8) + K[i]$$

Step-3: Initializing sub-key S . Sub-key S of size $t=2(r+1)$ is initialized using magic constants P and Q .

$$S[0] = P$$

for $i = 1$ to $2(r+1)-1$

$$S[i] = S[i-1] + Q$$

Step-4: Sub-key mixing. The RC5 encryption algorithm uses Sub key S . L is merely, a temporary array formed on the basis of user entered secret key. Mix in user's secret key with S and L .

$$i = j = 0$$

$$A = B = 0$$

do $3 * \max(t, c)$ times:

$$A = S[i] = (S[i] + A + B) \lll 3$$

$$B = L[j] = (L[j] + A + B) \lll (A + B)$$

$$i = (i + 1) \% t$$

$$j = (j + 1) \% c$$

Step-5: Encryption. We divide the input plain text block into two registers A and B each of size w bits. After undergoing the encryption process the result of A and B together forms the cipher text block. RC5 Encryption Algorithm:

1. One time initialization of plain text blocks A and B by adding $S[0]$ and $S[1]$ to A and B respectively. These operations are mod 2^w .
2. XOR A and B . $A=A \oplus B$
3. Cyclic left shift new value of A by B bits.
4. Add $S[2*i]$ to the output of previous step. This is the new value of A .
5. XOR B with new value of A and store in B .
6. Cyclic left shift new value of B by A bits.
7. Add $S[2*i+1]$ to the output of previous step. This is the new value of B .
8. Repeat entire procedure (except one time initialization) r times.

$$A = A + S[0]$$

$$B = B + S[1]$$

for $i = 1$ to r do:

```
A = ((A ^ B) <<< B) + S[2 * i]
B = ((B ^ A) <<< A) + S[2 * i + 1]
```

```
return A, B
```

Alternatively, RC5 Decryption can be defined as:

for i = r down to 1 do:

```
B = ((B - S[2 * i + 1]) >>> A) ^ A
A = ((A - S[2 * i]) >>> B) ^ B
```

```
B = B - S[1]
```

```
A = A - S[0]
```

```
return A, B
```

Advantages:

High level of security: RC5 is designed to provide a high level of security against attacks, including brute-force attacks and differential cryptanalysis. It uses a variable-length key and can operate on block sizes of up to 256 bits, making it difficult for attackers to break the encryption.

Fast encryption and decryption: RC5 is known for its fast encryption and decryption speeds. It uses simple mathematical operations such as modular arithmetic and bit shifting, which can be efficiently implemented on modern CPUs and hardware.

Flexible key length: RC5 allows for a variable-length key, which can range from 0 to 2040 bits. This flexibility allows users to choose a key length that suits their security needs and resources.

Disadvantages:

Vulnerable to side-channel attacks: RC5 is vulnerable to side-channel attacks, such as timing attacks and power analysis attacks. These attacks exploit information leaked through the implementation of the algorithm, rather than attacking the algorithm itself.

Limited adoption: RC5 is not widely adopted in practice compared to other encryption algorithms, such as AES. This means that there may be fewer resources and tools available to support RC5 in various applications and systems.

Patent issues: RC5 was subject to a patent held by RSA Security, which limited its adoption and use in commercial applications. Although the patent has since expired, it may have contributed to the limited adoption of RC5 compared to other encryption algorithms.

2.8] Blowfish

Blowfish is an encryption technique designed by **Bruce Schneier** in 1993 as an alternative to [DES Encryption Technique](#). It is significantly faster than DES and provides a good encryption rate with no effective [cryptanalysis technique](#) found to date. It is one of the first, secure block cyphers not subject to any patents and hence freely available for anyone to use. It is symmetric block cipher algorithm.

1. **Block Size:** 64-bits
2. **Key Size:** 32-bits to 448-bits variable size
3. **number of subkeys:** 18 [P-array]
4. **number of rounds:** 16
5. **number of substitution boxes:** 4 [each having 512 entries of 32-bits each]

Blowfish Encryption Algorithm Diag

Lets see each step one by one:

Step1: Generation of subkeys:

- 18 subkeys $\{P[0] \dots P[17]\}$ are needed in both encryption as well as decryption process and the same subkeys are used for both the processes.
- These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.
- It is initialized with the digits of pi(?).
- The hexadecimal representation of each of the subkeys is given by:

$P[0] = \text{"243f6a88"}$

$P[1] = \text{"85a308d3"}$

.

.

.

$P[17] = \text{"8979fb1b"}$

32-bit hexadecimal representation of initial values of sub-keys

$P[0] : 243f6a88$	$P[9] : 38d01377$
$P[1] : 85a308d3$	$P[10] : be5466cf$
$P[2] : 13198a2e$	$P[11] : 34e90c6c$
$P[3] : 03707344$	$P[12] : c0ac29b7$
$P[4] : a4093822$	$P[13] : c97c50dd$
$P[5] : 299f31d0$	$P[14] : 3f84d5b5$
$P[6] : 082efa98$	$P[15] : b5470917$
$P[7] : ec4e6c89$	$P[16] : 9216d5d9$
$P[8] : 452821e6$	$P[17] : 8979fb1b$

- Now each of the subkey is changed with respect to the input key as:

$P[0] = P[0] \text{ xor } 1\text{st } 32\text{-bits of input key}$

$P[1] = P[1] \text{ xor } 2\text{nd } 32\text{-bits of input key}$

- .
- .
- .
- .

$P[i] = P[i] \text{ xor } (i+1)\text{th 32-bits of input key}$
(roll over to 1st 32-bits depending on the key length)

- .
- .
- .

$P[17] = P[17] \text{ xor } 18\text{th 32-bits of input key}$
(roll over to 1st 32-bits depending on key length)

The resultant P-array holds 18 subkeys that is used during the entire encryption process

Step2: initialise Substitution Boxes:

- 4 Substitution boxes(S-boxes) are needed $\{S[0] \dots S[4]\}$ in both encryption as well as decryption process with each S-box having 256 entries $\{S[i][0] \dots S[i][255], 0 \leq i \leq 4\}$ where each entry is 32-bit.
- It is initialized with the digits of pi(?) after initializing the P-array. [You may find the s-boxes in here!](#)

Step3: Encryption:

- The encryption function consists of two parts:
a. Rounds: The encryption consists of 16 rounds with each round(R_i) taking inputs the plain Text(P.T.) from previous round and corresponding subkey(P_i). The description of each round is as follows:

: Here the function “add” is addition modulo 2^{32} .

b. Post-processing: The output after the 16 rounds is processed as follows(Diag)

Below is a Java Program to demonstrate Blowfish encryption:

- Java

```
// Java Program to demonstrate Blowfish encryption

import java.util.*;

public class Main {

    // Substitution boxes each string is a 32 bit hexadecimal value.

    String S[][]

    = { { "d1310ba6", "98dfb5ac", "2ffd72db", "d01adfb7", "b8e1afed",
```

"6a267e96", "ba7c9045", "f12c7f99", "24a19947", "b3916cf7",
"0801f2e2", "858efc16", "636920d8", "71574e69", "a458fea3",
"f4933d7e", "0d95748f", "728eb658", "718bcd58", "82154aee",
"7b54a41d", "c25a59b5", "9c30d539", "2af26013", "c5d1b023",
"286085f0", "ca417918", "b8db38ef", "8e79dcb0", "603a180e",
"6c9e0e8b", "b01e8a3e", "d71577c1", "bd314b27", "78af2fda
"55605c60", "e65525f3", "aa55ab94", "57489862", "63e81440",
"55ca396a", "2aab10b6", "b4cc5c34", "1141e8ce", "a15486af",
"7c72e993", "b3ee1411", "636fbc2a", "2ba9c55d", "741831f6",
"ce5c3e16", "9b87931e", "afd6ba33", "6c24cf5c", "7a325381",
"28958677", "3b8f4898", "6b4bb9af", "c4bfe81b", "66282193",
"61d809cc", "fb21a991", "487cac60", "5dec8032", "ef845d5d",
"e98575b1", "dc262302", "eb651b88", "23893e81", "d396acc5",
"0f6d6ff3", "83f44239", "2e0b4482", "a4842004", "69c8f04a",
"9e1f9b5e", "21c66842", "f6e96c9a", "670c9c61", "abd388f0",
"6a51a0d2", "d8542f68", "960fa728", "ab5133a3", "6eef0b6c",
"137a3be4", "ba3bf050", "7efb2a98", "a1f1651d", "39af0176",
"66ca593e", "82430e88", "8cee8619", "456f9fb4", "7d84a5c3",
"3b8b5ebe", "e06f75d8", "85c12073", "401a449f", "56c16aa6",
"4ed3aa62", "363f7706", "1bfedf72", "429b023d", "37d0d724",
"d00a1248", "db0fead3", "49f1c09b", "075372c9", "80991b7b",

"25d479d8", "f6e8def7", "e3fe501a", "b6794c3b", "976ce0bd",
"04c006ba", "c1a94fb6", "409f60c4", "5e5c9ec2", "196a2463",
"68fb6faf", "3e6c53b5", "1339b2eb", "3b52ec6f", "6dfc511f",
"9b30952c", "cc814544", "af5ebd09", "bee3d004", "de334afd",
"660f2807", "192e4bb3", "c0cba857", "45c8740f", "d20b5f39",
"b9d3fbdb", "5579c0bd", "1a60320a", "d6a100c6", "402c7279",
"679f25fe", "fb1fa3cc", "8ea5e9f8", "db3222f8", "3c7516df",
"fd616b15", "2f501ec8", "ad0552ab", "323db5fa", "fd238760",
"53317b48", "3e00df82", "9e5c57bb", "ca6f8ca0", "1a87562e",
"df1769db", "d542a8f6", "287effc3", "ac6732c6", "8c4f5573",
"695b27b0", "bbca58c8", "e1ffa35d", "b8f011a0", "10fa3d98",
"fd2183b8", "4afcb56c", "2dd1d35b", "9a53e479", "b6f84565",
"d28e49bc", "4bfb9790", "e1ddf2da", "a4cb7e33", "62fb1341",
"cee4c6e8", "ef20cada", "36774c01", "d07e9efe", "2bf11fb4",
"95dbda4d", "ae909198", "eaad8e71", "6b93d5a0", "d08ed1d0",
"afc725e0", "8e3c5b2f", "8e7594b7", "8ff6e2fb", "f2122b64",
"8888b812", "900df01c", "4fad5ea0", "688fc31c", "d1cff191",
"b3a8c1ad", "2f2f2218", "be0e1777", "ea752dfe", "8b021fa1",
"e5a0cc0f", "b56f74e8", "18acf3d6", "ce89e299", "b4a84fe0",
"fd13e0b7", "7cc43b81", "d2ada8d9", "165fa266", "80957705",
"93cc7314", "211a1477", "e6ad2065", "77b5fa86", "c75442f5",

"fb9d35cf", "ebcdaf0c", "7b3e89a0", "d6411bd3", "ae1e7e49",
"00250e2d", "2071b35e", "226800bb", "57b8e0af", "2464369b",
"f009b91e", "5563911d", "59dfa6aa", "78c14389", "d95a537f",
"207d5ba2", "02e5b9c5", "83260376", "6295cfa9", "11c81968",
"4e734a41", "b3472dca", "7b14a94a", "1b510052", "9a532915",
"d60f573f", "bc9bc6e4", "2b60a476", "81e67400", "08ba6fb5",
"571be91f", "f296ec6b", "2a0dd915", "b6636521", "e7b9f9b6",
"ff34052e", "c5855664", "53b02d5d", "a99f8fa1", "08ba4799",
"6e85076a" },

{ "4b7a70e9", "b5b32944", "db75092e", "c4192623", "ad6ea6b0",
"49a7df7d", "9cee60b8", "8fedb266", "ecaa8c71", "699a17ff",
"5664526c", "c2b19ee1", "193602a5", "75094c29", "a0591340",
"e4183a3e", "3f54989a", "5b429d65", "6b8fe4d6", "99f73fd6",
"a1d29c07", "efe830f5", "4d2d38e6", "f0255dc1", "4cdd2086",
"8470eb26", "6382e9c6", "021ecc5e", "09686b3f", "3ebaefc9",
"3c971814", "6b6a70a1", "687f3584", "52a0e286", "b79c5305",
"aa500737", "3e07841c", "7fdeae5c", "8e7d44ec", "5716f2b8",
"b03ada37", "f0500c0d", "f01c1f04", "0200b3ff", "ae0cf51a",
"3cb574b2", "25837a58", "dc0921bd", "d19113f9", "7ca92ff6",
"94324773", "22f54701", "3ae5e581", "37c2dadac", "c8b57634",
"9af3dda7", "a9446146", "0fd0030e", "ecc8c73e", "a4751e41",

"e238cd99", "3bea0e2f", "3280bba1", "183eb331", "4e548b38",
"4f6db908", "6f420d03", "f60a04bf", "2cb81290", "24977c79",
"5679b072", "bcaf89af", "de9a771f", "d9930810", "b38bae12",
"dccf3f2e", "5512721f", "2e6b7124", "501adde6", "9f84cd87",
"7a584718", "7408da17", "bc9f9abc", "e94b7d8c", "ec7aec3a",
"db851dfa", "63094366", "c464c3d2", "ef1c1847", "3215d908",
"dd433b37", "24c2ba16", "12a14d43", "2a65c451", "50940002",
"133ae4dd", "71dff89e", "10314e55", "81ac77d6", "5f11199b",
"043556f1", "d7a3c76b", "3c11183b", "5924a509", "f28fe6ed",
"97f1fbfa", "9ebabf2c", "1e153c6e", "86e34570", "eae96fb1",
"860e5e0a", "5a3e2ab3", "771fe71c", "4e3d06fa", "2965dcb9",
"99e71d0f", "803e89d6", "5266c825", "2e4cc978", "9c10b36a",
"c6150eba", "94e2ea78", "a5fc3c53", "1e0a2df4", "f2f74ea7",
"361d2b3d", "1939260f", "19c27960", "5223a708", "f71312b6",
"ebadfe6e", "eac31f66", "e3bc4595", "a67bc883", "b17f37d1",
"018cff28", "c332ddef", "be6c5aa5", "65582185", "68ab9802",
"eecea50f", "db2f953b", "2aef7dad", "5b6e2f84", "1521b628",
"29076170", "ecdd4775", "619f1510", "13cca830", "eb61bd96",
"0334fe1e", "aa0363cf", "b5735c90", "4c70a239", "d59e9e0b",
"cbaade14", "eccc86bc", "60622ca7", "9cab5cab", "b2f3846e",
"648b1eaf", "19bdf0ca", "a02369b9", "655abb50", "40685a32",

"3c2ab4b3", "319ee9d5", "c021b8f7", "9b540b19", "875fa099",
"95f7997e", "623d7da8", "f837889a", "97e32d77", "11ed935f",
"16681281", "0e358829", "c7e61fd6", "96dedfa1", "7858ba99",
"57f584a5", "1b227263", "9b83c3ff", "1ac24696", "cdb30aeb",
"532e3054", "8fd948e4", "6dbc3128", "58ebf2ef", "34c6ffea",
"fe28ed61", "ee7c3c73", "5d4a14d9", "e864b7e3", "42105d14",
"203e13e0", "45eee2b6", "a3aaabea", "db6c4f15", "facb4fd0",
"c742f442", "ef6abbb5", "654f3b1d", "41cd2105", "d81e799e",
"86854dc7", "e44b476a", "3d816250", "cf62a1f2", "5b8d2646",
"fc8883a0", "c1c7b6a3", "7f1524c3", "69cb7492", "47848a0b",
"5692b285", "095bbbf00", "ad19489d", "1462b174", "23820e00",
"58428d2a", "0c55f5ea", "1dadf43e", "233f7061", "3372f092",
"8d937e41", "d65fecf1", "6c223bdb", "7cde3759", "cbee7460",
"4085f2a7", "ce77326e", "a6078084", "19f8509e", "e8efd855",
"61d99735", "a969a7aa", "c50c06c2", "5a04abfc", "800bcadc",
"9e447a2e", "c3453484", "fdd56705", "0e1e9ec9", "db73dbd3",
"105588cd", "675fda79", "e3674340", "c5c43465", "713e38d8",
"3d28f89e", "f16dff20", "153e21e7", "8fb03d4a", "e6e39f2b",
"db83adf7" },
{ "e93d5a68", "948140f7", "f64c261c", "94692934", "411520f7",
"7602d4f7", "bcf46b2e", "d4a20068", "d4082471", "3320f46a",

"43b7d4b7", "500061af", "1e39f62e", "97244546", "14214f74",
"bf8b8840", "4d95fc1d", "96b591af", "70f4ddd3", "66a02f45",
"bfbc09ec", "03bd9785", "7fac6dd0", "31cb8504", "96eb27b3",
"55fd3941", "da2547e6", "abca0a9a", "28507825", "530429f4",
"0a2c86da", "e9b66dfb", "68dc1462", "d7486900", "680ec0a4",
"27a18dee", "4f3ffea2", "e887ad8c", "b58ce006", "7af4d6b6",
"aace1e7c", "d3375fec", "ce78a399", "406b2a42", "20fe9e35",
"d9f385b9", "ee39d7ab", "3b124e8b", "1dc9faf7", "4b6d1856",
"26a36631", "eae397b2", "3a6efa74", "dd5b4332", "6841e7f7",
"ca7820fb", "fb0af54e", "d8feb397", "454056ac", "ba489527",
"55533a3a", "20838d87", "fe6ba9b7", "d096954b", "55a867bc",
"a1159a58", "cca92963", "99e1db33", "a62a4a56", "3f3125f9",
"5ef47e1c", "9029317c", "fdf8e802", "04272f70", "80bb155c",
"05282ce3", "95c11548", "e4c66d22", "48c1133f", "c70f86dc",
"07f9c9ee", "41041f0f", "404779a4", "5d886e17", "325f51eb",
"d59bc0d1", "f2bcc18f", "41113564", "257b7834", "602a9c60",
"dff8e8a3", "1f636c1b", "0e12b4c2", "02e1329e", "af664fd1",
"cad18115", "6b2395e0", "333e92e1", "3b240b62", "eebeb922",
"85b2a20e", "e6ba0d99", "de720c8c", "2da2f728", "d0127845",
"95b794fd", "647d0862", "e7ccf5f0", "5449a36f", "877d48fa",
"c39dfd27", "f33e8d1e", "0a476341", "992eff74", "3a6f6eab",

"f4f8fd37", "a812dc60", "a1ebddf8", "991be14c", "db6e6b0d",
"c67b5510", "6d672c37", "2765d43b", "dcd0e804", "f1290dc7",
"cc00ffa3", "b5390f92", "690fed0b", "667b9ffb", "cedb7d9c",
"a091cf0b", "d9155ea3", "bb132f88", "515bad24", "7b9479bf",
"763bd6eb", "37392eb3", "cc115979", "8026e297", "f42e312d",
"6842ada7", "c66a2b3b", "12754ccc", "782ef11c", "6a124237",
"b79251e7", "06a1bbe6", "4bfb6350", "1a6b1018", "11caedfa",
"3d25bdd8", "e2e1c3c9", "44421659", "0a121386", "d90cec6e",
"d5abea2a", "64af674e", "da86a85f", "bebf988", "64e4c3fe",
"9dbc8057", "f0f7c086", "60787bf8", "6003604d", "d1fd8346",
"f6381fb0", "7745ae04", "d736fccc", "83426b33", "f01eab71",
"b0804187", "3c005e5f", "77a057be", "bde8ae24", "55464299",
"bf582e61", "4e58f48f", "f2ddfa2", "f474ef38", "8789bdc2",
"5366f9c3", "c8b38e74", "b475f255", "46fcd9b9", "7aeb2661",
"8b1ddf84", "846a0e79", "915f95e2", "466e598e", "20b45770",
"8cd55591", "c902de4c", "b90bace1", "bb8205d0", "11a86248",
"7574a99e", "b77f19b6", "e0a9dc09", "662d09a1", "c4324633",
"e85a1f02", "09f0be8c", "4a99a025", "1d6efe10", "1ab93d1d",
"0ba5a4df", "a186f20f", "2868f169", "dcb7da83", "573906fe",
"a1e2ce9b", "4fcd7f52", "50115e01", "a70683fa", "a002b5c4",
"0de6d027", "9af88c27", "773f8641", "c3604c06", "61a806b5",

"f0177a28", "c0f586e0", "006058aa", "30dc7d62", "11e69ed7",
"2338ea63", "53c2dd94", "c2c21634", "bbcbee56", "90bcb6de",
"ebfc7da1", "ce591d76", "6f05e409", "4b7c0188", "39720a3d",
"7c927c24", "86e3725f", "724d9db9", "1ac15bb4", "d39eb8fc",
"ed545578", "08fca5b5", "d83d7cd3", "4dad0fc4", "1e50ef5e",
"b161e6f8", "a28514d9", "6c51133c", "6fd5c7e7", "56e14ec4",
"362abfce", "ddc6c837", "d79a3234", "92638212", "670efa8e",
"406000e0" },
{ "3a39ce37", "d3faf5cf", "abc27737", "5ac52d1b", "5cb0679e",
"4fa33742", "d3822740", "99bc9bbe", "d5118e9d", "bf0f7315",
"d62d1c7e", "c700c47b", "b78c1b6b", "21a19045", "b26eb1be",
"6a366eb4", "5748ab2f", "bc946e79", "c6a376d2", "6549c2c8",
"530ff8ee", "468dde7d", "d5730a1d", "4cd04dc6", "2939bbdb",
"a9ba4650", "ac9526e8", "be5ee304", "a1fad5f0", "6a2d519a",
"63ef8ce2", "9a86ee22", "c089c2b8", "43242ef6", "a51e03aa",
"9cf2d0a4", "83c061ba", "9be96a4d", "8fe51550", "ba645bd6",
"2826a2f9", "a73a3ae1", "4ba99586", "ef5562e9", "c72fef3",
"f752f7da", "3f046f69", "77fa0a59", "80e4a915", "87b08601",
"9b09e6ad", "3b3ee593", "e990fd5a", "9e34d797", "2cf0b7d9",
"022b8b51", "96d5ac3a", "017da67d", "d1cf3ed6", "7c7d2d28",
"1f9f25cf", "adf2b89b", "5ad6b472", "5a88f54c", "e029ac71",

"e019a5e6", "47b0acfd", "ed93fa9b", "e8d3c48d", "283b57cc",
"f8d56629", "79132e28", "785f0191", "ed756055", "f7960e44",
"e3d35e8c", "15056dd4", "88f46dba", "03a16125", "0564f0bd",
"c3eb9e15", "3c9057a2", "97271aec", "a93a072a", "1b3f6d9b",
"1e6321f5", "f59c66fb", "26dcf319", "7533d928", "b155fdf5",
"03563482", "8aba3cbb", "28517711", "c20ad9f8", "abcc5167",
"ccad925f", "4de81751", "3830dc8e", "379d5862", "9320f991",
"ea7a90c2", "fb3e7bce", "5121ce64", "774fbe32", "a8b6e37e",
"c3293d46", "48de5369", "6413e680", "a2ae0810", "dd6db224",
"69852dfd", "09072166", "b39a460a", "6445c0dd", "586cdecf",
"1c20c8ae", "5bbef7dd", "1b588d40", "ccd2017f", "6bb4e3bb",
"dda26a7e", "3a59ff45", "3e350a44", "bcb4cdd5", "72eacea8",
"fa6484bb", "8d6612ae", "bf3c6f47", "d29be463", "542f5d9e",
"aec2771b", "f64e6370", "740e0d8d", "e75b1357", "f8721671",
"af537d5d", "4040cb08", "4eb4e2cc", "34d2466a", "0115af84",
"e1b00428", "95983a1d", "06b89fb4", "ce6ea048", "6f3f3b82",
"3520ab82", "011a1d4b", "277227f8", "611560b1", "e7933fdc",
"bb3a792b", "344525bd", "a08839e1", "51ce794b", "2f32c9b7",
"a01fbac9", "e01cc87e", "bcc7d1f6", "cf0111c3", "a1e8aac7",
"1a908749", "d44fbd9a", "d0dadecb", "d50ada38", "0339c32a",
"c6913667", "8df9317c", "e0b12b4f", "f79e59b7", "43f5bb3a",

```
"f2d519ff", "27d9459c", "bf97222c", "15e6fc2a", "0f91fc71",  
  
"9b941525", "fae59361", "ceb69ceb", "c2a86459", "12baa8d1",  
  
"b6c1075e", "e3056a0c", "10d25065", "cb03a442", "e0ec6e0e",  
  
"1698db3b", "4c98a0be", "3278e964", "9f1f9532", "e0d392df",  
  
"d3a0342b", "8971f21e", "1b0a7441", "4ba3348c", "c5be7120",  
  
"c37632d8", "df359f8d", "9b992f2e", "e60b6f47", "0fe3f11d",  
  
"e54cda54", "1edad891", "ce6279cf", "cd3e7e6f", "1618b166",  
  
"fd2c1d05", "848fd2c5", "f6fb2299", "f523f357", "a6327623",  
  
"93a83531", "56cccd02", "acf08162", "5a75ebb5", "6e163697",  
  
"88d273cc", "de966292", "81b949d0", "4c50901b", "71c65614",  
  
"e6c6c7bd", "327a140a", "45e1d006", "c3f27b9a", "c9aa53fd",  
  
"62a80f00", "bb25bfe2", "35bdd2f6", "71126905", "b2040222",  
  
"b6cbcf7c", "cd769c2b", "53113ec0", "1640e3d3", "38abbd60",  
  
"2547adf0", "ba38209c", "f746ce76", "77afa1c5", "20756060",  
  
"85cbfe4e", "8ae88dd8", "7aaaf9b0", "4cf9aa7e", "1948c25c",  
  
"02fb8a8c", "01c36ae4", "d6ebe1f9", "90d4f869", "a65cdea0",  
  
"3f09252d", "c208e69f", "b74e6132", "ce77e25b", "578fdfe3",  
  
"3ac372e6" } };
```

```
// Subkeys initialisation with digits of pi.
```

```
String P[] = { "243f6a88", "85a308d3", "13198a2e", "03707344", "a4093822",
```

```
"299f31d0", "082efa98", "ec4e6c89", "452821e6", "38d01377",
```

```
"be5466cf", "34e90c6c", "c0ac29b7", "c97c50dd", "3f84d5b5",
```

```
"b5470917", "9216d5d9", "8979fb1b" };
```

```
// to store  $2^{32}$ (for addition modulo  $2^{32}$ ).
```

```
long modVal = 1;
```

```
// to convert hexadecimal to binary.
```

```
private String hexToBin(String plainText)
```

```
{
```

```
    String binary = "";
```

```
    Long num;
```

```
    String binary4B;
```

```
    int n = plainText.length();
```

```
    for (int i = 0; i < n; i++) {
```

```
        num = Long.parseUnsignedLong(
```

```
            plainText.charAt(i) + "", 16);
```

```
        binary4B = Long.toBinaryString(num);
```

```
// each value in hexadecimal is 4 bits in binary.

binary4B = "0000" + binary4B;

binary4B = binary4B.substring(binary4B.length() - 4);

binary += binary4B;

}

return binary;

}
```

```
// convert from binary to hexadecimal.

private String binToHex(String plainText)

{

    long num = Long.parseUnsignedLong(plainText, 2);

    String hexa = Long.toHexString(num);

    while (hexa.length() < (plainText.length() / 4))

        // maintain output length same length

        // as input by appending leading zeroes.

        hexa = "0" + hexa;
```

```

        return hexa;
    }

    // xor two hexadecimal strings of the same length.

    private String xor(String a, String b)
    {
        a = hexToBin(a);

        b = hexToBin(b);

        String ans = "";

        for (int i = 0; i < a.length(); i++)

            ans += (char)((((a.charAt(i) - '0')

                            ^ (b.charAt(i) - '0'))

                        + '0'));

        ans = binToHex(ans);

        return ans;
    }

    // addition modulo 2^32 of two hexadecimal strings.

    private String addBin(String a, String b)

```

```

{

    String ans = "";

    long n1 = Long.parseUnsignedLong(a, 16);

    long n2 = Long.parseUnsignedLong(b, 16);

    n1 = (n1 + n2) % modVal;

    ans = Long.toHexString(n1);

    ans = "00000000" + ans;

    return ans.substring(ans.length() - 8);

}

```

// function F explained above.

```
private String f(String plainText)
```

```

{

    String a[] = new String[4];

    String ans = "";

    for (int i = 0; i < 8; i += 2) {

        // the column number for S-box

        // is 8-bit value(8*4 = 32 bit plain text)

        long col

        = Long.parseUnsignedLong(

            hexToBin(

```



```

        plainText

        .substring(i, i + 2)),

    2);

    a[i / 2] = S[i / 2][(int)col];

}

ans = addBin(a[0], a[1]);

ans = xor(ans, a[2]);

ans = addBin(ans, a[3]);

return ans;

}

// generate subkeys.

private void keyGenerate(String key)

{

    int j = 0;

    for (int i = 0; i < P.length; i++) {

        // xor-ing 32-bit parts of the key

        // with initial subkeys.

        P[i] = xor(P[i], key.substring(j, j + 8));

```

```

        System.out.println("subkey "

                               + (i + 1) + ": "

                               + P[i]);

        j = (j + 8) % key.length();

    }

}

// round function

private String round(int time, String plainText)

{

    String left, right;

    left = plainText.substring(0, 8);

    right = plainText.substring(8, 16);

    left = xor(left, P[time]);

    // output from F function

    String fOut = f(left);

    right = xor(fOut, right);

```

```
System.out.println(

    "round " + time + ": "

    + right + left);

// swap left and right

return right + left;

}

// encryption

private String encrypt(String plainText)

{

    for (int i = 0; i < 16; i++)

        plainText = round(i, plainText);

// post processing

String right = plainText.substring(0, 8);

String left = plainText.substring(8, 16);

right = xor(right, P[16]);

left = xor(left, P[17]);
```

```
        return left + right;

    }

    Main()

    {

        // storing 2^32 in modVal

        //( << 1 is equivalent to multiply by 2)

        for (int i = 0; i < 32; i++)

            modVal = modVal << 1;


        String plainText = "123456abcd132536";

        String key = "aabb09182736ccdd";


        keyGenerate(key);


        System.out.println("-----Encryption-----");

        String cipherText = encrypt(plainText);

        System.out.println("Cipher Text: " + cipherText);

    }
```

```
public static void main(String args[])

{

    new Main();

}

}
```

Output

subkey 1: 8e846390

subkey 2: a295c40e

subkey 3: b9a28336

subkey 4: 2446bf99

subkey 5: 0eb2313a

subkey 6: 0ea9fd0d

subkey 7: a295f380

subkey 8: cb78a054

subkey 9: ef9328fe

subkey 10: 1fe6dfaa

subkey 11: 14ef6fd7

subkey 12: 13dfc0b1

subkey 13: 6a1720af

subkey 14: ee4a9c00

subkey 15: 953fdcad

subkey 16: 9271c5ca

subkey 17: 38addcc1

subkey 18: ae4f37c6

-----Encryption-----

round 0: 77b3ba639cb0353b

round 1: 0cc7d63fd5267e6d
round 2: c799728ab5655509
round 3: 69612395e3dfcd13
round 4: f3f5b74b67d312af
round 5: 52023d4efd5c4a46
round 6: 5b785180f097cece
round 7: cc946d119000f1d4
round 8: 6af47a4b230745ef
round 9: 9fb82cc57512a5e1
round 10: 1106c1ab8b574312
round 11: 7d7a616502d9011a
round 12: 81e9ce71176d41ca
round 13: 9727e50a6fa35271
round 14: eb761e34021839a7
round 15: 0599d9367907dbfe
Cipher Text: d748ec383d3405f7

Decryption

The decryption process is similar to that of encryption and the subkeys are used in reverse{P[17] – P[0]}. The entire decryption process can be elaborated as:

Lets see each step one by one:

Step1: Generation of subkeys:

- 18 subkeys{P[0]...P[17]} are needed in decryption process.
 - These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.
- It is initialized with the digits of pi(?).
- The hexadecimal representation of each of the subkeys is given by:
P[0] = "243f6a88"
P[1] = "85a308d3"

.

.

.

P[17] = "8979fb1b"

Note: See encryption for the initial values of P-array.

- Now each of the subkeys is changed with respect to the input key as:

P[0] = P[0] xor 1st 32-bits of input key

P[1] = P[1] xor 2nd 32-bits of input key

.

.

.
 $P[i] = P[i] \text{ xor } (i+1)\text{th 32-bits of input key}$
(roll over to 1st 32-bits depending on the key length)

.
.
.
 $P[17] = P[17] \text{ xor } 18\text{th 32-bits of input key}$
(roll over to 1st 32-bits depending on key length)

The resultant P-array holds 18 subkeys that is used during the entire encryption process

Step2: initialize Substitution Boxes:

- 4 Substitution boxes(S-boxes) are needed $\{S[0] \dots S[4]\}$ in both encryption as well as decryption process with each S-box having 256 entries $\{S[i][0] \dots S[i][255], 0 \leq i \leq 4\}$ where each entry is 32-bit.
- It is initialized with the digits of pi(?) after initializing the P-array. [You may find the s-boxes in here !](#)

Step3: Decryption:

- The Decryption function also consists of two parts:
 1. **Rounds:** The decryption also consists of 16 rounds with each round(R_i)(as explained above) taking inputs the cipherText(C.T.) from previous round and corresponding subkey($P[17-i]$)(i.e for decryption the subkeys are used in reverse).
 2. **Post-processing:** The output after the 16 rounds is processed as follows:

Below is a Java program to demonstrate decryption

- Java

```
// Java program to demonstrate  
  
// Blowfish decryption Algorithm  
  
import java.util.*;  
  
public class Main {
```

// Substitution boxes each string is a 32 bit hexadecimal value.

String S[][]

```
= { { "d1310ba6", "98dfb5ac", "2ffd72db", "d01adfb7", "b8e1afed",  
      "6a267e96", "ba7c9045", "f12c7f99", "24a19947", "b3916cf7",  
      "0801f2e2", "858efc16", "636920d8", "71574e69", "a458fea3",  
      "f4933d7e", "0d95748f", "728eb658", "718bcd58", "82154aee",  
      "7b54a41d", "c25a59b5", "9c30d539", "2af26013", "c5d1b023",  
      "286085f0", "ca417918", "b8db38ef", "8e79dcb0", "603a180e",  
      "6c9e0e8b", "b01e8a3e", "d71577c1", "bd314b27", "78af2fda",  
      "55605c60", "e65525f3", "aa55ab94", "57489862", "63e81440",  
      "55ca396a", "2aab10b6", "b4cc5c34", "1141e8ce", "a15486af",  
      "7c72e993", "b3ee1411", "636fbc2a", "2ba9c55d", "741831f6",  
      "ce5c3e16", "9b87931e", "afd6ba33", "6c24cf5c", "7a325381",  
      "28958677", "3b8f4898", "6b4bb9af", "c4bfe81b", "66282193",  
      "61d809cc", "fb21a991", "487cac60", "5dec8032", "ef845d5d",  
      "e98575b1", "dc262302", "eb651b88", "23893e81", "d396acc5",  
      "0f6d6ff3", "83f44239", "2e0b4482", "a4842004", "69c8f04a",  
      "9e1f9b5e", "21c66842", "f6e96c9a", "670c9c61", "abd388f0",  
      "6a51a0d2", "d8542f68", "960fa728", "ab5133a3", "6eef0b6c",  
      "137a3be4", "ba3bf050", "7efb2a98", "a1f1651d", "39af0176",
```


"66ca593e", "82430e88", "8cee8619", "456f9fb4", "7d84a5c3",
"3b8b5ebe", "e06f75d8", "85c12073", "401a449f", "56c16aa6",
"4ed3aa62", "363f7706", "1bfedf72", "429b023d", "37d0d724",
"d00a1248", "db0fead3", "49f1c09b", "075372c9", "80991b7b",
"25d479d8", "f6e8def7", "e3fe501a", "b6794c3b", "976ce0bd",
"04c006ba", "c1a94fb6", "409f60c4", "5e5c9ec2", "196a2463",
"68fb6faf", "3e6c53b5", "1339b2eb", "3b52ec6f", "6dfc511f",
"9b30952c", "cc814544", "af5ebd09", "bee3d004", "de334afd",
"660f2807", "192e4bb3", "c0cba857", "45c8740f", "d20b5f39",
"b9d3fbdb", "5579c0bd", "1a60320a", "d6a100c6", "402c7279",
"679f25fe", "fb1fa3cc", "8ea5e9f8", "db3222f8", "3c7516df",
"fd616b15", "2f501ec8", "ad0552ab", "323db5fa", "fd238760",
"53317b48", "3e00df82", "9e5c57bb", "ca6f8ca0", "1a87562e",
"df1769db", "d542a8f6", "287effc3", "ac6732c6", "8c4f5573",
"695b27b0", "bbca58c8", "e1ffa35d", "b8f011a0", "10fa3d98",
"fd2183b8", "4afcb56c", "2dd1d35b", "9a53e479", "b6f84565",
"d28e49bc", "4bfb9790", "e1ddf2da", "a4cb7e33", "62fb1341",
"cee4c6e8", "ef20cada", "36774c01", "d07e9efe", "2bf11fb4",
"95dbda4d", "ae909198", "eaad8e71", "6b93d5a0", "d08ed1d0",
"afc725e0", "8e3c5b2f", "8e7594b7", "8ff6e2fb", "f2122b64",
"8888b812", "900df01c", "4fad5ea0", "688fc31c", "d1cff191",

"b3a8c1ad", "2f2f2218", "be0e1777", "ea752dfe", "8b021fa1",
"e5a0cc0f", "b56f74e8", "18acf3d6", "ce89e299", "b4a84fe0",
"fd13e0b7", "7cc43b81", "d2ada8d9", "165fa266", "80957705",
"93cc7314", "211a1477", "e6ad2065", "77b5fa86", "c75442f5",
"fb9d35cf", "ebcdaf0c", "7b3e89a0", "d6411bd3", "ae1e7e49",
"00250e2d", "2071b35e", "226800bb", "57b8e0af", "2464369b",
"f009b91e", "5563911d", "59dfa6aa", "78c14389", "d95a537f",
"207d5ba2", "02e5b9c5", "83260376", "6295cfa9", "11c81968",
"4e734a41", "b3472dca", "7b14a94a", "1b510052", "9a532915",
"d60f573f", "bc9bc6e4", "2b60a476", "81e67400", "08ba6fb5",
"571be91f", "f296ec6b", "2a0dd915", "b6636521", "e7b9f9b6",
"ff34052e", "c5855664", "53b02d5d", "a99f8fa1", "08ba4799",
"6e85076a" },

{ "4b7a70e9", "b5b32944", "db75092e", "c4192623", "ad6ea6b0",
"49a7df7d", "9cee60b8", "8fedb266", "ecaa8c71", "699a17ff",
"5664526c", "c2b19ee1", "193602a5", "75094c29", "a0591340",
"e4183a3e", "3f54989a", "5b429d65", "6b8fe4d6", "99f73fd6",
"a1d29c07", "efe830f5", "4d2d38e6", "f0255dc1", "4cdd2086",
"8470eb26", "6382e9c6", "021ecc5e", "09686b3f", "3ebaefc9",
"3c971814", "6b6a70a1", "687f3584", "52a0e286", "b79c5305",
"aa500737", "3e07841c", "7fdeae5c", "8e7d44ec", "5716f2b8",

"b03ada37", "f0500c0d", "f01c1f04", "0200b3ff", "ae0cf51a",
"3cb574b2", "25837a58", "dc0921bd", "d19113f9", "7ca92ff6",
"94324773", "22f54701", "3ae5e581", "37c2dad", "c8b57634",
"9af3dda7", "a9446146", "0fd0030e", "ecc8c73e", "a4751e41",
"e238cd99", "3bea0e2f", "3280bba1", "183eb331", "4e548b38",
"4f6db908", "6f420d03", "f60a04bf", "2cb81290", "24977c79",
"5679b072", "bc9f89af", "de9a771f", "d9930810", "b38bae12",
"dccf3f2e", "5512721f", "2e6b7124", "501adde6", "9f84cd87",
"7a584718", "7408da17", "bc9f9abc", "e94b7d8c", "ec7aec3a",
"db851dfa", "63094366", "c464c3d2", "ef1c1847", "3215d908",
"dd433b37", "24c2ba16", "12a14d43", "2a65c451", "50940002",
"133ae4dd", "71dff89e", "10314e55", "81ac77d6", "5f11199b",
"043556f1", "d7a3c76b", "3c11183b", "5924a509", "f28fe6ed",
"97f1fbfa", "9ebabf2c", "1e153c6e", "86e34570", "eae96fb1",
"860e5e0a", "5a3e2ab3", "771fe71c", "4e3d06fa", "2965dcb9",
"99e71d0f", "803e89d6", "5266c825", "2e4cc978", "9c10b36a",
"c6150eba", "94e2ea78", "a5fc3c53", "1e0a2df4", "f2f74ea7",
"361d2b3d", "1939260f", "19c27960", "5223a708", "f71312b6",
"ebadfe6e", "eac31f66", "e3bc4595", "a67bc883", "b17f37d1",
"018cff28", "c332ddef", "be6c5aa5", "65582185", "68ab9802",
"eecea50f", "db2f953b", "2aef7dad", "5b6e2f84", "1521b628",

"29076170", "ecdd4775", "619f1510", "13cca830", "eb61bd96",
"0334fe1e", "aa0363cf", "b5735c90", "4c70a239", "d59e9e0b",
"cbaade14", "eccc86bc", "60622ca7", "9cab5cab", "b2f3846e",
"648b1eaf", "19bdf0ca", "a02369b9", "655abb50", "40685a32",
"3c2ab4b3", "319ee9d5", "c021b8f7", "9b540b19", "875fa099",
"95f7997e", "623d7da8", "f837889a", "97e32d77", "11ed935f",
"16681281", "0e358829", "c7e61fd6", "96dedfa1", "7858ba99",
"57f584a5", "1b227263", "9b83c3ff", "1ac24696", "cdb30aeb",
"532e3054", "8fd948e4", "6dbc3128", "58ebf2ef", "34c6ffea",
"fe28ed61", "ee7c3c73", "5d4a14d9", "e864b7e3", "42105d14",
"203e13e0", "45eee2b6", "a3aaabea", "db6c4f15", "facb4fd0",
"c742f442", "ef6abbb5", "654f3b1d", "41cd2105", "d81e799e",
"86854dc7", "e44b476a", "3d816250", "cf62a1f2", "5b8d2646",
"fc8883a0", "c1c7b6a3", "7f1524c3", "69cb7492", "47848a0b",
"5692b285", "095bbf00", "ad19489d", "1462b174", "23820e00",
"58428d2a", "0c55f5ea", "1dadf43e", "233f7061", "3372f092",
"8d937e41", "d65fecf1", "6c223bdb", "7cde3759", "cbee7460",
"4085f2a7", "ce77326e", "a6078084", "19f8509e", "e8efd855",
"61d99735", "a969a7aa", "c50c06c2", "5a04abfc", "800bcadc",
"9e447a2e", "c3453484", "fdd56705", "0e1e9ec9", "db73dbd3",
"105588cd", "675fda79", "e3674340", "c5c43465", "713e38d8",

"3d28f89e", "f16dff20", "153e21e7", "8fb03d4a", "e6e39f2b",
"db83adf7" },
{ "e93d5a68", "948140f7", "f64c261c", "94692934", "411520f7",
"7602d4f7", "bcf46b2e", "d4a20068", "d4082471", "3320f46a",
"43b7d4b7", "500061af", "1e39f62e", "97244546", "14214f74",
"bf8b8840", "4d95fc1d", "96b591af", "70f4ddd3", "66a02f45",
"bfbc09ec", "03bd9785", "7fac6dd0", "31cb8504", "96eb27b3",
"55fd3941", "da2547e6", "abca0a9a", "28507825", "530429f4",
"0a2c86da", "e9b66dfb", "68dc1462", "d7486900", "680ec0a4",
"27a18dee", "4f3ffea2", "e887ad8c", "b58ce006", "7af4d6b6",
"aace1e7c", "d3375fec", "ce78a399", "406b2a42", "20fe9e35",
"d9f385b9", "ee39d7ab", "3b124e8b", "1dc9faf7", "4b6d1856",
"26a36631", "eae397b2", "3a6efa74", "dd5b4332", "6841e7f7",
"ca7820fb", "fb0af54e", "d8feb397", "454056ac", "ba489527",
"55533a3a", "20838d87", "fe6ba9b7", "d096954b", "55a867bc",
"a1159a58", "cca92963", "99e1db33", "a62a4a56", "3f3125f9",
"5ef47e1c", "9029317c", "fdf8e802", "04272f70", "80bb155c",
"05282ce3", "95c11548", "e4c66d22", "48c1133f", "c70f86dc",
"07f9c9ee", "41041f0f", "404779a4", "5d886e17", "325f51eb",
"d59bc0d1", "f2bcc18f", "41113564", "257b7834", "602a9c60",
"dff8e8a3", "1f636c1b", "0e12b4c2", "02e1329e", "af664fd1",

"cad18115", "6b2395e0", "333e92e1", "3b240b62", "eebeb922",
"85b2a20e", "e6ba0d99", "de720c8c", "2da2f728", "d0127845",
"95b794fd", "647d0862", "e7ccf5f0", "5449a36f", "877d48fa",
"c39dfd27", "f33e8d1e", "0a476341", "992eff74", "3a6f6eab",
"f4f8fd37", "a812dc60", "a1ebddf8", "991be14c", "db6e6b0d",
"c67b5510", "6d672c37", "2765d43b", "dcd0e804", "f1290dc7",
"cc00ffa3", "b5390f92", "690fed0b", "667b9ffb", "cedb7d9c",
"a091cf0b", "d9155ea3", "bb132f88", "515bad24", "7b9479bf",
"763bd6eb", "37392eb3", "cc115979", "8026e297", "f42e312d",
"6842ada7", "c66a2b3b", "12754ccc", "782ef11c", "6a124237",
"b79251e7", "06a1bbe6", "4bfb6350", "1a6b1018", "11caedfa",
"3d25bdd8", "e2e1c3c9", "44421659", "0a121386", "d90ceec6e",
"d5abea2a", "64af674e", "da86a85f", "bebf988", "64e4c3fe",
"9dbc8057", "f0f7c086", "60787bf8", "6003604d", "d1fd8346",
"f6381fb0", "7745ae04", "d736fccc", "83426b33", "f01eab71",
"b0804187", "3c005e5f", "77a057be", "bde8ae24", "55464299",
"bf582e61", "4e58f48f", "f2ddfd2", "f474ef38", "8789bdc2",
"5366f9c3", "c8b38e74", "b475f255", "46fcd9b9", "7aeb2661",
"8b1ddf84", "846a0e79", "915f95e2", "466e598e", "20b45770",
"8cd55591", "c902de4c", "b90bace1", "bb8205d0", "11a86248",
"7574a99e", "b77f19b6", "e0a9dc09", "662d09a1", "c4324633",

"e85a1f02", "09f0be8c", "4a99a025", "1d6efe10", "1ab93d1d",
"0ba5a4df", "a186f20f", "2868f169", "dcb7da83", "573906fe",
"a1e2ce9b", "4fcd7f52", "50115e01", "a70683fa", "a002b5c4",
"0de6d027", "9af88c27", "773f8641", "c3604c06", "61a806b5",
"f0177a28", "c0f586e0", "006058aa", "30dc7d62", "11e69ed7",
"2338ea63", "53c2dd94", "c2c21634", "bbcbee56", "90bcb6de",
"ebfc7da1", "ce591d76", "6f05e409", "4b7c0188", "39720a3d",
"7c927c24", "86e3725f", "724d9db9", "1ac15bb4", "d39eb8fc",
"ed545578", "08fca5b5", "d83d7cd3", "4dad0fc4", "1e50ef5e",
"b161e6f8", "a28514d9", "6c51133c", "6fd5c7e7", "56e14ec4",
"362abfce", "ddc6c837", "d79a3234", "92638212", "670efa8e",
"406000e0" },

{ "3a39ce37", "d3faf5cf", "abc27737", "5ac52d1b", "5cb0679e",
"4fa33742", "d3822740", "99bc9bbe", "d5118e9d", "bf0f7315",
"d62d1c7e", "c700c47b", "b78c1b6b", "21a19045", "b26eb1be",
"6a366eb4", "5748ab2f", "bc946e79", "c6a376d2", "6549c2c8",
"530ff8ee", "468dde7d", "d5730a1d", "4cd04dc6", "2939bbdb",
"a9ba4650", "ac9526e8", "be5ee304", "a1fad5f0", "6a2d519a",
"63ef8ce2", "9a86ee22", "c089c2b8", "43242ef6", "a51e03aa",
"9cf2d0a4", "83c061ba", "9be96a4d", "8fe51550", "ba645bd6",
"2826a2f9", "a73a3ae1", "4ba99586", "ef5562e9", "c72fefd3",

"f752f7da", "3f046f69", "77fa0a59", "80e4a915", "87b08601",
"9b09e6ad", "3b3ee593", "e990fd5a", "9e34d797", "2cf0b7d9",
"022b8b51", "96d5ac3a", "017da67d", "d1cf3ed6", "7c7d2d28",
"1f9f25cf", "adf2b89b", "5ad6b472", "5a88f54c", "e029ac71",
"e019a5e6", "47b0acfd", "ed93fa9b", "e8d3c48d", "283b57cc",
"f8d56629", "79132e28", "785f0191", "ed756055", "f7960e44",
"e3d35e8c", "15056dd4", "88f46dba", "03a16125", "0564f0bd",
"c3eb9e15", "3c9057a2", "97271aec", "a93a072a", "1b3f6d9b",
"1e6321f5", "f59c66fb", "26dcf319", "7533d928", "b155fdf5",
"03563482", "8aba3cbb", "28517711", "c20ad9f8", "abcc5167",
"ccad925f", "4de81751", "3830dc8e", "379d5862", "9320f991",
"ea7a90c2", "fb3e7bce", "5121ce64", "774fbe32", "a8b6e37e",
"c3293d46", "48de5369", "6413e680", "a2ae0810", "dd6db224",
"69852dfd", "09072166", "b39a460a", "6445c0dd", "586cdecf",
"1c20c8ae", "5bbef7dd", "1b588d40", "ccd2017f", "6bb4e3bb",
"dda26a7e", "3a59ff45", "3e350a44", "bcb4cdd5", "72eacea8",
"fa6484bb", "8d6612ae", "bf3c6f47", "d29be463", "542f5d9e",
"aec2771b", "f64e6370", "740e0d8d", "e75b1357", "f8721671",
"af537d5d", "4040cb08", "4eb4e2cc", "34d2466a", "0115af84",
"e1b00428", "95983a1d", "06b89fb4", "ce6ea048", "6f3f3b82",
"3520ab82", "011a1d4b", "277227f8", "611560b1", "e7933fdc",

"bb3a792b", "344525bd", "a08839e1", "51ce794b", "2f32c9b7",
"a01fbac9", "e01cc87e", "bcc7d1f6", "cf0111c3", "a1e8aac7",
"1a908749", "d44fbd9a", "d0dadecb", "d50ada38", "0339c32a",
"c6913667", "8df9317c", "e0b12b4f", "f79e59b7", "43f5bb3a",
"f2d519ff", "27d9459c", "bf97222c", "15e6fc2a", "0f91fc71",
"9b941525", "fae59361", "ceb69ceb", "c2a86459", "12baa8d1",
"b6c1075e", "e3056a0c", "10d25065", "cb03a442", "e0ec6e0e",
"1698db3b", "4c98a0be", "3278e964", "9f1f9532", "e0d392df",
"d3a0342b", "8971f21e", "1b0a7441", "4ba3348c", "c5be7120",
"c37632d8", "df359f8d", "9b992f2e", "e60b6f47", "0fe3f11d",
"e54cda54", "1edad891", "ce6279cf", "cd3e7e6f", "1618b166",
"fd2c1d05", "848fd2c5", "f6fb2299", "f523f357", "a6327623",
"93a83531", "56cccd02", "acf08162", "5a75ebb5", "6e163697",
"88d273cc", "de966292", "81b949d0", "4c50901b", "71c65614",
"e6c6c7bd", "327a140a", "45e1d006", "c3f27b9a", "c9aa53fd",
"62a80f00", "bb25bfe2", "35bdd2f6", "71126905", "b2040222",
"b6cbcf7c", "cd769c2b", "53113ec0", "1640e3d3", "38abbd60",
"2547adf0", "ba38209c", "f746ce76", "77afa1c5", "20756060",
"85cbfe4e", "8ae88dd8", "7aaaf9b0", "4cf9aa7e", "1948c25c",
"02fb8a8c", "01c36ae4", "d6ebe1f9", "90d4f869", "a65cdea0",
"3f09252d", "c208e69f", "b74e6132", "ce77e25b", "578fdfe3",

```
"3ac372e6" } };
```

```
// Subkeys initialisation with digits of pi.
```

```
String P[] = { "243f6a88", "85a308d3", "13198a2e", "03707344", "a4093822",
```

```
"299f31d0", "082efa98", "ec4e6c89", "452821e6", "38d01377",
```

```
"be5466cf", "34e90c6c", "c0ac29b7", "c97c50dd", "3f84d5b5",
```

```
"b5470917", "9216d5d9", "8979fb1b" };
```

```
// to store  $2^{32}$ (for addition modulo  $2^{32}$ ).
```

```
long modVal = 1;
```

```
// to convert hexadecimal to binary.
```

```
private String hexToBin(String plainText)
```

```
{
```

```
    String binary = "";
```

```
    Long num;
```

```
    String binary4B;
```

```
    int n = plainText.length();
```

```
    for (int i = 0; i < n; i++) {
```

```
        num
```

```

        = Long.parseUnsignedLong(

            plainText.charAt(i) + "", 16);

        binary4B = Long.toBinaryString(num);


        // each value in hexadecimal

        // is 4 bits in binary.

        binary4B = "0000" + binary4B;

        binary4B

        = binary4B.substring(

            binary4B.length() - 4);

        binary += binary4B;

    }

    return binary;

}


// convert from binary to hexadecimal.

private String binToHex(String plainText)

{

    long num = Long.parseUnsignedLong(plainText, 2);

    String hexa = Long.toHexString(num);

```

```
while (hexa.length() < (plainText.length() / 4))

    // maintain output length same length

    // as input by appending leading zeroes.

    hexa = "0" + hexa;

return hexa;

}
```

```
// xor two hexadecimal strings of same length.
```

```
private String xor(String a, String b)
```

```
{

    a = hexToBin(a);

    b = hexToBin(b);

    String ans = "";

    for (int i = 0; i < a.length(); i++)

        ans += (char)((((a.charAt(i) - '0')

                        ^ (b.charAt(i) - '0'))

                    + '0'));

    ans = binToHex(ans);

    return ans;
```

```
}
```

```
// addition modulo  $2^{32}$  of two hexadecimal strings.
```

```
private String addBin(String a, String b)
```

```
{
```

```
    String ans = "";
```

```
    long n1 = Long.parseUnsignedLong(a, 16);
```

```
    long n2 = Long.parseUnsignedLong(b, 16);
```

```
    n1 = (n1 + n2) % modVal;
```

```
    ans = Long.toHexString(n1);
```

```
    ans = "00000000" + ans;
```

```
    return ans.substring(ans.length() - 8);
```

```
}
```

```
// function F explained above.
```

```
private String f(String plainText)
```

```
{
```

```
    String a[] = new String[4];
```

```
    String ans = "";
```

```
    for (int i = 0; i < 8; i += 2) {
```

```

        // the column number for S-box

        // is 8-bit value(8*4 = 32 bit plain text)

        long col

        = Long.parseUnsignedLong(

            hexToBin(

                plainText.substring(i, i + 2)),

                2);

        a[i / 2] = S[i / 2][(int)col];

    }

    ans = addBin(a[0], a[1]);

    ans = xor(ans, a[2]);

    ans = addBin(ans, a[3]);

    return ans;

}

// generate subkeys.

private void keyGenerate(String key)

{

    int j = 0;

    for (int i = 0; i < P.length; i++) {

        P[i] = xor(P[i], key.substring(j, j + 8));
    }
}

```

```

        System.out.println("subkey "

                                + (i + 1) + ": "

                                + P[i]);

        j = (j + 8) % key.length();

    }

}

// round function

private String round(int time, String plainText)

{

    String left, right;

    left = plainText.substring(0, 8);

    right = plainText.substring(8, 16);

    left = xor(left, P[time]);

    String fOut = f(left); // output from F function

    right = xor(fOut, right);

    System.out.println("round " + time + ": "

                                + right + left);

```

```
// swap left and right

return right + left;

}


// decryption

private String decrypt(String plainText)

{

    for (int i = 17; i > 1; i--)

        plainText = round(i, plainText);


    // postprocessing

    String right = plainText.substring(0, 8);

    String left = plainText.substring(8, 16);

    right = xor(right, P[1]);

    left = xor(left, P[0]);

    return left + right;

}


Main()

{
```



```
// storing 2^32 in modVal

//(<<1 is equivalent to multiply by 2)

for (int i = 0; i < 32; i++)

    modVal = modVal << 1;

String cipherText = "d748ec383d3405f7";

String key = "aabb09182736ccdd";


keyGenerate(key);


System.out.println("-----Decryption-----");

String plainText = decrypt(cipherText);

System.out.println("Plain Text: "

    + plainText);

}


public static void main(String args[])

{

    new Main();

}

}
```

Output

subkey 1: 8e846390

subkey 2: a295c40e

subkey 3: b9a28336

subkey 4: 2446bf99

subkey 5: 0eb2313a

subkey 6: 0ea9fd0d

subkey 7: a295f380

subkey 8: cb78a054

subkey 9: ef9328fe

subkey 10: 1fe6dfaa

subkey 11: 14ef6fd7

subkey 12: 13dfc0b1

subkey 13: 6a1720af

subkey 14: ee4a9c00

subkey 15: 953fdcad

subkey 16: 9271c5ca

subkey 17: 38addcc1

subkey 18: ae4f37c6

-----Decryption-----

round 17: 3ab5e5667907dbfe

round 16: fdd297bb021839a7

round 15: 82529d676fa35271

round 14: ec939d1a176d41ca

round 13: e14063bd02d9011a

round 12: 66cd65508b574312

round 11: 37e82a387512a5e1

round 10: 8fe62e7e230745ef

round 9: 1f04e6309000f1d4

round 8: 3624ea12f097cece

round 7: c546e12ffd5c4a46

round 6: ed76301e67d312af

round 5: bbd76433e3dfcd13

round 4: f160c1f4b5655509

round 3: 2512b60dd5267e6d

round 2: 6f86e1389cb0353b

Plain Text: 123456abcd132536

Advantages and Disadvantages of Blowfish Algorithm:

- Blowfish is a fast block cipher except when changing keys. Each new key requires a pre-processing equivalent to 4KB of text.
- It is faster and much better than DES Encryption.
- Blowfish uses a 64-bit block size which makes it vulnerable to [birthday attacks](#).
- A reduced round variant of blowfish is known to be susceptible to known plain text attacks([2nd order differential attacks](#) – 4 rounds).

Applications of Blowfish Algorithm:

- Bulk Encryption.
- Packet Encryption(ATM Packets)
- [Password Hashing](#)

2.9] Advanced Encryption Standard (AES)

The Advanced Encryption Standard, or AES, is an encryption standard established in 2001 by the National Institute of Standards and Technology (NIST) of USA. It is implemented worldwide both in hardware and software to encrypt sensitive data. AES finds wide usage while transmitting data over computer networks, particularly in wireless networks.

Features of AES

- AES is a subset of Rijndael block cipher.
- It is a successor of Data Encryption Standard (DES) and is stronger and faster than DES.
- It is a symmetric key symmetric block cipher.
- It operates on 128-bit (16 bytes) data.
- The cipher key may be of 128, 192 or 256 bits.
- All computations are performed on bytes rather than bits.
- AES gives full specification and design details.
- It can be implemented using languages C and Java for software protection.

Operation of AES

The encryption process of Advanced Encryption Standard is based upon substitution and permutation operations in iterative manner. The 16 bytes of data are arranged in a matrix of four columns and four rows. On this matrix, AES performs rounds of substitution-permutation operations. Each of these rounds uses a different cipher key, which is calculated from the original AES key. The number of rounds of operations depends upon the size of the key in the following manner –

- For 128-bit cipher key, 10 rounds
- For 192-bit cipher key, 12 rounds
- For 256-bit cipher key, 14 rounds

The following diagram shows the schematic structure –

The encryption sub-processes in the above algorithm are –

- **Key Expansion** – The round keys are calculated from the cipher key using Rijndael's block cipher schedule.
- **Pre-Transformation** – This comprises of only 1 process namely Add_Round_Key. Here, XOR operation is performed on each data byte with a byte of the round key.
- **Round 1 to Round (N-1)** – Four sub-processes are performed here.
 - Sub_Bytes – Non-linear substitution is performed on each byte whereby the byte is replaced with another byte as per a lookup table.
 - Shift_Rows – Transposition is performed wherein a certain number of cyclical shifting of the last three rows is done.
 - Mix_Columns – Mixing of rows and columns in a pre-defined manner is performed.
 - Add_Round_Key – XOR operation is performed on each byte with a byte of the round key.
- **Round N** – The final round comprises of three sub-processes, namely –
 - Sub_Bytes
 - Shift_Rows
 - Add_Round_Key

The following diagram shows the sub processes in Round 1 of the AES algorithm

- ❖ [Advanced Encryption Standard \(AES\)](#) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and

Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

Points to remember

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

Working of the cipher :

AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows :

- 128 bit key – 10 rounds
- 192 bit key – 12 rounds
- 256 bit key – 14 rounds

Creation of Round keys :

A Key Schedule algorithm is used to calculate all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption.

Encryption :

AES considers each block as a 16 byte (4 byte x 4 byte = 128) grid in a column major arrangement.

```
[ b0 | b4 | b8 | b12 |  
| b1 | b5 | b9 | b13 |  
| b2 | b6 | b10 | b14 |  
| b3 | b7 | b11 | b15 ]
```

Each round comprises of 4 steps :

- SubBytes
- ShiftRows
- MixColumns
- Add Round Key

The last round doesn't have the MixColumns round.

The SubBytes does the substitution and ShiftRows and MixColumns performs the permutation in the algorithm.

SubBytes :

This step implements the substitution.

In this step each byte is substituted by another byte. Its performed using a lookup table also called the S-box. This substitution is done in a way that a byte is never

substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step is a 16 byte (4 x 4) matrix like before.

The next two steps implement the permutation.

ShiftRows :

This step is just as it sounds. Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.

(A left circular shift is performed.)

[b0 b1 b2 b3]		[b0 b1 b2 b3]
b4 b5 b6 b7	->	b5 b6 b7 b4
b8 b9 b10 b11		b10 b11 b8 b9
[b12 b13 b14 b15]		[b15 b12 b13 b14]

MixColumns :

This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

This step is skipped in the last round.

[c0]		[2 3 1 1]		[b0]
c1	=	1 2 3 1		b1
c2		1 1 2 3		b2
[c3]		[3 1 1 2]		[b3]

Add Round Keys :

Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes is not considered as a grid but just as 128 bits of data. Diag.

After all these rounds 128 bits of encrypted data is given back as output. This process is repeated until all the data to be encrypted undergoes this process.

Decryption :

The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes. Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

The stages of each round in decryption is as follows :

- Add round key
- Inverse MixColumns
- ShiftRows
- Inverse SubByte

The decryption process is the encryption process done in reverse so i will explain the steps with notable differences.

Inverse MixColumns :

This step is similar to the MixColumns step in encryption, but differs in the matrix used to carry out the operation.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Inverse SubBytes :

Inverse S-box is used as a lookup table and using which the bytes are substituted during decryption.

Applications:

AES is widely used in many applications which require secure data storage and transmission. Some common use cases include:

- **Wireless security:** AES is used in securing wireless networks, such as Wi-Fi networks, to ensure data confidentiality and prevent unauthorized access.
- **Database Encryption:** AES can be applied to encrypt sensitive data stored in databases. This helps protect personal information, financial records, and other confidential data from unauthorized access in case of a data breach.
- **Secure communications:** AES is widely used in protocols like such as internet communications, email, instant messaging, and voice/video calls. It ensures that the data remains confidential.
- **Data storage:** AES is used to encrypt sensitive data stored on hard drives, USB drives, and other storage media, protecting it from unauthorized access in case of loss or theft.
- **Virtual Private Networks (VPNs):** AES is commonly used in VPN protocols to secure the communication between a user's device and a remote server. It ensures that data sent and received through the VPN remains private and cannot be deciphered by eavesdroppers.
- **Secure Storage of Passwords:** AES encryption is commonly employed to store passwords securely. Instead of storing plaintext passwords, the encrypted version is stored. This adds an extra layer of security and protects user credentials in case of unauthorized access to the storage.
- **File and Disk Encryption:** AES is used to encrypt files and folders on computers, external storage devices, and cloud storage. It protects sensitive data stored on devices or during data transfer to prevent unauthorized access.

Summary :

AES instruction set is now integrated into the CPU (offers throughput of several GB/s) to improve the speed and security of applications that use AES for encryption and decryption. Even though it's been 20 years since its introduction we have failed to break the AES algorithm as it is infeasible even with the current technology. Till date the only vulnerability remains in the implementation of the algorithm.

Unit 3 : Assymmetric Key Algorithm, Digital Signatures And RSA

3.1] Introduction

Assymmetric Key Algorithm :

Unlike symmetric encryption methods, asymmetric encryption methods use two separate keys for the encryption and decryption process. Asymmetric encryption is also referred to as public key encryption. A public key is used during the encryption process to encrypt the plaintext message and a private key is used during the decryption process to decrypt ciphertext.

Advantages	Disadvantages
Easier to deal with key distribution and ensure integrity of message	Slower due to multiple keys

If Alice wants to send a message that only Bob can read, Alice will use Bob's public key to encrypt the message. The only way the message can be decrypted is when Bob uses his private key.

Figure 2 – Asymmetric encryption

The most popular asymmetric algorithm is RSA which can provide key exchange, encryption, and digital signatures. The strength of this algorithm lies in the difficulty of calculating the prime factors of very large numbers. Other asymmetric algorithms include ElGamal, ECC, and Knapsack. Read the article, Asymmetric Encryption Algorithms, Diffie-Hellman, RSA, ECC, ElGamal, DSA, for a comparison of these algorithms.

❖ Digital Signature :

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is

very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

3.2] History And Overview Of Assymmetric Key Cryptography :

Asymmetric cryptography is a second form of cryptography. It is called a Public-key cryptography. There are two different keys including one key is used for encryption and only the other corresponding key should be used for decryption. There is no other key can decrypt the message and not even the initial key used for encryption. The style of the design is that every communicating party needs only a key pair for communicating with any number of other communicating parties.

Asymmetric cryptography is scalable for use in high and ever expanding environments where data are generally exchanged between different communication partners. Asymmetric cryptography is used to exchange the secret key to prepare for using symmetric cryptography to encrypt information.

In the case of a key exchange, one party produce the secret key and encrypts it with the public key of the recipient. The recipient can decrypt it with their private key. The remaining communication would be completed with the secret key being the encryption key. Asymmetric encryption is used in key exchange, email security, Web security, and some encryption systems that needed key exchange over the public network.

The main problem for asymmetric-key cryptography is proving that a public key is authentic, and has not been tampered with or restored by a malicious third party. The beneficial method to this problem is to use a Public-key Infrastructure (PKI), in which one or more third parties, called a certificate authorities, certify ownership of key pairs. Another method, used by PGP, is the “web of trust” method to provide authenticity of key pairs.

Asymmetric key cryptography is a cryptographic technique that uses a several verification key in the CAD than the proving key used by the smart card. This method is usually implemented using trapdoor one-way functions, in which smart card creates an electronic signature with its secret key and the CAD needs a public key to authenticate the signature.

The Rivest-Shamir-Adelman (RSA) public key cryptosystem is the most frequently implemented asymmetric key authentication method. It places the security of the algorithm in the complexity of factoring large prime numbers.

The Digital Signature Standard (DSS) developed by the National Institute of Standards and Testing (NIST) can be used to smart card authentication. This method was produced for making document signatures in the U.S. Government and can be a preferred choice for government

application authentication. Its strength lies in the complexity in computing discrete logarithms. It has an advantage that several intermediate values for an authentication can be "pre-evaluated" so that the authentication time can be decreased.

The crux of public key systems, particularly those that are used for authentication purposes, is key distribution. The certificate enables the smart cards to support the verification key to the CAD. The certificate is formed when the card is boot up using a system wide secret key.

3.3] The RSA Algorithm :

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption. It was invented by Rivest, Shamir and Adleman in year 1978 and hence name **RSA** algorithm.

Algorithm

The RSA algorithm holds the following features –

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.
- There are two sets of keys in this algorithm: private key and public key.

You will have to go through the following steps to work on RSA algorithm –

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q , and then calculating their product N , as shown –

$$N = p * q$$

Here, let N be the specified large number.

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than $(p-1)$ and $(q-1)$. The primary condition will be that there should be no common factor of $(p-1)$ and $(q-1)$ except 1

Step 3: Public key

The specified pair of numbers n and e forms the RSA public key and it is made public.

Step 4: Private Key

Private Key d is calculated from the numbers p , q and e . The mathematical relationship between the numbers is as follows –

$$ed = 1 \bmod (p-1)(q-1)$$

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

Encryption Formula:

Consider a sender who sends the plain text message to someone whose public key is (n,e) . To encrypt the plain text message in the given scenario, use the following syntax –

$$C = P^e \bmod n$$

Decryption Formula:

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver C has the private key d , the result modulus will be calculated as –

$$\text{Plaintext} = C^d \bmod n$$

3.4] Symmetric And Asymmetric Key Cryptography

❖ Symmetric Key Cryptography

Symmetric Key Cryptography, or Symmetric Encryption, uses a secret key for both encryption and decryption. This approach is the inverse of Asymmetric Encryption, which uses one key to encrypt and another to decrypt. Data is translated to a format that cannot be interpreted or inspected by someone who does not have the secret key used to encrypt it during this phase.

The strength of the random number generator used to generate the secret key determines the effectiveness of this method. Symmetric Key Cryptography, commonly used on the Internet today, comprises two kinds of algorithms: Block and Stream. The Advanced Encryption Standard (AES) and the Data Encryption Standard (DES) are two common encryption algorithms. This type of encryption is typically much faster than Asymmetric Encryption, but it allows the secret key to be held by both the sender and the data receiver.

Symmetric cryptography is based on a single shared key that all parties are aware of and can use to encrypt and decrypt data.

Secret-key, single-key, shared-key, one-key, and private-key encryption are other words for symmetric-key cryptography. The usage of the last and first words will lead to misunderstanding compared to the related language used in public-key cryptography.

Symmetric key encryption employs one of the following encryption techniques:

Stream ciphers: Encrypt a message's digits or letters one at a time.

Block ciphers: Encrypt a group of bits as a single entity, inserting the plaintext to make it a block size multiple. 64-bit blocks are widely used. The NIST-approved Advanced Encryption Standard (AES) algorithm and the GCM block cipher mode of operation all use 128-bit blocks.

What is the Purpose of Symmetric Encryption?

Although symmetric encryption is an older type of encryption, it is simpler and more effective than asymmetric encryption, which strains networks due to data size performance problems and heavy CPU usage.

Since symmetric encryption performs smoother and quicker than asymmetric encryption, it is commonly used for bulk encryption / encrypting massive volumes of data, such as database encryption. In a database, the secret key can be used only by the database to encrypt or decrypt data.

Here are a few instances of where symmetric cryptography is used:

- Payment applications, such as bank purchases, where personally identifiable information (PII) must be secured to avoid identity theft or fraudulent charges
- Validations are performed to ensure that the sender of a message is who he appears to be.
- Hashing or random number creation

Advantages

- A symmetric cryptosystem is more effective.
- Encrypted data can be transmitted over a network in Symmetric Cryptosystems even though it is certain that the data would be intercepted. Since no key is sent with the files, the chances of data decryption are zero.
- To confirm the receiver's existence, a symmetric cryptosystem employs password authentication.
- A message can only be decrypted by a device that has a hidden key.
- Prevents widespread message protection breaches. For communicating with each party, a separate secret key is used. Only communications from a specific pair of sender and recipient are impacted when a key is corrupted. Communication with others is always safe.
- This type of encryption is simple to implement. All users need to do is specify and exchange the secret key until they can begin encrypting and decrypting messages.
- Encrypt and decrypt your files. There is no need to build separate keys if you use encryption for messages or data that you wish to access just once. For this, single-key encryption is ideal.

- Symmetric key encryption is much faster than asymmetric key encryption.
- Uses fewer computer resources. As opposed to public-key encryption, single-key encryption needs fewer computing resources.

Disadvantages

- Key transportation is a concern in symmetric cryptosystems. The secret key must be sent to the receiving device before the final message is sent. Electronic communication is unreliable, and no one can guarantee the communication networks will not be tapped. As a result, the only safe method of sharing keys will be to do it in person.
- It is not possible to have digital signatures that cannot be revoked.
- The message's origin and validity cannot be assured. Messages cannot be proven to have originated from a specific person since both sender and recipient use the same key. If there is a disagreement, this may be a challenge.
- For communication between each different party, a new shared key must be created. This poses a challenge with handling and securing both of these keys.

❖ Asymmetric Key Cryptography

Asymmetric cryptography, better known as public-key cryptography, encrypts and decrypts a message using a pair of similar keys. In asymmetric key cryptography, the private key is kept by one public key and one private key — to prevent unauthorized entry or usage. Anybody can use a public key to encrypt a document so that only the expected receiver can decrypt it with their private key. A private key or secret key is only known to the key's generator.

When anyone tries to submit an encrypted message, they will use a shared directory to retrieve the recipient's public key and use it to encrypt the message until submitting it. The message will then be decrypted by the receiver using their associated private key.

However, when the sender encrypts the message using their private key, the message may only be decrypted using the sender's public key, thus authenticating the sender. These encryption and decryption procedures are automatic; users don't need to lock and unlock the message manually.

Numerous protocols, including the transport layer security (TLS) and safe sockets layer (SSL) protocols that allow HTTPS, depend on asymmetric cryptography. Encryption is often used in browsers that need to create a stable link over an unstable network, such as the Internet, or to verify a digital signature.

The key advantage of asymmetric cryptography is increased data security. Since users are never expected to disclose or exchange their private keys, the risks of cyber activity on a user's private key during transmission are reduced.

What is the Purpose of Asymmetric Encryption?

Asymmetric encryption uses: Asymmetric cryptography is often used to check the authenticity of data using digital signatures. A digital signature is a cryptographic technique for verifying the validity and credibility of a message, software, or digital record. It's the equivalent of an in-person signature or a sealed seal in digital form.

Digital signatures, which are based on asymmetric cryptography, may include proof of the origin, identification, and status of an electronic record, transaction, or post, as well as acknowledge the signer's informed consent.

You can also use asymmetric encryption in applications where many users can encrypt and decode messages, such as:

- Encrypted email entails using a public key to encrypt a file and a private key to decode it.
- Asymmetric encryption is often used in the SSL/TSL cryptographic protocols, which provide encrypted connections between websites and browsers.
- Asymmetric encryption is used for Bitcoin and other crypto currencies when consumers provide public keys that everybody will use and private keys held hidden. Bitcoin employs a cryptographic algorithm to guarantee that only the funds belonging to rightful owners can be spent.
- Each unspent transaction output (UTXO) in the Bitcoin ledger is usually associated with a public key. So, suppose user A needs to give user B some money and has a UTXO associated with his public key. In that case, he uses his private key to sign a contract that spends the UTXO and makes a new UTXO associated with user B's public key.

Advantages

- There is no need to exchange keys in asymmetric or public key cryptography, eliminating the key distribution issue.
- The main benefit of public-key cryptography is improved security: private keys are never exchanged or exposed to others.
- May provide digital signatures that can be revoked.
- Message verification is provided by public-key cryptography, which requires the use of digital signatures, which allows the receiver of a message to check that the message is actually from a specific sender.
- The usage of digital signatures in public-key cryptography helps the recipient to determine whether or not the message was altered during transit. A digitally signed message cannot be altered without rendering the signature null.

- Signing a message digitally is analogous to physically signing a document. Since it is an acknowledgement of the message, the sender cannot reject it.

Disadvantages

- One drawback of using public-key cryptography for encryption is the lack of speed. Popular secret-key encryption systems are substantially quicker than any commonly accessible public-key encryption technique.
- Authentication of public keys is recommended/required. No one can be certain that a public key corresponds to the individual it identifies, so everybody must verify that their public keys are theirs.
- It consumes more computer resources. It necessitates much more computing resources than single-key encryption.
- A widespread security breach is likely if an intruder obtains a person's private key and reads his or her entire message.
- The loss of a private key can be irreversible. When a private key is lost, all incoming messages cannot be decrypted.

Symmetric Key Cryptography	Asymmetric Key Cryptography
There is just one key (symmetric key) used, and it is the same key used to encrypt and decrypt the message.	For encryption and decryption, two different cryptographic keys (asymmetric keys), known as the public and private keys, are used.
Since it is a straightforward procedure, the encryption method can be completed easily.	It is a much more complex and time-consuming mechanism than symmetric key encryption.
Key lengths are usually 128 or 256 bits, depending on the security criteria.	The key length is even greater; for asymmetric encryption example, recommended RSA key size is 2048 bits or higher.
It is also called secret-key cryptography or private key cryptography.	Asymmetric key is also called a conventional cryptography system or public-key cryptography.
Represented mathematically as: $P=D(K,E(P))$	Represented mathematically as: $P=D(K_d,E(K_e,P))$
It uses fewer resources than an asymmetric key cipher uses.	It consumes more resources than symmetric key cryptography.
It is used where massive amounts of data may be transmitted.	It is mainly used in smaller transactions to authenticate and create a stable contact channel before data transfer.
The secret key is shared. As a result, the likelihood of compromise is increased.	Since the private key is not shared, the overall process is more reliable than symmetric encryption.
Symmetric encryption is an old technique.	Asymmetric encryption is relatively new.
RC4, AES, DES, 3DES, and other algorithms are	RSA, Diffie-Hellman, ECC, and other asymmetric encryption algorithms

examples.

are examples.

3.5] Digital Signature :

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

Model of Digital Signature :

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration – Diag

The following points explain the entire process in detail –

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.
- Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence **signing a hash is more efficient than signing the entire data.**

Importance of Digital Signature :

Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security.

Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature –

- **Message authentication** – When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.
- **Data Integrity** – In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.
- **Non-repudiation** – Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely – Privacy, Authentication, Integrity, and Non-repudiation.

3.5.1] Encryption with Digital Signature:

In many digital communications, it is desirable to exchange an encrypted messages than plaintext to achieve confidentiality. In public key encryption scheme, a public (encryption) key of sender is available in open domain, and hence anyone can spoof his identity and send any encrypted message to the receiver.

This makes it essential for users employing PKC for encryption to seek digital signatures along with encrypted data to be assured of message authentication and non-repudiation.

This can archived by combining digital signatures with encryption scheme. Let us briefly discuss how to achieve this requirement. There are **two possibilities, sign-then-encrypt** and **encrypt-then-sign**.

However, the crypto system based on sign-then-encrypt can be exploited by receiver to spoof identity of sender and sent that data to third party. Hence, this method is not preferred. The process of encrypt-then-sign is more reliable and widely adopted. This is depicted in the following illustration – Diag

The receiver after receiving the encrypted data and signature on it, first verifies the signature using sender's public key. After ensuring the validity of the signature, he then retrieves the data through decryption using his private key.

3.6] Knapsack And Other Algorithm

Knapsack Encryption Algorithm is the first general public key cryptography algorithm. It is developed by **Ralph Merkle** and **Mertin Hellman** in 1978. As it is a Public key cryptography, it needs two different keys. One is Public key which is used for Encryption process and the other one is Private key which is used for Decryption process. In this algorithm we will use two different knapsack problems in which one is easy and other one is hard. The easy knapsack is used as the private key and the hard knapsack is used as the public key. The easy knapsack is used to derived the hard knapsack.

For the easy knapsack, we will choose a **Super Increasing knapsack problem**. Super increasing knapsack is a sequence in which every next term is greater than the sum of all preceding terms.

Example –

{1, 2, 4, 10, 20, 40} is a super increasing as
 $1 < 2$, $1+2 < 4$, $1+2+4 < 10$, $1+2+4+10 < 20$ and $1+2+4+10+20 < 40$.

Derive the Public key

- **Step-1:**

Choose a super increasing knapsack {1, 2, 4, 10, 20, 40} as the private key.

- **Step-2:**

Choose two numbers n and m. Multiply all the values of private key by the number n and then find modulo m. The value of m must be greater than the sum of all values in private key, for example 110. And the number n should have no common factor with m, for example 31.

- **Step-3:**

Calculate the values of Public key using m and n.

$$1 \times 31 \bmod(110) = 31$$

$$2 \times 31 \bmod(110) = 62$$

$$4 \times 31 \bmod(110) = 14$$

$$10 \times 31 \bmod(110) = 90$$

$$20 \times 31 \bmod(110) = 70$$

$$40 \times 31 \bmod(110) = 30$$

- Thus, our public key is {31, 62, 14, 90, 70, 30}

And Private key is {1, 2, 4, 10, 20, 40}.

Now take an example for understanding the process of encryption and decryption.

Example –

Lets our plain text is 100100111100101110.

1. Encryption :

As our knapsacks contain six values, so we will split our plain text in a groups of six:

100100 111100 101110

Multiply each values of public key with the corresponding values of each group and take their sum.

100100 {31, 62, 14, 90, 70, 30}

$$1 \times 31 + 0 \times 62 + 0 \times 14 + 1 \times 90 + 0 \times 70 + 0 \times 30 = 121$$

111100 {31, 62, 14, 90, 70, 30}

$$1 \times 31 + 1 \times 62 + 1 \times 14 + 1 \times 90 + 0 \times 70 + 0 \times 30 = 197$$

101110 {31, 62, 14, 90, 70, 30}

$$1 \times 31 + 0 \times 62 + 1 \times 14 + 1 \times 90 + 1 \times 70 + 0 \times 30 = 205$$

So, our cipher text is 121 197 205.

2. Decryption :

The receiver receive the cipher text which has to be decrypt. The receiver also knows the values of m and n.

So, first we need to find the $n^{-1} \pmod{m}$, which is multiplicative inverse of n mod m i.e.,

$$n \times \text{mod}(m) = 131 \times \text{mod}(110) = 1 = 71$$

Now, we have to multiply 71 with each block of cipher text take modulo m.

$$121 \times 71 \pmod{110} = 11$$

Then, we will have to make the sum of 11 from the values of private key {1, 2, 4, 10, 20, 40} i.e.,

$$1 + 10 = 11 \text{ so make that corresponding bits 1 and others 0 which is } 100100.$$

Similarly,

$$197 \times 71 \pmod{110} = 17$$

$$1 + 2 + 4 + 10 = 17 = 111100$$

$$\text{And, } 205 \times 71 \pmod{110} = 35$$

$$1 + 4 + 10 + 20 = 35 = 101110$$

After combining them we get the decoded text.

100100111100101110 which is our plain text.

Example:

Here's a Python example of the Knapsack Encryption Algorithm with proper explanation and output:

- Python

```
import random
```

```
# Function to generate a super-increasing sequence for the public key
```

```
def generate_super_increasing_sequence(n):
```

```
    sequence = [random.randint(1, 100)]
```

```
    while len(sequence) < n:
```

```
        next_element = sum(sequence) + random.randint(1, 10)
```

```
        sequence.append(next_element)
```

```
    return sequence
```

```
# Function to generate the private key from the public key
```

```
def generate_private_key(public_key, q, r):
```

```
    private_key = [(r * element) % q for element in public_key]
```

```
    return private_key
```

```
# Function to encrypt the plaintext using the public key
```

```
def knapsack_encrypt(plaintext, public_key):
```

```
    encrypted_message = sum(public_key[i] for i in range(len(plaintext)) if plaintext[i] == '1')
```

```
    return encrypted_message
```

```
# Function to decrypt the ciphertext using the private key
```

```
def knapsack_decrypt(ciphertext, private_key, q):
```

```
r_inverse = pow(r, -1, q) # Modular multiplicative inverse of r
```

```
decrypted_message = ""
```

```
for element in reversed(private_key):
```

```
    if (ciphertext * r_inverse) % q >= element:
```

```
        decrypted_message = '1' + decrypted_message
```

```
        ciphertext -= element
```

```
    else:
```

```
        decrypted_message = '0' + decrypted_message
```

```
return decrypted_message
```

```
# Example usage
```

```
if __name__ == "__main__":
```

```
    n = 8 # Number of elements in the super-increasing sequence
```

```
    q = 103 # Modulus (should be greater than the sum of the super-increasing sequence)
```

```
    r = 3 # Multiplier for generating private key
```

```
# Generate the public key and private key
```

```
public_key = generate_super_increasing_sequence(n)
```

```
private_key = generate_private_key(public_key, q, r)
```

```
plaintext = "11001010"

ciphertext = knapsack_encrypt(plaintext, public_key)

decrypted_message = knapsack_decrypt(ciphertext, private_key, q)

print("Original Message:", plaintext)

print("Encrypted Ciphertext:", ciphertext)

print("Decrypted Message:", decrypted_message)
```

Explanation:

1. The 'generate_super_increasing_sequence()' function generates a super-increasing sequence, which serves as the public key. The function starts with a random element and adds random values to it until the sequence reaches the desired length n .
2. The 'generate_private_key()' function generates the private key from the public key using the formula: $\text{private_key}[i] = (r * \text{public_key}[i]) \% q$.
3. The 'knapsack_encrypt()' function takes the plaintext and the public key as input. It converts the plaintext to binary and encrypts it by summing the elements of the public key corresponding to '1' bits in the binary representation of the plaintext.
4. The 'knapsack_decrypt()' function takes the ciphertext, private key, and modulus q as input. It uses the private key to reconstruct the original binary representation of the plaintext by finding '1' bits in the ciphertext and subtracting the corresponding elements from the private key.

Output:

Original Message: 11001010

Encrypted Ciphertext: 426

Decrypted Message: 11001010

In this example, the plaintext "11001010" is encrypted using the Knapsack Encryption Algorithm, resulting in the ciphertext "426". The ciphertext is then decrypted back to the original plaintext, which demonstrates the correctness of the algorithm.

Unit 4 – Digital Certificate And Public Key Infrastructure (PKI)

4.1] Introduction

Digital certificates and PKI

Digital certificates facilitate secure electronic communication and data exchange between people, systems, and devices online. They are issued by Certificate Authorities (CAs) and perform two primary functions:

- Verifying the identity of the sender/receiver of an electronic message
- Providing the means to encrypt/decrypt messages between sender and receiver (i.e., binding and entity to their public key)

There are three basic types of digital signature certificates:

- **Individual digital signature certificates (signing certificates):** These certificates are used to identify a person and include personal information. They can be used to sign electronic documents (i.e., to provide electronic signatures) and emails, and to implement access control mechanisms for sensitive or valuable information.
- **Server certificates:** These certificates identify a server (computer) and contain the host name or IP address. They are used for one- or two-layer SSL to ensure secure communication of data over a network.
- **Encryption certificates:** These certificates are used to encrypt a message using the public key of the recipient to ensure data confidentiality during transmission. Different signatures for encryption and digital signatures are available from different CAs. (adapted from [Government of India 2010](#))
- **A system—including policies, institutions, and technologies—that manages the distribution, authentication, and revocation of digital certificates is often referred to as public-key infrastructure (PKI).** Because digital certificates are standard in data exchange and security protocols for digital ID systems (including the TLS encryption measures described above, as well as smartcard- and mobile-based authentication), a country's PKI landscape is a common building block for many ID systems.

For example, when a smartcard or SIM card that uses PKI for authentication and digital signatures is personalized, it is issued with a private key and digital certificate signed by a CA that attests to the authenticity of the credential and provides the public-key necessary for other devices (e.g., card readers, servers, etc.) to verify the authenticity and integrity of the card.

4.2] Digital Certificates

Digital Certificates are a standard of security for establishing an encrypted link between a server and a client. Generally, between a mail server or a webserver, which protects data in transitions by encrypting them. A Digital Certificate is also a Digital ID or a passport which is issued by a Third Party Authority which verifies the identity of the server's owner and not claiming a false identity. (Diag)

Components of a Digital Certificate :

All these following components can be found in the certificate details –

- **Serial Number** – Used to uniquely identify the certificate.
- **Subject** – The person, or entity identified.
- **Signature Algorithm** – The algorithm used to create the signature.
- **Signature** – The actual signature to verify that it came from the issuer.
- **Issuer** – The entity that verified the information and issued the certificate.
- **Valid-From** – The date the certificate is first valid from.
- **Valid-To** – The expiration date.
- **Key-Usage** – Purpose of the public key (For example: encipherment, signature, certificate signing...).
- **Public Key** – The public key.
- **Thumbprint Algorithm** – The algorithm used to hash the public key certificate.
- **Thumbprint** – The hash itself, used as an abbreviated form of the public key certificate.

Levels of Validations

In this section we will discuss the various levels of validations of an SSL (Secure Sockets Layer) Certificate. Some of the most important ones are as follows –

- **Domain Validation SSL Certificate** – It validates the domain that is registered by a system administrators and they have administrator rights to approve the certificate request, this validation generally is done by email request or by a DNS record.
- **Organization Validated SSL Certificates** – It validates the domain ownership and also the business information like the Official Name, City, Country, etc. This validation is done by email or DNS record entering and the certificate authority would also need some genuine documents to verify the Identity.
- **Extended Validation SSL Certificates** – It validates domain ownership and organization information, plus the legal existence of the organization. It also validates that the organization is aware of the SSL certificate request and approves it. The validation requires documentation to certify the company identity plus a set of additional steps and checks. The Extended Validation SSL Certificates are generally identified with a green address bar in the browser containing the company name.

4.3] Private Key Management

A private key, also known as a *secret key*, is a variable in cryptography that is used with an algorithm to encrypt and decrypt data. Secret keys should only be shared with the key's generator or parties authorized to decrypt the data. Private keys play an important role in symmetric cryptography, asymmetric cryptography and [crypto currencies](#).

A private key is typically a long, randomly or pseudo-randomly generated sequence of bits that cannot be easily guessed. The complexity and length of the private key determine how easily an attacker can execute a [brute-force attack](#), where they try out different keys until the right one is found.

How does a private key work?

Private key encryption is also referred to as [symmetric encryption](#), where the same private key is used for both encryption and decryption. In this case, a private key works as follows:

- **Generating a new private key.** Prior to encryption, generate a new key that is as random as possible; encryption software is typically used to generate private keys.
- **Securely storing the private key.** Once generated, the private key must be stored securely. Depending on the application, keys may be stored offline or on the computer used to generate, encrypt and decrypt data. Private keys may be protected with a password, encrypted or hashed for security -- or all three.
- **Key exchange.** The private key is used to decrypt, as well as to encrypt, so using it for symmetric encryption requires a [key exchange](#) to share that key securely with trusted parties authorized to exchange secured data. Cryptographic software is usually used to automate this process.
- **Key management.** Private key management is required to prevent any individual key from being used for too long. It helps to securely retire keys after their useful lifetime is reached.

A private key is also used in [asymmetric cryptography](#), which is also known as [public key cryptography](#). In this case, the private key refers to the secret key of a public key pair. In public key cryptography, the private key is used for encryption and digital signatures. It works as follows for asymmetric cryptography:

- **Generating a public-private key pair.** Randomness is even more important for this process. Encryption application software is usually used to generate key pairs. It should require a source of randomness, such as mouse movement.
- **Securely storing the private key.** Once generated, the private key must be stored securely. Like the symmetric cryptography process, keys may be stored offline or on the computer used to generate, encrypt and decrypt data. Here, too, private keys should be protected with a password, encrypted or hashed for security.
- **Key exchange.** The private key of a public key pair should almost never be shared with others. Public key cryptography, including digital signatures, is typically used to securely share session keys used for symmetric encryption. However, other protocols for [public key infrastructure](#) are used to authoritatively share public keys between cooperating parties.
- **Using the private key.** The owner of a public key pair uses their private key for decrypting data that has been encrypted with the public key of the pair. Only the holder of the private key should be able to decrypt data encrypted with the public key. For digital signatures, the owner of the key pair uses their private key to encrypt the signature. In this way, anyone with access to the public key can decrypt the signature and verify that it was signed by the private key owner.
- **Key management.** Public key pairs are often generated with expiration dates, and key management is vital to maintaining access to data protected with a key pair. For example, an expired public key certificate, which depends on a public key pair, may cause browsers to flag access to a website as insecure. Secret keys should be stored with the highest security, and public key pairs should be managed to avoid compromise or issues related to key pair expiration.

Advantages of private encryption keys:

Private key encryption provides several useful features. They include the following four benefits:

1. **More secure.** Private keys that are longer and have greater [entropy](#), or randomness, are more secure from brute-force or dictionary attacks.
2. **Faster.** Symmetric key encryption is faster computationally than asymmetric encryption with its public-private key pairs.
3. **Best for encryption.** Most cryptographic processes use private key encryption to encrypt data transmissions. They typically use a public key algorithm to securely share secret keys.
4. **Work for stream and block ciphers.** Secret key [ciphers](#) -- the algorithm for encrypting and decrypting data -- generally fall into one of two categories: stream ciphers or [block ciphers](#). A block cipher applies a private key and algorithm to a block of data simultaneously, whereas a stream cipher applies the key and algorithm one bit at a time.

Private keys can be shared securely using the Diffie-Hellman public key exchange protocol.(Diag)

Challenges of private encryption key management :

The security of encryption keys depends on choosing a strong encryption algorithm and maintaining high levels of operational security. [Encryption key management](#) is necessary for any organization using encryption to protect its data. That goes for symmetric, as well as asymmetric, encryption.

While private key encryption can ensure a high level of security, the following key management challenges must be considered:

- **Overall management.** Encryption key management is necessary to protect cryptographic keys from loss, corruption or unauthorized access.
- **Continual updating.** Private keys used to encrypt sensitive data should be changed regularly to minimize exposure should they be leaked or stolen.

- **Recoverability and loss.** If an encryption key becomes inaccessible, data encrypted with that key will be unrecoverable and lost.

Private keys vs. public keys

Asymmetric cryptography, also known as *public key cryptography*, uses pairs of public and private keys. These two different but mathematically linked keys are used to transform plaintext into encrypted [ciphertext](#) or encrypted text back to plaintext.

When the public key is used to encrypt ciphertext, that text can only be decrypted using the private key. This approach enables anyone with access to the public key to encrypt a message, and only the private key holder will be able to decrypt it.

4.4] The PKIX Model

As we know, the X.509 standard defines the digital-certificate structure, format and fields.

It also specifies the procedure for distributing the public keys.

In order to extend such standards and make them universal, the Internet Engineering Task Force (IETF) formed the Public Key Infrastructure X.509(PKIX) working group.

This extends the basic philosophy of the X.509 standard, and specifies how the digital certificates can be deployed in the world of the Internet.

Services of PKIX :

PKIX identifies the primary goals of a PKI infrastructure in the form of the following broad level services:

Registration -It is the process where an end-entity (subject) makes itself known to a CA. Usually, this is via an RA.

Initialization– This deals with the basic problems, such as the methodology of verifying that the end-entity is talking to the right CA.

Certification—In this step, the CA creates a digital certificate for the end-entity and returns it to the end-entity, maintains a copy for its own records, and also copies it in public directories, if required.

Key-Pair Recovery Keys— used for encryption may be required to be recovered at a later date for decrypting some old documents. Key archival and recovery services can be provided by a CA or by an independent key-recovery system.

Key Generation PKIX specifies that the end-entity should be able to generate private-and public-key pairs, or the CA/RA should be able to do this for the end-entity (and then distribute these keys securely to the end-entity).

Key Update

This allows a smooth transition from one expiring key pair to a fresh one, by the automatic renewal of digital certificates. However, there is a provision for manual digital certificate renewal request and response.

Cross-certification

Helps in establishing trust models, so that end-entities that are certified by different CAs can cross-verify each other.

Revocation PKIX provides support for the checking of the certificate status in two modes: online (using OCSP) or offline (using CRL).

OR

PKIX stands for Public Key Infrastructure X.509. The X.509 standard defines the structure, format, and fields for digital certificates, it also specifies the procedures for distributing public keys. To extend these standards and make them universal IETF (Internet Engineering Task Force) forms the PKIX working group. PKIX extends the basic philosophy of the X.509 standard and specifies the implementation of digital certificates in the world of the Internet. Additionally, other PKI models have been defined for application use in various domains. For example, ANSI, ASC X9F standards are used by financial organizations.

PKIX Services :

Services provided by are as follows:

- **Registration:** It is a process where end-entity registers itself to a CA. Usually, the registration is done via the RA.
- **Initialization:** This deals with basic problems such as the methodology of verifying that the end entity is talking to the right CA.
- **Certification:** It is a process where CA creates a digital certificate for end-entity and returns it to the end entity. CA also maintain a copy for certificate for its records. If required, CA also copied it in public directories.
- **Key pair recovery:** Keys which are used for encrypting documents may be required to be recovered later for decrypting the same old documents. Key archival and recovery services can be provided by CA or by an independent key recovery system.
- **Key generation:** PKIX model specifies that the end entity should be able to generate the public key and private key pairs or CA should be able to do this for the end entity.
- **Key update:** It is a process where the expired key of the digital certificate is automatically renewed and replaced with a new key pair. However, there is a provision for manual digital certificate renewal requests and responses.
- **Cross certification:** It is a process where end entities that re-certified by different CA, can cross verify each other. It helps in establishing trust models.
- **Revocation:** PKIX model provides support for checking certificate status in two modes, online using OCSP and offline using CRL.

PKIX Architectural Model

PKIX has developed a document that describes five areas of its architectural model. These areas are as follows:

1. 509 V3 certificate and V2 certificate revocation list profiles

X.509 standard allows the use of various options while describing the extension of digital certificates. PKIX has grouped all options that are deemed fit for internet users. It calls this group of options as the profile of internet users. This profile is described in RFC2459 and specifies which attributes must/may/may not be supported. Appropriate value ranges for the values used in each extension category are also provided. For instance, the X.509 standard does not specify the instruction codes when the certificate is suspended. PKIX defines them.

2. *Operational protocols*

These define the underlying protocols that provide the transport mechanism for delivering certificates. CRLs and other management and status information to PKI users. Since each of these requirements demands a different way of service, how to use HTTP, LDAP, FTP, X.500, etc. are defined for this purpose.

3. Management Protocols

These protocols enable exchange information between various PKI entities. For example, how to carry registration request revocation status or cross-certification request and response. The management protocol specifies the structure of the message that floats between the entities. They also specify what details are required to process these messages. Examples of management protocols include CMP (Certificate Management Protocol) for requesting a certificate.

4. Policy outlines

PKIX defines the outlines for CP (Certificate Policies) and CPS (Certificate Practice Statements) in RFC2527. These define the policies for the creation of a document such as certificate policies which determine what considerations are important when choosing a type of certificate for a particular application domain.

5. Timestamp and data certification service

Time stamping service is provided by a trusted third party which is called Time Stamp Authority. The main purpose of this service is to sign a message to guarantee that it existed before a specific date and time. This helps deal with non-repudiation claims. DCS (Data certification Service) is a trusted third party's service that verifies the correctness of the data that it receives. This is similar to the notary service in real life, where for instance, it can use it for getting one's property certified.

Conclusion

PKIX stands for Public Key Infrastructure X.509 standard is a model which deals with the issue related to PKI technology i.e. Public Key Infrastructure. In this article, we have discussed the concept of PKIX with its working, Services, and architecture.

4.5] Public Key Cryptography Standards (PKCS)

Public key cryptography standards may not be in the everyday lexicon of a business, but every business in every industry runs on it. Large technology companies make a concerted effort to integrate these standards into their infrastructure. Companies in sectors such as healthcare, education, and even government unwittingly subscribe to these standards when they incorporate various [SaaS](#) solutions and cloud-based infrastructure decisions into their systems.

At an individual level, every website and application accessed from a personal device is also subject to PKCS standards. It makes sense, considering how much personal information flows into an online shopping or dating app.

The reasons that PKCS are so essential are:

1. Security.

There isn't a single business in any industry today that is not connected to the internet. Even basic mailing software is subject to cyber-attacks. According to the Cost of a Data Breach 2022 Report by IBM, the average cost of a [data breach](#) in the United States is \$9.44 million.

The PKCS standards ensure no gaps while implementing encryption across these different applications consumed. They provide an encryption checklist for [cyber security](#) personnel.

2. Compliance

With so much private and sensitive data doing the rounds, governing bodies across all industries have created regulations to control how this data is transmitted and stored. Encrypted data is usually the first mandate in all these regulations.

For example, HIPAA regulations in the healthcare industry require that patients' protected health information (PHI) and electronic PHI (ePHI) must be encrypted while stored and in transit. Failing to comply with HIPAA can cost around \$50,000 per violation.

3. Versatility

New devices and exceptional hardware capabilities are surfacing every day. The concept of stand-alone hardware is moot now, with every device recording, transmitting, or transmitting some form of data.

The [Internet-of-things](#) (IoT) has caught on, with data constantly being uploaded and used to make operations more efficient. An architecture that complies with the PKCS standards makes it easier to adapt these new technologies without necessarily uprooting the sections of the existing system.

4. Interoperability

The most significant advantage of using PKCS is the ability to securely allow various hardware and software forms to communicate without many development overheads. This makes it easier to explore different solutions and vendors. It also works well in the current cloud-driven infrastructure models.

The 15 PKCS Specifications

Each PKCS standard deals with a specific aspect of the PKI. They are reviewed and maintained individually and updated as necessary. The 15 standards are:

1. PKCS #1 – RSA cryptography standard

Description: This is the base standard that defines the syntax of the public and private keys, the format of the messages and digital signatures, the different algorithms to be used for each level of encryption and decryption, and the format of the digital certificate.

Besides the syntax of the key, it also specifies which mathematical requirements the keys must satisfy to be unbreachable for [brute-force attacks](#). Some libraries implementing PKCS#1 are BSAFE, cryptlib, OpenSSL, and Nettle.

Is this standard still maintained? Yes. It was also republished as RFC 8017 in 2012. RFC is a formal document maintained by the Internet engineering task force (IETF) that contains specifications and notes for everything related to computing and networking.

Current version: 2.2

2. PKCS #2 – RSA message digest standard

Description: The PKCS #2 standard defines how message digests must be created and encrypted.

Is this standard still maintained? No, this standard was withdrawn in 2010 and merged with PKCS #1.

Current version: Nil

3. PKCS #3 – Diffie-Helman key agreement standard

Description: This standard is named after Diffie and Helman, who first developed the public/private key mode of encryption and decryption. It defines how the receiver and sender can establish a secret key to envelope private communication.

This key is assumed to be created and shared in a non-secure [network](#). Many key establishment schemes, such as ANSI X9.63 and IEEE 1363a, come into play here. Organizations such as Oracle have their own implementation of this standard.

Is this standard still maintained? Yes, it is still being used.

Current version: 1.4

4. PKCS #4 – RSA key syntax standard

Description: This standard covers the key syntax to be used by RSA.

Is this standard still maintained? No. PKCS #4 was withdrawn in 2010 and merged with the first standard.

Current version: Nil

5. PKCS #5 – Password-based cryptography standard

Description: PKCS #5 explores the usage of passwords in the key agreement phase. It specifies how one should apply pseudo-random functions such as HMAC to the password along with a salt value. This text derived from the password now forms the secret key. This additional layer of encryption makes up for the pitfalls of using plain, user-fed passwords during communication.

Password-Based Key Derivation Function 2 (PBKDF2) is the recommended implementation by PKCS #5's version 2.1. It involves hashing the password repeatedly to derive the key. The number of repetitions is based on the CPU speed. The more repetitions, the harder it is to break. However, newer and more robust implementations such as Argon2 and SCrypt are being used today.

Is this standard still maintained? Yes, this standard still plays a crucial role in encrypted communication. It has also been published as Internet Engineering Task Force's RFC 2898.

Current version: 2.1

6. PKCS #6 – Extended-certificate syntax standard

Description: PKCS #6 was meant to provide the syntax for extending and modifying the attributes established in a digital certificate.

Is this standard still maintained? No, this standard was made obsolete by the third version of X.509. X.509 is a different standard that defines the format of public key certificates.

Current version: Nil

7. PKCS #7 – Cryptographic message syntax standard

Description: Cryptographic message syntax (CMS) standard specifies the syntax of stored, encrypted data. PKCS#7 is used by certificate authorities to store digital certificates that they've issued.

It also specifies how to handle certificates that have been invalidated for reasons other than expiry. Any storage of digital signatures should also adhere to PKCS #7. One implementation of this standard can be seen in [Single sign-on](#) applications.

Is this standard still maintained? This standard forms the basis for RFC 5652, and while still being implemented, it is being taken over by RFC 3369.

Current version: 1.5

8. PKCS #8 – Private key information syntax standard

Description: As the name suggests, this standard defines the syntax for private keys. It explains which algorithms and attributes suit the best to create a private key. It also looks into how users must store private keys.

Usually, this standard is used in conjunction with PKCS #5, using a passcode and a salt to store private keys.

Is this standard still maintained? This standard is maintained and available as RFC 5208.

Current version: 1.2

9. PKCS #9 – Selected attribute types standard

Description: PKCS #9 defines the data type, length, and other details of attributes necessary for certificates, signatures, and private keys. It does not give the format of the certificate or the key itself. It deals with the attributes or details necessary to create one.

For example, a business name and URL are necessary attributes for certain digital certificates.

Is this standard still maintained? PKCS #9 is still being supported.

Current version: 2.0

10. PKCS #10 – Certification request syntax standard

Description: PKCS #10 specifies the format of the messages sent to a certificate authority (CA) to request a digital certificate. This is called a certificate signing request.

These messages usually contain the public key chosen by a business, domain name or other identifying details, and proof of authenticity.

Is this standard still maintained? Yes, this standard is still being maintained.

Current version: 1.7

11. PKCS #11 – Cryptographic token interface standard

Description: The cryptographic token interface standard is also known as Cryptoki. It describes a platform-independent [application programming interface](#) (API) that can be used by cryptography token generators such as smart cards and key fobs.

The API is a generic interface that allows the generation, modification, and deletion of the different types of keys and certificates used by security hardware.

This standard ensures that encryption is platform-agnostic and allows various devices to communicate without extensive workarounds.

Cryptoki serves well in scenarios that require single sign-on and disk encryption systems. Certificate authorities use it to access the CA's signing key and to generate new certificates. Organizations such as Oracle Solaris and Red Hat have their own implementations for proprietary use.

Is this standard still maintained? PKCS #11 is still being supported, with RSA Security turning it over to the OASIS PKCS 11 technical committee for all further work.

Current version: 3.0

12. PKCS #12 – Personal information exchange syntax standard

Description: PKCS #12 defines the file format in which the private key issued with the public key is stored. It also specifies how to store various other private information, such as the digital certificate.

More often than not, all the private encryption information is bundled and stored in internal containers called Safe Bags. Some Safe Bags have a predefined structure that enables the storage of certificate requests, X.509 certificates, and private keys. Some Safe Bags are more flexible and allow the storage of custom, private information.

Besides storage, PKCS #12 also provides the mechanism for transferring this data across a network, along with the necessary syntax. PKCS #12 is built on PKCS #8 which only deals with private keys. It adds an extra layer to check for the integrity of this data both in transit and storage.

Is this standard still maintained? Yes, PKCS #12 is still being maintained.

Current version: 1.1

13. PKCS #13 – Elliptic curve cryptography standard

Description: The elliptic curve cryptography (ECC) standard was a type of cryptography that used modified mathematical methods in encryption.

The current public key algorithm is the RSA algorithm, created by Rivest, Shamir, and Adelman. It involves the modular exponentiation of the product of two large prime numbers.

ECC tries to find a distinct logarithm within a random elliptic curve. This yields more secure keys than RSA but is still in the exploration phase.

Is this standard still maintained? While proposed initially in 1998, there is no documentation of this standard available.

Current version: Nil

14. PKCS #14 – Pseudo-random number generation standard

Description: Random number generation plays a significant role in PKI, particularly with enveloping and hashing. It makes sense to standardize this aspect of encrypted communication. However, progress has yet to be made on this front.

Is this standard still maintained? No, there is no documentation of PKCS #14 that is available as of today.

Current version: Nil

15. PKCS #15 – Cryptographic token information format standard

Description: While all these standards have been established keeping interoperability in mind, it also makes sense to specify a standard that will allow for customization of certain aspects of encryption.

For example, integrated circuit cards are cryptographic tokens used across various devices. PKCS #15 specified the format of the credentials required by these tokens to identify themselves to the user. This is akin to digital certificates but at a token level.

Is this standard still maintained? PKCS #15 is still supported, though some nuances have been handed over to ISO/IEC 7816-15.

Current version: 1.1

4.6] XML

XML encryption is the process of encrypting and decrypting digital XML content, using certain syntax and algorithms. The basic concept of cryptography remains much the same, but the difference is in adopting a standard format for representing and exchanging encrypted XML data.

This standard format, often known as the XML encryption specification, includes a standard syntax for representing the encrypted content within the XML, as well as information required to decrypt its contents at the receiving end.

The W3C XML Encryption Working Group—a collaborative effort between the [W3C](#) (World Wide Web Consortium) and the [IETF](#) (Internet Engineering Task Force)—advocates XML encryption standards and specifications. (W3C and IETF are two of the frontline organizations involved in defining interoperable standards and specifications for XML/Web services technologies.)

As of March 2002, the Working Group has released the [Candidate Recommendation Specification](#) for XML encryption, which includes the following:

- XML Encryption Requirements
- XML Encryption Syntax and Processing

Inspirations behind the XML Encryption specification effort, can be summarized as follows:

- It is good to have standard element tags for representing encrypted elements within the XML documents. This will enable parsers to better understand encrypted elements and data during the validation process.
- It is necessary to provide means for encrypting only the desired elements within an XML document instead of encrypting the whole document. This will pave the way for incorporating several confidential data elements that are intended for different recipients within a single XML document.

- There should be standard mechanisms for exchanging the secret keys used for encryption and decryption processes.
- The standard should allow encryption of different parts of the document with different keys, so that multiple recipients can decrypt only those portions that are intended for them.
- The standards should be adaptable to both ASCII and binary data.
- The standards should be adaptable to different cryptographic algorithms.
- The standards should work along with other XML security standards and specifications

4.7] PKI And Security

Public key infrastructure or PKI is the governing body behind issuing digital certificates. It helps to protect confidential data and gives unique identities to users and systems. Thus, it ensures security in communications.

The public key infrastructure uses a pair of keys: the public key and the private key to achieve security. The public keys are prone to attacks and thus an intact infrastructure is needed to maintain them.

Managing Keys in the Cryptosystem:

The security of a cryptosystem relies on its keys. Thus, it is important that we have a solid key management system in place. The 3 main areas of key management are as follows:

- A cryptographic key is a piece of data that must be managed by secure administration.
- It involves managing the key life cycle which is as follows:
 - **Keeping the private key secret:** Only the owner of a private key is authorized to use a private key. It should thus remain out of reach of any other person.
 - **Assuring the public key:** Public keys are in the open domain and can be publicly accessed. When this extent of public accessibility, it becomes hard to know if a key is correct and what it will be used for. The purpose of a public key must be explicitly defined.

PKI or public key infrastructure aims at achieving the assurance of public key.

Public Key Infrastructure:

Public key infrastructure affirms the usage of a public key. PKI identifies a public key along with its purpose. It usually consists of the following components:

- A digital certificate also called a public key certificate
- Private Key tokens
- Registration authority
- Certification authority
- CMS or Certification management system

Working on a PKI:

Let us understand the working of PKI in steps.

- **PKI and Encryption:** The root of PKI involves the use of [cryptography](#) and [encryption](#) techniques. Both [symmetric](#) and [asymmetric encryption](#) uses a public key. The challenge here is – “how do you know that the public key belongs to the right person or to the person you think it belongs to?”. There is always a risk of MITM(Man in the middle). This issue is resolved by a PKI using digital certificates. It gives identities to keys in order to make the verification of owners easy and accurate.
- **Public Key Certificate or Digital Certificate:** Digital certificates are issued to people and electronic systems to uniquely identify them in the digital world. Here are a few noteworthy things about a digital certificate. Digital certificates are also called X.509 certificates. This is because they are based on the ITU standard X.509.
 - The Certification Authority (CA) stores the public key of a user along with other information about the client in the digital certificate. The information is signed and a digital signature is also included in the certificate.
 - The affirmation for the public key then thus be retrieved by validating the signature using the public key of the Certification Authority.
- **Certifying Authorities:** A CA issues and verifies certificates. This authority makes sure that the information in a certificate is real and correct and it also digitally signs the certificate. A CA or ***Certifying Authority performs these basic roles:***
 - Generates the key pairs – This key pair generated by the CA can be either independent or in collaboration with the client.
 - Issuing of the digital certificates – When the client successfully provides the right details about his identity, the CA issues a certificate to the client. Then CA further signs this certificate digitally so that no changes can be made to the information.
 - Publishing of certificates – The CA publishes the certificates so that the users can find them. They can do this by either publishing them in

an electronic telephone directory or by sending them out to other people.

- Verification of certificate – CA gives a public key that helps in verifying if the access attempt is authorized or not.
- Revocation – In case of suspicious behavior of a client or loss of trust in them, the CA has the power to revoke the digital certificate.

Classes of a Digital Certificate:

A digital certificate can be divided into four broad categories. These are :

- Class 1: These can be obtained by only providing the email address.
- Class 2: These need more personal information.
- Class 3: This first checks the identity of the person making a request.
- Class 4: They are used by organizations and governments.

Process of creation of certificate:

The creation of a certificate takes place as follows:

- Private and public keys are created.
- CA requests identifying attributes of the owner of a private key.
- Public key and attributes are encoded into a CSR or Certificate Signing Request.
- Key owner signs that CSR to prove the possession of a private key.
- CA signs the certificate after validation.

Creation of Trust layers among CA Hierarchies:

Each CA has its own certificate. Thus, trust is built hierarchically where one CA issues certificates to other CAs. Moreover, there is a root certificate that is self-signed. For a root CA, the issuer and the subject are not two separate parties but a single party.

Security of Root CA:

As you saw above, the ultimate authority is the root CA. Hence, the security of root CA is of huge importance. If the private key of a root CA is not taken care of, then it might turn into a catastrophe. This is because anyone disguised as the root CA can then issue certificates. To meet security standards, a root CA should be offline 99.9% of the time. However, it does need to come online to create public and private keys and to issue new certificates. Ideally, these activities should be performed 2-4 times a year.

Use of PKI in Today's Digital Age:

Today, there are an enormous number of applications that need require authentication. Certifications are needed at millions of places. This can not be done without a Public key infrastructure. The importance of PKI, depending on the use case and needs, has evolved over time. Here is a part of that track.

- For the very first time during the period of 1995 to 2002, the use of PKI was limited to the most important and high-value certificates. This included the certificates of eCommerce websites that enabled them to display the lock icon in the search bar. The goal was to make consumers confident about the security and authenticity of various websites.
- The second episode of PKI emerged around 2003 to 2010 when enterprises came into the picture. It was at this time that employees received laptops and the use of mobile phones was rising. Thus, employees needed access to the organization's assets even outside the office. That is when the use of PKI looked like the best way for authentication.
- The third phase started in 2011 and is continuing to date. With the advent of new technologies like IoT(Internet of Things) and need the to scale PKI, the use, as well as the challenges in using PKI, have increased tremendously. Today, millions of certificates are issued to authenticate mobile workforces. However, managing this huge number of certificates is quite challenging.
- S/MIME, Document Signing, code or app signing also uses PKI.

Challenges that a PKI Solves:

PKI owes its popularity to the various problems its solves. Some use cases of PKI are:

- Securing web browsers and communicating networks by SSL/TLS certifications.
- Maintaining Access Rights over Intranets and VPNs.
- Data Encryption
- Digitally Signed Software
- Wi-fi Access Without Passwords

Other than these, one of the most important use cases of PKI is based around IoT(Internet of Things). Here are two industries that are using PKI for IoT devices:

- **Auto Manufacturers:** Cars these days have features like GPS, call for services, assistants, etc. These require communication paths where a lot of data is passed. Making these connections secure is very important to avoid malicious parties hacking into the cars. This is where PKI comes in.
- **Medical device Manufacturers:** Devices like surgical robots require high security. Also, FDA mandates that any next-generation medical device must be updatable so that bugs can be removed and security issues can be dealt with. PKI is used to issues certificates to such devices.

Disadvantages of PKI:

- **Speed:** Since PKI uses super complex algorithms to create a secure key pair. So it eventually slows down the process and data transfer.
- **Private Key Compromise:** Even though PKI can't be hacked very easily but a private key can be hacked by a professional hacker, since PKI uses Public and Private key to encrypt and decrypt data so with user's private key in hand and public key which is easily available the information can be decrypted easily.

4.8] Creating Digital Certificate

The steps required to create a [digital certificate](#) involves three parties first the end user, second the registration authority and third is certificate authority. The end user request for a digital certificate and the request goes to the registration authority(RA) which then assist the certificate authority(CA) to create the digital certificate. Registration authority act as a intermediate between end user and the certificate authority. It also assist in day to day task of certificate authority.

Services of Registration Authority:

- Accepting and verifying the details of new user's registration.
- User key generation.
- Backups and recovery of key.
- Certificate cancellation.

Steps for Digital Certificate Creation:

- **Step-1:** Key generation is done by either user or registration authority. The public key which is generated is sent to the registration authority and private key is kept secret by user.
- **Step-2:** In the next step the registration authority registers the user.
- **Step-3:** Next step is verification which is done by registration authority in which the user's credentials are being verified by registration authority. It also checks that the user who send the public key have corresponding private key or not.
- **Step-4:** In this step the details are sent to certificate authority by registration authority who creates the digital certificate and give it to users and also keeps a copy to itself.

Unit 5 – Internet Security Protocols

5.1] Introduction And Concepts

Internet security refers to securing communication over the internet. It includes specific security protocols such as:

- Internet Security Protocol (IPSec)
- Secure Socket Layer (SSL)

❖ Internet Security Protocol (IPSec)

It consists of a set of protocols designed by Internet Engineering Task Force (IETF). It provides security at network level and helps to create authenticated and confidential packets for IP layer.

❖ Secure Socket Layer (SSL)

It is a security protocol developed by Netscape Communications Corporation.). It provides security at transport layer. It addresses the following security issues:

- Privacy
- Integrity
- Authentication

❖ Threats

Internet security threats impact the network, data security and other internet connected systems. Cyber criminals have evolved several techniques to threat privacy and integrity of bank accounts, businesses, and organizations.

Following are some of the internet security threats:

- Mobile worms
- Malware
- PC and Mobile ransomware
- Large scale attacks like Stuxnet that attempts to destroy infrastructure.
- Hacking as a Service
- Spam
- Phishing

❖ Email Phishing

Email phishing is an activity of sending emails to a user claiming to be a legitimate enterprise. Its main purpose is to steal sensitive information such as usernames, passwords, and credit card details.

Such emails contains link to websites that are infected with malware and direct the user to enter details at a fake website whose look and feels are same to legitimate one.

- What a phishing email may contain?

Following are the symptoms of a phishing email:

Spelling and bad grammar

Most often such emails contain grammatically incorrect text. Ignore such emails, since it can be a spam.

Beware of links in email

Don't click on any links in suspicious emails.

Threats

Such emails contain threat like “your account will be closed if you didn't respond to an email message”.

Spoofing popular websites or companies

These emails contain graphics that appear to be connected to legitimate website but they actually are connected to fake websites.



Internet Protocol Security (IPSec) is a framework of open standards for ensuring private, secure communications over Internet Protocol (IP) networks, through the use of [cryptographic security](#) services. IPSec is a suite of cryptography-based protection services and security protocols. Because it requires no changes to programs or protocols, you can easily deploy IPSec for existing networks.

The driving force for the acceptance and deployment of secure IP is the need for business and government users to connect their private WAN/LAN infrastructure to the Internet for providing access to Internet

services and use of the Internet as a component of the WAN transport system. As we all know, users need to isolate their networks and at the same time send and receive traffic over the Internet. The authentication and privacy mechanisms of secure IP provide the basis for a security strategy for us.

IPsec protects one or more paths between a pair of hosts, a pair of security gateways, or a security gateway and a host. A security gateway is an intermediate device, such as a switch or firewall, that implements IPsec. Devices that use IPsec to protect a path between them are called peers.

IPsec requires a PCI Accelerator Card (PAC) to provide hardware data compression and encryption. A PAC is a hardware processing unit the switch's CPU controls.

IPsec provides the following security services for traffic at the IP layer:

- Data origin authentication—identifying who sent the data.
- Confidentiality (encryption)—ensuring that the data has not been read en route.
- Connectionless integrity—ensuring the data has not been changed en route.
- Replay protection—detecting packets received more than once to help protect against denial of service attacks.

Applications of IPsec

As we all know to help in the security of a network the Internet community has done lot of work and developed application-specific security mechanisms in numerous application areas, including electronic mail (*Privacy Enhanced Mail*, *Pretty Good Privacy* [PGP]), network management (*Simple Network Management Protocol Version 3* [SNMPv3]), Web access (*Secure HTTP*, *Secure Sockets Layer* [SSL]), and others. However, users have some security concerns that cut across protocol layers. For example, an enterprise can run a secure, private TCP/IP network by disallowing links to untrusted sites, encrypting packets that leave the premises, and authenticating packets that enter the premises. By implementing security at the IP level, an organization can ensure secure networking not only for applications that have security mechanisms but also for the many security-ignorant applications.

Benefits of IPSec :

- When IPSec is implemented in a firewall or router, it provides strong security whose application is to all traffic crossing this perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.
- IPSec is below the transport layer (TCP, UDP), and is thus transparent to applications. There is no need to change software on a user or server system when IPSec is implemented in the firewall or router.
- Even if IPSec is implemented in end systems, upper layer software, including applications is not affected. IPSec can be transparent to end users.
- There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization. IPSec can provide security for individual users if needed. This feature is useful for offsite workers and also for setting up a secure virtual sub network within an organization for sensitive applications.

5.2] Secure Socket Layer

[Secure Socket Layer \(SSL\)](#) provides security to the data that is transferred between web browser and server. SSL encrypts the link between a web server and a browser which ensures that all data passed between them remain private and free from attack.

Secure Socket Layer Protocols:

- SSL record protocol
- Handshake protocol
- Change-cipher spec protocol
- Alert protocol

SSL Protocol Stack: Diag

SSL Record Protocol:

SSL Record provides two services to SSL connection.

- Confidentiality
- Message Integrity

In the SSL Record Protocol application data is divided into fragments. The fragment is compressed and then encrypted MAC (Message Authentication Code) generated by algorithms like SHA (Secure Hash Protocol) and MD5 (Message Digest) is appended. After that encryption of the data is done and in last SSL header is appended to the data. (Diag)

Handshake Protocol:

Handshake Protocol is used to establish sessions. This protocol allows the client and server to authenticate each other by sending a series of messages to each other.

Handshake protocol uses four phases to complete its cycle.

- **Phase-1:** In Phase-1 both Client and Server send hello-packets to each other. In this IP session, cipher suite and protocol version are exchanged for security purposes.
- **Phase-2:** Server sends his certificate and Server-key-exchange. The server ends phase-2 by sending the Server-hello-end packet.
- **Phase-3:** In this phase, Client replies to the server by sending his certificate and Client-exchange-key.
- **Phase-4:** In Phase-4 Change-cipher suite occurs and after this the Handshake Protocol ends.

SSL Handshake Protocol Phases diagrammatic representation

Change-cipher Protocol:

This protocol uses the SSL record protocol. Unless Handshake Protocol is completed, the SSL record Output will be in a pending state. After the handshake protocol, the Pending state is converted into the current state.

Change-cipher protocol consists of a single message which is 1 byte in length and can have only one value. This protocol's purpose is to cause the pending state to be copied into the current state.

Alert Protocol:

This protocol is used to convey SSL-related alerts to the peer entity. Each message in this protocol contains 2 bytes.

The level is further classified into two parts:

Warning (level = 1):

This Alert has no impact on the connection between sender and receiver. Some of them are:

Bad certificate: When the received certificate is corrupt.

No certificate: When an appropriate certificate is not available.

Certificate expired: When a certificate has expired.

Certificate unknown: When some other unspecified issue arose in processing the certificate, rendering it unacceptable.

Close notify: It notifies that the sender will no longer send any messages in the connection.

Unsupported certificate: The type of certificate received is not supported.

Certificate revoked: The certificate received is in revocation list.

Fatal Error (level = 2):

This Alert breaks the connection between sender and receiver. The connection will be stopped, cannot be resumed but can be restarted. Some of them are :

Handshake failure: When the sender is unable to negotiate an acceptable set of security parameters given the options available.

Decompression failure: When the decompression function receives improper input.

Illegal parameters: When a field is out of range or inconsistent with other fields.

Bad record MAC: When an incorrect MAC was received.

Unexpected message: When an inappropriate message is received.

The second byte in the Alert protocol describes the error.

Salient Features of Secure Socket Layer:

- The advantage of this approach is that the service can be tailored to the specific needs of the given application.
- Secure Socket Layer was originated by Netscape.
- SSL is designed to make use of TCP to provide reliable end-to-end secure service.
- This is a two-layered protocol.

Versions of SSL:

SSL 1 – Never released due to high insecurity.

SSL 2 – Released in 1995.

SSL 3 – Released in 1996.

TLS 1.0 – Released in 1999.

TLS 1.1 – Released in 2006.

TLS 1.2 – Released in 2008.

TLS 1.3 – Released in 2018.

SSL (Secure Sockets Layer) certificate is a digital certificate used to secure and verify the identity of a website or an online service. The certificate is issued by a trusted third-party called a Certificate Authority (CA), who verifies the identity of the website or service before issuing the certificate.

The SSL certificate has several important characteristics that make it a reliable solution for securing online transactions:

1. **Encryption:** The SSL certificate uses encryption algorithms to secure the communication between the website or service and its users. This ensures that the sensitive information, such as login credentials and credit card information, is protected from being intercepted and read by unauthorized parties.
2. **Authentication:** The SSL certificate verifies the identity of the website or service, ensuring that users are communicating with the intended party and not with an impostor. This provides assurance to users that their information is being transmitted to a trusted entity.
3. **Integrity:** The SSL certificate uses message authentication codes (MACs) to detect any tampering with the data during transmission. This ensures that the data being transmitted is not modified in any way, preserving its integrity.
4. **Non-repudiation:** SSL certificates provide non-repudiation of data, meaning that the recipient of the data cannot deny having received it. This is important in situations where the authenticity of the information needs to be established, such as in e-commerce transactions.
5. **Public-key cryptography:** SSL certificates use public-key cryptography for secure key exchange between the client and server. This allows the client and

server to securely exchange encryption keys, ensuring that the encrypted information can only be decrypted by the intended recipient.

6. **Session management:** SSL certificates allow for the management of secure sessions, allowing for the resumption of secure sessions after interruption. This helps to reduce the overhead of establishing a new secure connection each time a user accesses a website or service.
7. **Certificates issued by trusted CAs:** SSL certificates are issued by trusted CAs, who are responsible for verifying the identity of the website or service before issuing the certificate. This provides a high level of trust and assurance to users that the website or service they are communicating with is authentic and trustworthy.

5.3] Transport Layer Security

Transport Layer Securities (TLS) are designed to provide security at the transport layer. TLS was derived from a security protocol called [Secure Socket Layer \(SSL\)](#). TLS ensures that no third party may eavesdrop or tamper with any message. There are several benefits of TLS:

- **Encryption:**
TLS/SSL can help to secure transmitted data using encryption.
- **Interoperability:**
TLS/SSL works with most web browsers, including Microsoft Internet Explorer and on most operating systems and web servers.
- **Algorithm flexibility:**
TLS/SSL provides operations for authentication mechanism, encryption algorithms and hashing algorithm that are used during the secure session.
- **Ease of Deployment:**
Many applications TLS/SSL temporarily on a windows server 2003 operating systems.
- **Ease of Use:**
Because we implement TLS/SSL beneath the application layer, most of its operations are completely invisible to client.

Working of TLS:

The client connect to server (using [TCP](#)), the client will be something. The client sends number of specification:

1. Version of SSL/TLS.
2. which cipher suites, compression method it wants to use.

The server checks what the highest SSL/TLS version is that is supported by them both, picks a cipher suite from one of the clients option (if it supports one) and optionally picks a compression method. After this the basic setup is done, the server provides its certificate. This certificate must be trusted either by the client itself or a party that the client trusts. Having verified the certificate and being certain this server really is who he claims to be (and not a man in the middle), a key is

exchanged. This can be a public key, “PreMasterSecret” or simply nothing depending upon cipher suite.

5.4] Secure Hypertext Transport Protocol

Secure Hypertext Transfer Protocol (S-HTTP) is a protocol used for encrypting [HTTP traffic](#), enhancing the security of data as it travels across the [World Wide Web](#). Essentially, S-HTTP is designed to secure individual messages transmitted between clients and servers, differentiating itself from other security protocols by its granularity and message-oriented approach.

- **Basic Functionality and Principles**

The primary functionality of S-HTTP is to provide end-to-end encryption of web communications. It achieves this by encrypting the data before transmission and decrypting it upon receipt. This process ensures confidentiality and protects against eavesdropping and tampering. S-HTTP supports various encryption algorithms, enabling flexible security arrangements based on the needs of the communication parties. Furthermore, it provides options for authentication, ensuring that both sender and receiver can verify each other’s identity, adding an additional layer of security to web interaction. S-HTTP is also a server software.

The Secure Hypertext Transfer Protocol is not as widely implemented as Secure Sockets Layer ([SSL](#)), which is independent of protocol and works with HTTP, Simple Mail Transfer Protocol (SMTP), [Network News Transfer Protocol](#) (NNTP), and other Internet protocols.

S-HTTP is also the name given to World Wide Web (WWW) server software that implements the S-HTTP protocol. This software was developed by Enterprise Integrations Technologies (EIT), the National Center for Supercomputing Applications (NCSA), and RSA Security.

2. The Creation of S-HTTP

Historical Context and Origin

S-HTTP was developed in the early [1990s](#), a time when the internet was rapidly expanding, and the need for secure communication methods was becoming increasingly apparent. The protocol was developed to address the lack of inherent security in the HTTP protocol, which is the foundational technology for data communication on the World Wide Web.

The development of S-HTTP was primarily led by R. T. Fielding and J. C. Mogul. Their work was instrumental in the early stages of defining and standardizing the protocol. The key milestones in the development of S-HTTP include:

1. **Initial Proposal:** The concept of Secure Hypertext Transfer Protocol was proposed to address the growing concerns over web security and privacy.
2. **Collaboration and Standardization:** Collaborative efforts among various internet security experts and organizations led to the refinement and standardization of the protocol.
3. **RFC Publication:** The formal documentation of S-HTTP, known as RFC 2660, was published in 1999, detailing the specifications and use cases of the protocol. (see the [references section](#))

The creation of S-HTTP marked a significant step in the evolution of web security, laying the groundwork for future developments in secure web communication

3. How S-HTTP Works

Technical Mechanism and Encryption Process

S-HTTP operates by encrypting individual HTTP messages, unlike other protocols that encrypt the entire communication session. This mechanism provides flexibility in securing specific parts of the data being transmitted. Each message, whether it's a request from the client or a response from the server, is encrypted independently.

The encryption process involves several key steps:

1. **Encryption Initiation:** When a message is ready for transmission, S-HTTP initiates encryption using a selected algorithm.
2. **Key Exchange:** S-HTTP supports various key exchange mechanisms. These keys are used to encrypt and decrypt messages at both ends.
3. **Data Encryption:** The actual data in the HTTP message is encrypted using the agreed-upon encryption standards, rendering it unreadable to unauthorized parties.
4. **Transmission and Decryption:** Once the encrypted message reaches its destination, it is decrypted using the corresponding decryption key.

Setting Up and Configuring S-HTTP

To set up S-HTTP:

1. **Server and Client Configuration:** Both the server and client must support S-HTTP. This involves configuring the web server and client applications to handle S-HTTP requests and responses.
2. **Certificate Management:** Obtain and install [digital certificates](#) to facilitate authentication and key exchange.
3. **Selecting Encryption Algorithms:** Choose suitable encryption algorithms based on the required level of security and the capabilities of the server and client.

4. S-HTTP vs. HTTPS: Understanding the Differences

Comparative Analysis of Protocols

While both S-HTTP and [HTTPS](#) are designed to secure HTTP communications, they differ fundamentally in their approach:

- **Encryption Scope:** Secure-HTTP encrypts individual messages, whereas HTTPS encrypts the entire communication session.
- **Flexibility:** SHTTP offers more flexibility in terms of what parts of the communication are encrypted. HTTPS, on the other hand, provides a uniform layer of encryption over all data exchanged.
- **Compatibility:** The Secure Hypertext Transfer Protocol can coexist with regular HTTP on a single server, while HTTPS typically requires a dedicated port.

Use Cases and Suitability

- **S-HTTP:** Best suited for scenarios where selective encryption of messages is needed, such as in applications that only require certain parts of the communication to be secured.
- **HTTPS:** Ideal for general web browsing and transactions where uniform security is required for all data exchanges, like in e-commerce or online banking.

Understanding these differences is crucial for selecting the right protocol based on specific security requirements and the nature of the data being transmitted.

[Back to Index](#)

5. Current Relevance of S-HTTP

Is S-HTTP Still Used Today?

S-HTTP, despite its initial promise, has seen limited adoption in modern web communications. The protocol, while innovative, has been overshadowed by the widespread acceptance and implementation of HTTPS. Today, S-HTTP is rarely used in standard web applications or by major [web browsers](#), making it more of a historical footnote in the evolution of secure web communication protocols.

Modern Implementations and Legacy

Although S-HTTP is not prevalent in current web applications, its legacy persists in the form of ideas and methodologies that have influenced modern encryption techniques. The concept of selectively encrypting web messages has been integrated into other technologies and protocols, albeit in different forms. S-HTTP's emphasis on flexible encryption and message-specific security remains a relevant concept in the design of secure communication systems.

[Back to Index](#)

6. Challenges and Limitations of S-HTTP

Technical Constraints

One of the major technical challenges facing S-HTTP was the complexity of its implementation. The protocol's message-specific encryption approach required more processing power and presented challenges in efficiently managing encryption keys. Moreover, the necessity for both client and server to support S-HTTP added to the complexity, making it less attractive compared to more straightforward alternatives like HTTPS.

Adoption and Compatibility Issues

S-HTTP's adoption was hindered by several factors:

- **Limited Browser and Server Support:** Many web browsers and servers did not adopt S-HTTP, favoring the more straightforward HTTPS protocol.
- **Compatibility with Web Infrastructure:** The existing web infrastructure, including caching mechanisms and proxy servers, was optimized for HTTP and HTTPS, leading to compatibility issues with S-HTTP.
- **Emergence of HTTPS as a Standard:** The rapid adoption of HTTPS as the de facto standard for secure web communication further marginalized S-HTTP, as developers and organizations preferred the more universally accepted protocol.

These challenges limited the widespread adoption of S-HTTP, relegating it to a niche role in the history of web security protocol.

7. Future Outlook and Evolution of Secure Web Protocols

Emerging Trends in Web Security

The future of web security is likely to be shaped by several emerging trends:

1. **Advanced Encryption Techniques:** The development of more sophisticated encryption methods to counteract evolving cyber threats.
2. **Quantum Computing:** The advent of [quantum computing](#) poses new challenges to current encryption standards, driving the need for quantum-resistant protocols.
3. **AI and Machine Learning:** The integration of AI and [machine learning](#) for predictive threat detection and automated security responses.
4. **Increased Focus on End-User Privacy:** Growing concerns over data privacy are prompting the development of protocols that enhance user anonymity and data protection.

Predictions for the Future of Secure HTTP and Related Technologies

Finally, while S-HTTP itself may not see a revival, its principles could influence future protocols:

- **Selective Encryption:** The concept of selective encryption, central to Secure-HTTP, may find relevance in specialized applications requiring flexible security measures.
- **Incorporation into Hybrid Protocols:** Elements of the Secure Hypertext Transfer Protocol might be integrated into new or existing protocols to enhance their security capabilities, especially in environments where message-level encryption is crucial.

5.5] Time Stamping Protocol

Timestamping is a fundamental aspect of end-to-end digitization that enables the verification of the order of events in a digital system. In simpler terms, timestamping aids in maintaining the record when a document has been **digitally signed**. It is a critical tool in ensuring the accuracy and integrity of various digital transactions, including financial transactions, legal documents, and **digitally signed documents**. Needless to say, **digital signing** is a growing trend and will continue to do so. This shifting impetus towards **digital signing solutions** has created a corresponding demand for timestamping.

Despite its importance, enterprises are yet to understand its role in ensuring integrity and confidentiality in the digital ecosystem. A lack of **timestamping protocol** in your **enterprise workflow** can have significant consequences, including data tampering, data loss, and legal disputes.

Therefore, in this blog, we aim to explain timestamping by providing a comprehensive guide to its mechanisms and applications. We will delve into the technical aspects of timestamping, exploring how it functions.

What is Timestamping?

Timestamping is a process of assigning a unique identifier to a specific event or transaction that occurs in a digital system. It involves recording the time and date when an event or transaction occurred, along with additional metadata that may be necessary to verify its authenticity and integrity.

The **primary purpose of timestamping** is to ensure the order of events. It plays a critical part in the workflow, where the accuracy and integrity of data are of utmost importance. For instance, in financial transactions, timestamping is used to provide evidence of the timing and order of trades, helping to prevent fraud and ensure fair and transparent markets. In legal transactions, timestamping is often used to authenticate documents and establish a chain of custody, helping to prevent tampering and provide evidence in legal disputes.

The Need for Timestamping

Timestamping plays a critical role in **ensuring trust and credibility** in the digital ecosystem, but the question remains, what is the need for assigning and recording time to digitally signed documents? Let us delve a little deeper into the growing need for timestamping.

To understand the **need for timestamping** let us recap the concept of digital signatures. **Digital signatures** are equivalent to traditional handwritten signatures, used to signify the signer's intent to agree to the terms and conditions of a document or transaction. Employing a **Public Key Infrastructure (PKI)**-based digital signature to digitally execute a document ensures minimal manual intervention. While it undoubtedly has provided ease of doing business, implementing such a process requires addressing the issue of non-repudiation.

For instance, when a document is signed with a **PKI-based certificate**, it captures the individual's information and the signature's date and time. However, the date and time depend on the computer's local time, which is prone to tampering. One could change the computer's time to make an expired or revoked certificate appear valid. Therefore, ensuring the authenticity of the digital signature requires additional measures.

This is where timestamping comes to play. Enterprises can resolve the issue at hand by employing a **Time Stamping Authority (TSA)**, which leverages the **RFC 3161 Time-Stamp Protocol (TSP)**. This protocol substantiates that the data has not been subjected to any form of unauthorized modification, thereby ensuring the integrity of the information in question. By integrating the **use of a TSA** in the process of document or code signing, any attempts to tamper with the timestamp will be effectively prevented.

Trusted Timestamping

A **trusted timestamp** is a **digital timestamp** that is created and verified by a trusted **third-party timestamping authority (TSA)** using secure **FIPS-compliant hardware**. It is used to establish the authenticity and integrity of a digital document or message by providing a reliable record of the time at which it was created, sent, or received. It employs a **cryptographic algorithm** that generates unique code associated with a document or message.

The code is further encrypted using a **PKI certificate** which ensures that the timestamp is authentic and has not been tampered with. It is important to note that the reliability of a trusted timestamp depends on the TSA that creates and verifies the timestamp. Therefore, it is essential to choose a **reputable and trustworthy TSA, like eMudhra** to ensure that the trusted timestamp is reliable and can be used to establish the authenticity and integrity of the digital document or message.

How does Timestamping Work in Digital Documents?

The process of timestamping a digitally signed document can be concluded in the following steps:

- A hash code is created for the data or the document that is to be timestamped.
- The hash code is sent to a TSA (time stamping authority) which adds time to the data.
- Now, the hash value and the time are hashed together (typically a SHA-1/2 hash value) to create a unique hash value.
- The newly created unique hash value is then encrypted with TSA's private key using PKI technology.
- The digital signature of the document concatenated with the time stamp is sent to the original owner of the document.
- When the document is accessed, the encrypted code and timestamp can be decrypted using the PKI certificate to confirm that the document has not been altered since the timestamp was applied.

This provides a reliable record of the **document's authenticity and integrity** and can be used to demonstrate compliance with legal and regulatory requirements.

Timestamping in Network Security

In **network security**, **timestamping** protocols are used to provide a reliable and accurate way of recording the occurrence of events in a networked environment. The primary function of timestamping in this context is to establish a chronological order of events and to provide evidence of the occurrence of a particular event at a specific time.

One of the **primary applications of timestamping in network security** is in the prevention of replay attacks, where an attacker intercepts and replays a legitimate message to gain unauthorized access to a system. Timestamping can be used to prevent such attacks by ensuring that each message is unique and only valid for a specific time.

Furthermore, timestamping can also be used to provide evidence of compliance with legal or regulatory requirements, such as data retention laws. By accurately timestamping data, organizations can prove when specific events occurred and demonstrate that they have met their legal and regulatory obligations.

Overall, timestamping protocols play a critical role in network security by providing an accurate and reliable way of recording events in a networked environment. By ensuring the integrity and authenticity of data, timestamping helps to prevent attacks and provides a basis for legal and regulatory compliance.

Legal Framework of Timestamping Across the Globe

In India, the **use of timestamping** is recognized under the **Information Technology Act, of 2000**. The act defines **electronic signatures** and provides for their legal recognition, which includes the use of timestamping to establish the authenticity and integrity of electronic documents. The Indian government has also established a framework for **digital signatures and certificates**, which includes the use of a trusted third-party timestamping authority.

In the United States, it is governed by **federal and state laws**, as well as industry-specific regulations. The Electronic Signatures in **Global and National Commerce Act (ESIGN)** and the **Uniform Electronic Transactions Act (UETA)** provide legal recognition for electronic signatures, including the use of timestamping.

In Europe, the legality is governed by the **eIDAS Regulation**, which provides a framework for **electronic identification, authentication, and trust services**. The regulation recognizes its use as a means of establishing the authenticity and

integrity of documents and recognizes the use of a trusted third-party timestamping authority.

The **use of timestamping in digital transactions** is becoming increasingly important as more industries shift to digital platforms. While the legal framework varies across different countries and regions, there is a growing recognition of the **importance of timestamping** in ensuring the authenticity and integrity of documents.

5.5] Secure Electronic Transaction

Secure Electronic Transaction or SET is a system that ensures the security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied to those payments. It uses different encryption and hashing techniques to secure payments over the internet done through credit cards. The SET protocol was supported in development by major organizations like Visa, Mastercard, and Microsoft which provided its Secure Transaction Technology (STT), and Netscape which provided the technology of Secure Socket Layer (SSL).

SET protocol restricts the revealing of credit card details to merchants thus keeping hackers and thieves at bay. The SET protocol includes Certification Authorities for making use of standard Digital Certificates like X.509 Certificate.

Before discussing SET further, let's see a general scenario of electronic transactions, which includes client, payment gateway, client financial institution, merchant, and merchant financial institution. (Diag)

Requirements in SET: The SET protocol has some requirements to meet, some of the important requirements are:

- It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is an intended user or not, and merchant authentication.
- It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.
- It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.
- SET also needs to provide interoperability and make use of the best security mechanisms.

Participants in SET: In the general scenario of online transactions, SET includes similar participants:

1. **Cardholder** – customer
2. **Issuer** – customer financial institution

3. **Merchant**
4. **Acquirer** – Merchant financial
5. **Certificate authority** – Authority that follows certain standards and issues certificates (like X.509V3) to all other participants.

SET functionalities:

- **Provide Authentication**
 - **Merchant Authentication** – To prevent theft, SET allows customers to check previous relationships between merchants and financial institutions. Standard X.509V3 certificates are used for this verification.
 - **Customer / Cardholder Authentication** – SET checks if the use of a credit card is done by an authorized user or not using X.509V3 certificates.
- **Provide Message Confidentiality:** Confidentiality refers to preventing unintended people from reading the message being transferred. SET implements confidentiality by using encryption techniques. Traditionally DES is used for encryption purposes.
- **Provide Message Integrity:** SET doesn't allow message modification with the help of signatures. Messages are protected against unauthorized modification using RSA digital signatures with SHA-1 and some using HMAC with SHA-1,

Dual Signature: The dual signature is a concept introduced with SET, which aims at connecting two information pieces meant for two different receivers :

Order Information (OI) for merchant

Payment Information (PI) for bank

You might think sending them separately is an easy and more secure way, but sending them in a connected form resolves any future dispute possible. Here is the generation of dual signature: Diag.

Where,

PI stands for payment information

OI stands for order information

PIMD stands for Payment Information Message Digest

OIMD stands for Order Information Message Digest

POMD stands for Payment Order Message Digest

H stands for Hashing

E stands for public key encryption

KPc is customer's private key

|| stands for append operation

Dual signature, $DS = E(KPc, [H(H(PI)||H(OI))])$

Purchase Request Generation: The process of purchase request generation requires three inputs:

- Payment Information (PI)
- Dual Signature
- Order Information Message Digest (OIMD)

The purchase request is generated as follows:

Here,

PI, OIMD, OI all have the same meanings as before.

The new things are :

EP which is symmetric key encryption

Ks is a temporary symmetric key

KUbank is public key of bank

CA is Cardholder or customer Certificate

Digital Envelope = $E(KUbank, Ks)$

Purchase Request Validation on Merchant Side: The Merchant verifies by comparing POMD generated through PIMD hashing with POMD generated through decryption of Dual Signature as follows: Diag

Since we used Customer's private key in encryption here we use KUC which is the public key of the customer or cardholder for decryption 'D'.

Payment Authorization and Payment Capture: Payment authorization as the name suggests is the authorization of payment information by the merchant which ensures payment will be received by the merchant. Payment capture is the process by which a merchant receives payment which includes again generating some request blocks to gateway and payment gateway in turn issues payment to the merchant.

The disadvantages of Secure Electronic Exchange: At the point when SET was first presented in 1996 by the SET consortium (Visa, Mastercard, Microsoft, Verisign, and so forth), being generally taken on inside the following couple of years was normal. Industry specialists additionally

anticipated that it would immediately turn into the key empowering influence of worldwide internet business. Notwithstanding, this didn't exactly occur because of a few serious weaknesses in the convention.

The security properties of SET are better than SSL and the more current TLS, especially in their capacity to forestall web based business extortion. Be that as it may, the greatest downside of SET is its intricacy. SET requires the two clients and traders to introduce extraordinary programming – – card perusers and advanced wallets – – implying that exchange members needed to finish more jobs to carry out SET. This intricacy likewise dialed back the speed of web based business exchanges. SSL and TLS don't have such issues.

5.6] SSL Vs SET

S. No.	Secure Socket Layer	Secure Electronic Transaction
	<p style="text-align: center;">Basics-</p> <p>1. SSL is an encryption mechanism for order taking, queries, and other applications and is available on the customer's browser. It does not protect against all security hazards and is naturally simple and widely used. SSL is a protocol for general-purpose secure message exchange. SSL protocol may use a certificate, but the payment gateway is not available. So, the merchant needs to receive both the ordering information and credit card information because the capturing process should be generated by the merchant. SSL protocol has been the industry standard for securing internet communication.</p>	<p style="text-align: center;">Basics-</p> <p>SET is a very comprehensive protocol. It provides privacy, integration, and authenticity. It is not used frequently due to its complexity and the need for a special card reader by the user. It may be abandoned if it is not simplified. SET is tailored to the credit card payment to the merchant. SET protocols hide the customer's credit card information from merchant and also hides the order information from banks to protect privacy called a dual signature. The SET protocol is complex and more secure.</p>
2.	<p style="text-align: center;">Developed by-</p> <p>SSL protocol was developed by Netscape for the secure online</p>	<p style="text-align: center;">Developed by-</p> <p>The SET protocol was jointly developed by MasterCard and visa to secure web browsers for a bank</p>

S. No.	Secure Socket Layer	Secure Electronic Transaction
	transaction.	card transaction.
3.	Working- SSL uses a combination of public-key and symmetric-key encryption to safeguard data transactions. The handshake technique is used by the SSL protocol, which permits the server to verify its identity to the client. In case of unsuccessful authentication, the connection will not be formed.	Working- The dual signature mechanism is deployed by SET to safeguard a transaction. To use an e-commerce site, SET requires the purchase of software. The design of the protocol necessitates the client's installation of an e-wallet.
4.	Integrity- The technique of Hash functions is used for this purpose.	Integrity- The technique of digital signatures is used for this purpose.
5.	Acceptability- Its acceptability is more as compared to SET.	Acceptability- SET acceptability is less because it's necessary to build an open PKI.
6.	Functionality- The Secure Sockets Layer (SSL) is not a payment protocol. SSL encrypts the communication channel between the cardholder and the merchant website and is not backed by any financial institution. As a result, SSL is unable to ensure the security of a transaction.	Functionality- SET was created with the sole purpose of securing and ultimately guaranteeing a payment transaction. For example, increase in the possibilities for online retail growth only when consumer confidence grows in online shopping.
7.	Encryption- The purpose of SSL lies in prevention of data tampering in client/server applications and has considerably weaker encryption, with a maximum of 128-bit	Encryption- SET, which was created expressly to address the security of all parties involved in an electronic payment transaction, uses 1024-bit encryption throughout the

S. No.	Secure Socket Layer	Secure Electronic Transaction
	encryption.	transaction.
8.	Authentication- SSL certificates are not endorsed by any financial institution or payment brand association, so they cannot effectively validate all parties.	Authentication- Here, all parties get authentication to the transaction because SET's certificates are backed not just by a Certificate Authority, but also by financial institutions and MasterCard International.
9.	Security- SSL only protects the cardholder and the merchant, which is insufficient to prevent fraud. SSL transactions, in other words, are never assured.	Security- SET enables transaction security from the cardholder's desktop to the merchant via bank approvals and back through the gateway, leaving an indisputable audit trail and, as a result, a guaranteed transaction.

5.7] 3-D Secure protocol

What is 3D Secure?

3D Secure is a security protocol that is designed to provide an additional layer of security for online credit card and debit card transactions. This protocol was created by Arcot Systems (now CA Technologies) and was first used by Visa to provide improved security for Internet payments.

It's likely that you have heard of programs such as Verified by Visa, MasterCard Secure Code, J/Secure, and American Express Safe Key. These payment security services are based on 3D Secure.

How Does 3D Secure Work?

The name 3D comes from the three-domain model used to provide the additional layer of secure authentication between the financial authorization process and online authentication process. The three domains used to provide this security are:

- **Acquirer Domain:** The bank and the merchant receiving the transaction payment.

- **Issuer Domain:** The bank that issued the credit or debit card used for the transaction.
- **Interoperability Domain:** The infrastructure provided for the card that's used to support the 3D Secure protocol.

3D Secure uses XML messaging and SSL communication to secure and authenticate transactions.

As a merchant, it's important that you understand how 3D Secure impacts the customer transaction process.

1.
 1. After completing the checkout process, customers paying with Visa or MasterCard are prompted to enter their [Verified by Visa](#) or [MasterCard Secure Code](#)
 2. The customer is then either redirected to the issuer's website for authorization, or the authorization is completed within your payment solution.

If your customer is using an American Express, Discover card, or gift card, they are not prompted to enter a password.

If your customer is not enrolled in Verified by Visa or MasterCard Secure Code, the customer is prompted to enroll and create a password. The cardholder can decline registration, but when the customer reaches the maximum number of opt-out occurrences, the customer is no longer given the option to opt-out. The maximum number of opt-out occurrences is up to the credit card issuer, and the actions taken after the maximum is reached are also up to the credit card issuer.

It's important to note that as a merchant you can decide which transactions require 3D Secure authentication. For example, you might identify high-risk transactions with your payment solution's rules engine and then decide to further secure these transaction with 3D Secure.

What Are the Benefits of Using 3D Secure?

3D Secure offers merchants the following benefits:

1. **Liability shift:** As you know, when a chargeback occurs, you're typically liable for this transaction. With 3D Secure, the liability is shifted from you onto your issuing bank. Note: it's important you review all documentation provided by your issuer to ensure you understand the rules and regulations regarding 3D Secure implementation.
2. **Chargeback protection:** When you use Verified by Visa, you're guaranteed to never receive a chargeback. This protection helps secure you against friendly fraud or chargeback fraud. Note: MasterCard does not have a similar policy.
3. **Interchange benefits:** You may be able to access lower interchange fees and benefit from longer payment terms with your acquirer.
4. **Customer confidence:** As a CNP merchant, customer confidence is critical. Many customers will feel more confident knowing there is an extra level of security in place to protect their data.

What Should I Tell Customers about 3D Secure?

It's important to remember that while customers do want assurances that their transaction and data are secure, they also do not like extra confirmation steps and passwords. This means you must make sure you clearly explain that you are using 3D Secure (Verified by Visa or MasterCard Secure Code) to provide them with extra protection.

Provide some clear information in the early stages of the checkout process that explains the 3D Secure process in simple terms, to allow the customer to be prepared for the extra authentication step.

Remind the customer that this extra security is free of charge and that there aren't any additional fees hidden within the overall transaction. Provide a link that gives the customer more information about the program. Additionally, make sure the [Verified by Visa](#) or [MasterCard Secure Code](#) logo is clearly displayed.

How Can I Learn More about 3D Secure?

Read the documentation provided to you by your credit card issuer, and do not hesitate to ask them any questions you have.

Also, we suggest you browse the Verify website and learn more about [new device technology](#) and the future of [password authentication](#).

We welcome you to [contact us](#) with your questions about 3D Secure, and how to best ensure that your business and customers are protected from malicious and fraudulent activity.

5.8] Electronic Money

Electronic money is the currency that exists in banking computer systems that may be used to facilitate [electronic transactions](#). Although its value is backed by fiat currency and may be exchanged into a physical, tangible form, electronic money is primarily used for electronic transactions.

KEY TAKEAWAYS

- Electronic money is currency stored in banking computer systems.
- Electronic money is backed by fiat currency and the central banking system.
- Various companies allow for transactions with electronic money, such as Square or PayPal.

How eMoney Works

Electronic money is used for transactions globally and is commonly accessed through electronic banking systems and monitored through electronic processing. eMoney can be exchanged between individuals and businesses electronically using digital apps.

Electronic money is backed and controlled by [central banks](#). The [U.S. Federal Reserve](#) and 12 supporting banks manage the fiat currency in physical form and control the money supply through monetary policies and open market operations.

Central banks are exploring the use of a [Central Bank Digital Currency](#). In the United States, there is a physical currency issued by the Federal Reserve and digital balances held by commercial banks at the Federal Reserve. A CBDC would be a liability of the Federal Reserve, not of a commercial bank.¹

Electronic Payment Processing

Americans process transactions electronically by receiving pay checks through [direct deposits](#), moving money from one account to another via electronic fund transfers, or spending money with credit cards and debit cards.

The U.S. financial market has a robust infrastructure for transacting electronic money, which includes payment processing networks, such as Visa and MasterCard. Prepaid cards and digital wallets like [PayPal](#) and [Square](#) likewise allow users to deposit fiat currency for electronic money. Electronic money allows for immediate transactions through e-commerce.

eMoney vs. Cryptocurrency

Electronic money may be exchanged for fiat currency and is represented within the regulated banking system. [Cryptocurrencies](#) are privately issued and are not backed by financial assets. Cryptocurrency is created using block chain technology without government control or regulation.

What Is a Disadvantage of Using Electronic Money?

Fraud may be an issue when money can be transferred from one party to another without the necessity for verification of the original owner's true identity.

Is Cash Losing Popularity as the Use of eMoney Rises?

In 2022, only 18% of all payments were made with cash in the United States.²

What Is an Electronic Wallet or Digital Wallet?

A [digital wallet](#) or electronic wallet is a financial transaction application that runs on any connected device. It securely stores payment information and passwords in the cloud. Digital wallets may be accessible from a computer; [mobile wallets](#), which are a subset, are primarily used on mobile devices.

The Bottom Line

Electronic money or eMoney is currency stored in banking computer systems and backed by fiat currency. Individuals process and receive electronic money through paycheck direct deposits, electronic fund transfers, or online payments and purchases.

5.9] Email Security

E-mail Hacking

Email hacking can be done in any of the following ways:

- Spam
- Virus
- Phishing

#Spam

E-mail spamming is an act of sending **Unsolicited Bulk E-mails (UBE)** which one has not asked for. Email spams are the junk mails sent by commercial companies as an advertisement of their products and services.

#Virus

Some emails may incorporate with files containing malicious script which when run on your computer may lead to destroy your important data.

#Phishing

Email phishing is an activity of sending emails to a user claiming to be a legitimate enterprise. Its main purpose is to steal sensitive information such as usernames, passwords, and credit card details.

Such emails contains link to websites that are infected with malware and direct the user to enter details at a fake website whose look and feels are same to legitimate one.

E-mail Spamming and Junk Mails

Email spamming is an act of sending Unsolicited Bulk E-mails (UBI) which one has not asked for. Email spams are the junk mails sent by commercial companies as an advertisement of their products and services.

Spams may cause the following problems:

- It floods your e-mail account with unwanted e-mails, which may result in loss of important e-mails if inbox is full.
- Time and energy is wasted in reviewing and deleting junk emails or spams.
- It consumes the bandwidth that slows the speed with which mails are delivered.
- Some unsolicited email may contain virus that can cause harm to your computer.

Blocking Spams

Following ways will help you to reduce spams:

- While posting letters to newsgroups or mailing list, use a separate e-mail address than the one you used for your personal e-mails.
- Don't give your email address on the websites as it can easily be spammed.
- Avoid replying to emails which you have received from unknown persons.
- Never buy anything in response to a spam that advertises a product.

E-mail Clean-up and Archiving

In order to have light weighted Inbox, it's good to archive your inbox from time to time. Here I will discuss the steps to clean up and archive your Outlook inbox.

- Select **File** tab on the mail pane.
- Select **Cleanup Tools** button on account information screen.
- Select **Archive** from cleanup tools drop down menu.
- Select **Archive this folder and all subfolders** option and then click on the folder that you want to archive. Select the date from the **Archive items older than:** list. Click **Browse** to create new **.pst** file name and location. Click **OK**.

5.10] Wireless Application Protocol(WAP)

WAP is a collection of communication protocols meant to standardise how wireless devices, such as mobile phones and radio transceivers, may access the Internet and its services, such as email and newsgroups. Prior to the advent of WAP, Internet connection was available, but

various manufacturers utilised different technologies; WAP promised compatibility across these technologies.

WAP was created in 1997 at the WAP Forum by Ericsson, Motorola, Nokia, and Unwired Planet (now Phone.com). The WAP Forum was renamed the Open Mobile Alliance in 2002. (OMA).

Operation of the WAP:

WAP is an acronym for a collection of protocols. WAP hardware, such as WAP-enabled web browsers and network technologies, and WAP software, such as WAP-enabled web browsers, are designed to be compatible with this standard. These standards enhance mobile experiences that were previously limited by the capabilities of portable devices and wireless networks. WAP does this through the use of the following methods –

- Extensible Markup Language (XML), user datagram protocol (UDP), and Internet Protocol (IP) are efficient for wireless settings and are based on standards such as HTML, HTTP; the WML format for pages that may be sent via WAP.
- The WAP paradigm works in the same way as the traditional client-server model, but an additional WAP gateway acts as a middleman between the client and the server. This gateway transforms a micro browser's WAP device request into an HTTP URL request and sends it across the internet to the server. When the server answers, the WAP gateway analyses the response and sends the webpage as a WML file compatible with micro browsers to the WAP mobile device.

The Protocol Stack for WAP:

Here's a breakdown of the WAP standard's protocol stack that allows WAP devices to communicate with each other.

- In addition to WAE, WTP (Wireless Transaction Protocol) provides transaction functionality for server requests and answers.
- In order to protect privacy and data integrity, WTLS uses public-key cryptography.
- *Wireless Datagram Protocol* (WDP)– This protocol describes how data travels from the receiver to the sender and vice versa.

Why Would You Want to Use WAP?

WAP offered benefits to wireless network operators, content providers, and end-users when it was first presented in 1999.

- WAP was intended to improve existing wireless data services like voicemail while also allowing developers to create new mobile apps. These applications may be set up without the need for extra infrastructure or phone updates.
- WAP provides a market for third-party app developers to create new apps and mobile phone features. WML application development is introduced as an alternative programming language that allows developers to build effective mobile device apps.

5.11] Security In GSM

GSM is the most secured cellular telecommunications system available today. GSM has its security methods standardized. GSM maintains end-to-end security by retaining the confidentiality of calls and anonymity of the GSM subscriber.

Temporary identification numbers are assigned to the subscriber's number to maintain the privacy of the user. The privacy of the communication is maintained by applying encryption algorithms and frequency hopping that can be enabled using digital systems and signalling.

This chapter gives an outline of the security measures implemented for GSM subscribers.

Mobile Station Authentication

The GSM network authenticates the identity of the subscriber through the use of a challenge-response mechanism. A 128-bit Random Number (RAND) is sent to the MS. The MS computes the 32-bit Signed Response (SRES) based on the encryption of the RAND with the authentication algorithm (A3) using the individual subscriber authentication key (Ki). Upon receiving the SRES from the subscriber, the GSM network repeats the calculation to verify the identity of the subscriber.

The individual subscriber authentication key (Ki) is never transmitted over the radio channel, as it is present in the subscriber's SIM, as well as the AUC, HLR, and VLR databases. If the received SRES agrees with the calculated value, the MS has been successfully authenticated and may continue. If the values do not match, the connection is terminated and an authentication failure is indicated to the MS.

The calculation of the signed response is processed within the SIM. It provides enhanced security, as confidential subscriber information such as the IMSI or the individual subscriber authentication key (Ki) is never released from the SIM during the authentication process.

Signalling and Data Confidentiality

The SIM contains the ciphering key generating algorithm (A8) that is used to produce the 64-bit ciphering key (Kc). This key is computed by applying the same random number (RAND) used in the authentication process to ciphering key generating algorithm (A8) with the individual subscriber authentication key (Ki).

GSM provides an additional level of security by having a way to change the ciphering key, making the system more resistant to eavesdropping. The ciphering key may be changed at regular intervals as required. As in case of the authentication process, the computation of the ciphering key (Kc) takes place internally within the SIM. Therefore, sensitive information such as the individual subscriber authentication key (Ki) is never revealed by the SIM.

Encrypted voice and data communications between the MS and the network is accomplished by using the ciphering algorithm A5. Encrypted communication is initiated by a ciphering mode request command from the GSM network. Upon receipt of this command, the mobile station begins encryption and decryption of data using the ciphering algorithm (A5) and the ciphering key (Kc).

Subscriber Identity Confidentiality

To ensure subscriber identity confidentiality, the Temporary Mobile Subscriber Identity (TMSI) is used. Once the authentication and encryption procedures are done, the TMSI is sent to the mobile station. After the receipt, the mobile station responds. The TMSI is valid in the location area in which it was issued. For communications outside the location area, the Location Area Identification (LAI) is necessary in addition to the TMSI.

5.11] Security In 3G :

Mobile 3G network attacks are attacks that are specifically targeted at 3G mobile networks. 3G, or third-generation, mobile networks are the third generation of mobile communication technology and are used to provide wireless data and voice services to mobile devices. There are several types of attacks that can be launched against 3G mobile networks, including eavesdropping, man-in-the-middle attacks, denial of service (DoS) attacks, malware infections, and interference with communication signals. It's important for mobile network operators to implement security measures to protect against these types of attacks and keep their networks secure.

Types of 3G Mobile Attack

There are several types of attacks that can be launched against mobile networks –

- **Eavesdropping** – This type of attack involves intercepting and listening to the communication between two parties without their knowledge.
- **Man-in-the-middle attack** – In this type of attack, an attacker intercepts the communication between two parties and poses as one of the parties, allowing them to alter the communication and potentially steal sensitive information.
- **Denial of service (DoS) attack** – A DoS attack involves flooding a network or device with traffic, making it unavailable to legitimate users.
- **Malware** – Malware, or malicious software, can be used to compromise the security of a mobile network and steal sensitive information.
- **Interference** – Attackers can interfere with the communication between two parties by disrupting the signals used to transmit data.
- **Rogue base stations** – An attacker can set up a fake base station to intercept communication between mobile devices and the legitimate network.
- **SMS spoofing** – Attackers can send text messages from a fake or spoofed sender, potentially tricking users into divulging sensitive information or downloading malware.
- **Location tracking** – An attacker can track a mobile device's location by accessing the location data transmitted by the device.

It's important for mobile network operators to implement security measures to protect against these types of attacks and keep their networks secure.

How to secure Mobile 3G network traffic

There are several measures that can be taken to secure 3G mobile network traffic –

- **Encryption** – Encrypting the communication between devices and the network can help prevent eavesdropping and man-in-the-middle attacks.
- **Virtual private networks (VPNs)** – VPNs can be used to secure the communication between devices and the network by creating a secure, encrypted connection.
- **Firewalls** – Firewalls can be used to block unwanted traffic and protect against DoS attacks.
- **Antivirus software** – Antivirus software can be used to detect and prevent malware infections on mobile devices.
- **Network segmentation** – Segmenting the network into different sections can help prevent an attacker from gaining access to the entire network if they are able to compromise one section.
- **Regular software updates** – Keeping software up to date can help fix known vulnerabilities and prevent attacks.

- **User education** – Educating users on how to identify and avoid potential threats, such as phishing attacks, can help prevent them from inadvertently compromising the security of the network.

How to protect mobile data from network attacks

Here are some steps you can take to protect your mobile data from network attacks –

- Use a strong, unique password for your device and don't share it with anyone.
- Enable two-factor authentication whenever possible.
- Use a virtual private network (VPN) to encrypt your internet connection and secure your data when using public Wi-Fi.
- Keep your device's operating system and antivirus software up to date to fix known vulnerabilities.
- Avoid accessing sensitive information on public or untrusted networks.
- Be cautious when clicking on links or downloading attachments, as they could contain malware.
- Don't jailbreak or root your device, as doing so can remove important security measures.
- Consider using a mobile security app to provide additional protection for your device and data.

Unit 6 - User Authentication And kerberos

6.1] Introduction

User authentication verifies the identity of a user attempting to gain access to a network or computing resource by authorizing a human-to-machine transfer of credentials during interactions on a network to confirm a user's authenticity. The term contrasts with [machine authentication](#), which is an automated authentication method that does not require user input.

[Authentication](#) helps ensure only authorized users can gain access to a system by preventing unauthorized users from gaining access and potentially damaging systems, stealing information or causing other problems. Almost all human-to-computer interactions -- other than guest and automatically logged-in accounts -- perform a user authentication. It authorizes access on both wired and wireless networks to enable access to networked and internet-connected systems and resources.

A straightforward process, user authentication consists of three tasks:

1. **Identification.** Users have to prove who they are.
2. **Authentication.** Users have to prove they are who they say they are.
3. **Authorization.** Users have to prove they're allowed to do what they are trying to do.

User [authentication](#) can be as simple as requiring a user to type a unique identifier, such as a user ID, along with a [password](#) to access a system. It can also be more complex, however -- for example, requiring a user to provide information about physical objects or the environment or even take actions, such as placing a finger on a fingerprint reader.

User authentication methods

The main factors used in user authentication include the following:

- **Knowledge factors** include all things users must know in order to log in to gain access to a system. Usernames, IDs, passwords and personal identification numbers (PINs) all fall under this category.
- **Possession factors** consist of anything users must have in their possession in order to log in. This category includes one-time password tokens, [key fobs](#), [smartphone](#) apps, and employee ID cards.
- **Inherence factors** include characteristics inherent to individuals that confirm their identity. This category includes the scope of biometrics, such as [retina scans](#), fingerprint scans, [facial recognition](#) and Voice authentication.

❖ Kerberos :

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. In Kerberos Authentication server and database is used for client authentication. Kerberos runs as a third-party trusted server known as the Key Distribution Centre (KDC). Each user and service on the network is a principal. The main components of Kerberos are:

- **Authentication Server (AS):**
The Authentication Server performs the initial authentication and ticket for Ticket Granting Service.
- **Database:**
The Authentication Server verifies the access rights of users in the database.
- **Ticket Granting Server (TGS):**
The Ticket Granting Server issues the ticket for the Server

Kerberos Overview: Diag

- **Step-1:**
User login and request services on the host. Thus user requests for ticket-granting service.
- **Step-2:**
Authentication Server verifies user's access right using database and then gives ticket-granting-ticket and session key. Results are encrypted using the Password of the user.
- **Step-3:**
The decryption of the message is done using the password then send the ticket to Ticket Granting Server. The Ticket contains authenticators like user names and network

addresses.

- **Step-4:**
Ticket Granting Server decrypts the ticket sent by User and authenticator verifies the request then creates the ticket for requesting services from the Server.
- **Step-5:**
The user sends the Ticket and Authenticator to the Server.
- **Step-6:**
The server verifies the Ticket and authenticators then generate access to the service. After this User can access the services.

Kerberos Limitations

- Each network service must be modified individually for use with Kerberos
- It doesn't work well in a timeshare environment
- Secured Kerberos Server
- Requires an always-on Kerberos server
- Stores all passwords are encrypted with a single key
- Assumes workstations are secure
- May result in cascading loss of trust.
- Scalability

Is Kerberos Infallible?

No security measure is 100% impregnable, and Kerberos is no exception. Because it's been around for so long, hackers have had the ability over the years to find ways around it, typically through forging tickets, repeated attempts at password guessing (brute force/credential stuffing), and the use of malware, to downgrade the encryption.

Despite this, Kerberos remains the best access security protocol available today. The protocol is flexible enough to employ stronger encryption algorithms to combat new threats, and if users employ good password-choice guidelines, you shouldn't have a problem!

What is Kerberos Used For?

Although Kerberos can be found everywhere in the digital world, it is commonly used in secure systems that rely on robust authentication and auditing capabilities. Kerberos is used for Posix, Active Directory, NFS, and Samba authentication. It is also an alternative authentication system to SSH, POP, and SMTP.

Applications

- **User Authentication:** User Authentication is one of the main applications of Kerberos. Users only have to input their username and password once with Kerberos to gain

access to the network. The Kerberos server subsequently receives the encrypted authentication data and issues a ticket granting ticket (TGT).

- **Single Sign-On (SSO):** Kerberos offers a Single Sign-On (SSO) solution that enables users to log in once to access a variety of network resources. A user can access any network resource they have been authorized to use after being authenticated by the Kerberos server without having to provide their credentials again.
- **Mutual Authentication:** Before any data is transferred, Kerberos uses a mutual authentication technique to make sure that both the client and server are authenticated. Using a shared secret key that is securely kept on both the client and server, this is accomplished. A client asks the Kerberos server for a service ticket whenever it tries to access a network resource. The client must use its shared secret key to decrypt the challenge that the Kerberos server sends via encryption. If the decryption is successful, the client responds to the server with evidence of its identity.
- **Authorization:** Kerberos also offers a system for authorization in addition to authentication. After being authenticated, a user can submit service tickets for certain network resources. Users can access just the resources they have been given permission to use thanks to information about their privileges and permissions contained in the service tickets.
- **Network Security:** Kerberos offers a central authentication server that can regulate user credentials and access restrictions, which helps to ensure network security. In order to prevent unwanted access to sensitive data and resources, this server may authenticate users before granting them access to network resources.

6.2] Password Authentication Networks

A password is a secret word or phrase or code that you need to know in order to have access to a place or system. In technical terms, it is a series of letters or numbers that you must type into a computer or computer system in order to be able to use it. A password is a real-life implementation of challenge-response authentication (a set of protocols to protect digital assets and data).

A string of characters i.e letters, numbers, special characters, used to verify the identity of a user during the authentication process is known as password.

Password Management:

Since passwords are meant to keep the files and data secret and safe so it is prevented the unauthorized access, password management refers to the practices and set of rules or principles or standards that out must follow or at least try to seek help from in order to be a good/strong password and along with its storage and management for the future requirements.

Issues Related to Managing Passwords:

The main problem with password management is that it is not safe to use the same password for multiple sites, therefore having different passwords for different sites and on top of that remembering them is quite difficult. As per the statistics, more

than 65% of people reuse passwords across accounts and the majority do not change them, even after a known breach. Meanwhile, 25% reset their passwords once a month or more because they forgot them.

To escape from this situation people often tend to use password managers (A password manager is a computer program that allows users to store, generate, and manage their passwords for local applications and online services.). Password managers to a certain extent reduce the problem by having to remember only one “master password” instead of having to remember multiple passwords. The only problem with having a master password is that once it is out or known to an attacker, the rest of all the passwords become available.

The main issues related to managing passwords are as follows:

- Login spoofing
- Sniffing attack
- [Brute force attack](#)
- [Shoulder surfing attack](#)
- Data breach

Methods to Manage Password:

There are a lot of good practices that we can follow to generate a strong password and also the ways to manage them.

- **Strong and long passwords:** A minimum length of 8 to 12 characters long, also it should contain at least three different character sets (e.g., uppercase characters, lowercase characters, numbers, or symbols)
- **Password Encryption:** Using irreversible end-to-end encryption is recommended. In this way, the password remains safe even if it ends up in the hands of cybercriminals.
- **Multi-factor Authentication (MFA):** Adding some security questions and a phone number that would be used to confirm that it is indeed you who is trying to log in will enhance the security of your password.
- **Make the password pass the test:** Yes, put your password through some testing tools that you might find online in order to ensure that it falls under the strong and safe password category.
- **Avoid updating passwords frequently:** Though it is advised or even made mandatory to update or change your password as frequently as in 60 or 90 days.

6.3] Authentication Tokens :

Password is the most common way of authentication. However, they are not as cheap and simple as we think, especially from the organization’s point of view because they have to provide an id and password to each resource they use as well as they’ve to manage the passwords of so many people using their portal. Also, it becomes cumbersome for the users to remember the passwords of every website

they log into. So we have an alternative to the passwords and that is Authentication Tokens.

Authentication Token is an immensely useful alternative to passwords. It is a small device that generates a new random value every time it is used. This random value is used for authenticating the users.

Components of Authentication Token

- Processor
- LCD for displaying output (Liquid Crystal Display)
- Battery
- A small keypad for entering information
- A real-time clock

Every such device(authentication token) is pre-programmed with a unique number called **random seed or seed**. This seed ensures that the output generated by the authentication token (the device) is unique. An authentication token is an example of 2-factor authentication because the token itself is protected with some PIN.

Working of Authentication Token

1. Creation of Token

When an authentication token is created, the corresponding random seed is generated for the token by the authentication server. This seed is automatically used by the authentication token due to which the value of the seed is not known by the user. This seed is pre-programmed inside the token, as well as its entry is made against that user's record in the user database.

2. Use of Token

An authentication token automatically generates pseudo-random numbers, called one-time passwords or one-time passcode (these codes/passwords can be used only once). Once they are used, they cannot be reused. This one time password is basically a 4 digit PIN. Below are some important points to use this one time password.

1. The user being authenticated will enter his/her id and one-time password which goes to the server.
2. The server receives the seed corresponding to the user id from the user database, using a *seed-retrieval program*.
3. The server gives the seed and one-time passcode to a *Password Validation Program*.
4. This program checks if the one-time password and seed are related to each other.

3. Server responds

The server finally responds back with a suitable message based on the output (success/failure) of the previous step.

Types of Authentication Token:

Response/Challenge Token

1. User sends a login request by providing only his user id and not the one-time password.
2. Server checks whether the user id is valid. If it is not valid, it responds with an error message otherwise if it is valid, then the server creates a random challenge. Then sends the random challenge to the user.
3. User receives the random challenge. Open the authentication token using the PIN and keys in the random challenge using the small keypad.
4. The seed of the token encrypts the random challenge which is then entered by the user in the password section of the login request.
5. Server verifies the encrypted random challenge received by the user which can be done in two ways-
 - The server can decrypt the encrypted random challenge received from the user with the seed value for the user, which is available to the server via the user database. If this decryption matches the original random challenge available on the server, the authentication is successful.
 - The server can encrypt its own version of the random challenge, which was sent earlier to the user, with the seed for the user. If this encryption matches with the encrypted random challenge received from the user, the authentication is successful.

Time-Based Token

In a time-based token, the server need not send any random challenge to the user. The token need not have a keypad for entry. In fact, it uses time in place of a random challenge. The tokens automatically generate a password every 60 seconds and display the latest password on the LCD output for the user.

For generating the password, the time-based tokens use seed and current system time.

- When a user wants to log in he/she enters the password displayed on the LCD of the token and uses it to login along with their user id.
- The server receives the password and performs an independent cryptographic function on the user's seed value and the current system time to generate its version of the password. If the two values match, it considers the user as a valid one.

- Finally, the server sends an appropriate message back to the user based on the result of the previous step.

6.4] Certificate Based Authentication

- Certificate-based authentication (CBA) has been a staple of government agencies and other high security environments for decades, long before the invention of [FIDO U2F](#) and [FIDO2](#), mostly due to its reliability and effectiveness in physical environments.
- To this day, it remains a favourite amongst many security experts, and is still being deployed across countless industries for a variety of scenarios.
- A couple of notable examples of its ubiquity include the humble smart card often used for access to offices and other buildings, the widely used SSL/TLS protocol featured in web browsers, and CBA is even found in any implementation of Public-Key Infrastructure (PKI).
- Generally speaking, client certificate-based authentication refers to an end user's device proving its own identity by providing a digital certificate that can be verified by a server in order to gain access to a network or other resources.
- In a nutshell, certificate-based authentication (CBA) uses a digital certificate derived from cryptography to identify a user, device or machine, before granting access to an application, network or other resource. Unlike some authentication solutions that are targeted at humans, such as one time passwords (OTP) and biometrics, certificate-based authentication can be adopted for all endpoints, including servers, personal computers, e-passports and literally anything that may be classified under the Internet of Things (IoT).
- In general, client certificate-based authentication and other methods where the secret is never exposed to even the user, is preferable to password-based authentication. Username and password authentication is based only on what the user knows (the password), but certificate-based client authentication also leverages what the user has (the private key), which cannot be phished, guessed or socially engineered.

But it is also important to highlight some of the conditions that help to maintain this level of control:

- no unauthorized users have gained access to the private-key underlying the digital certificate
- the lifecycle of distributed certificates is properly managed, including registration, renewal and revocation
- the proper infrastructure is set up to support the sending and validation of certificates

Users must be especially vigilant in protecting private keys in addition to the physical security of their devices generally speaking, but parties responsible for the infrastructure to support CBA also have their part to play in the process.

How Does Certificate-Based Authentication Work?

A digital identity certificate is an electronic document used to prove private key ownership. Certificate-based authentication uses the information within said document to verify the user, device or machine, in contrast to the classic username and password combination which is strictly limited to verifying only those who are in possession, i.e. potentially not just the user who *should* have access. The increased sophistication of attacks, coupled with the exploding number of attack surfaces as more and more devices access the network, make this additional capability more necessary than ever.

A digital certificate contains a number of important details, depending on the standard in use. For example, the popular X.509 certificate includes the below elements:

- The public key
- The user or device's name
- The name of the Certificate Authority (CA) that issued the certificate
- The date from which the certificate is valid
- The expiry date of the certificate
- The version number of the certificate data
- A serial number

As shown above here are the relevant details of the certificate based authentication flow:

- Client will initiate connection to the server.
- The server will respond and provide the server's public certificate to the client.
- The client will perform some validation to make sure the server's public certificate is trusted.
- The server requests the certificate from the client.
- A nonce is signed by the client using the client's private key, and is returned to the server and also includes the client's public certificate.
- The server receives both the certificate and the signed nonce. Using the client's public certificate the server will verify that the nonce was signed by the client.
- The server checks that the certificate has not expired and that it also has not been revoked.
- If all the verifications have passed the server may map attributes of the certificate to a user in their system to identify and authenticate the user.

There are usually even additional layers of verification in addition to the above process, such as examining the signature of the CA that issued the certificate, and climbing that hierarchy of signatures all the way to the root CA, to verify that it is known and trusted by the server.

6.5] Biometric Authentication

Biometric authentication refers to a cyber security process that verifies a user's identity using their unique biological traits such as fingerprints, voices, retinas, and facial features. Biometric authentication systems store this information in order to verify a user's identity when that user accesses their account. This type of authentication is usually more secure than traditional forms of multi-factor authentication.

- **Types of Authentication Methods**

Biometric authentication refers to a cyber security process that verifies a user's identity using their unique biological traits such as fingerprints, voices, retinas, and facial features. Biometric authentication systems store this information in order to verify a user's identity when that user accesses their account. This type of authentication is usually more secure than traditional forms of multi-factor authentication.

Types of Authentication Methods

The following are a few common authentication methods used for network security designed to beat cybercriminals and some of the biometric authentication technologies below are ones that you might use daily.

Facial recognition: These systems use a person's unique facial features to identify them. It's used in a variety of places such as smartphones, credit card payments, and law enforcement.

Fingerprint Recognition: Fingerprint authentication uses a person's unique fingerprint to verify their identity. It can be used to secure everything from mobile devices to automobiles, even buildings, making it the most widespread biometric authentication technology.

Eye Recognition: Eye recognition uses the unique pattern of someone's iris or retina to identify them. Because this type of biometric authentication is harder to implement, it's less common than the other types of biometric authentication options. An iris scan requires an infrared light source, a camera that can see IR, and minimal light pollution in order to ensure accuracy. Although it poses its challenges, it is one of the most accurate biometric authentication systems available when those conditions are met. Eye recognition is generally used in situations where security is most critical such as nuclear research facilities, etc.

Voice Recognition: Voice recognition uses the tone, pitch, and frequencies that are unique to an individual to authenticate them. This is the most commonly used biometric to verify users when they contact a call center for customer service support (for example, online banking)

Retina/Iris Recognition: Retina or also known as iris recognition, uses the pattern of someone's iris or retina to identify them. This type of biometric authentication is less common as it is harder to implement. It requires the implementation of an infrared light source, a camera that can see IR, and minimal light pollution to ensure accuracy. However, it happens to be one of the most accurate biometric authentication methods when those conditions are met. So it's typically used in situations where security is most critical (nuclear research facilities, for instance).

Gait Recognition: Gait recognition authenticates using the way someone walks to identify them. Each person walks a little differently, so the way a person puts one foot in front of the other is an effective way to verify their identity. As of now, it's not a common form of authentication but it's expected to become more common as future forms of authentication become more popular.

Vein Recognition: Vein recognition uses the pattern of blood vessels in a person's hand or finger to identify them. This type of biometric authentication uses infrared light to map the veins under the skin in your hands or fingers. Vein recognition is extremely accurate, more than retina/iris recognition.

What is Multimodal Biometric Authentication ?

First, we need to understand what a unimodal biometric authentication system is. Essentially it's a system that verifies only one distinct characteristic, e.g face, retina. However, this system is very susceptible to spoofing.

This is where multimodal biometric authentication comes into play. It's an approach in which various biometrics are checked during identity verification. This makes it harder for a malicious hacker to spoof.

An example of multimodal authentication: A hacker may be able to find a person's photo on the internet, which they then use to successfully trick a facial recognition system into thinking it's the actual user. If the system just had the one authentication, then the user's accounts would be hacked. However, if the system requires the user to provide additional authentication such as a video of the person saying their password, then the hacker is very unlikely to find it.

By combining physical and behavioral authentication, you can enhance your security posture. Even if a malicious actor manages to spoof a fingerprint, the system can detect change in behaviour and deny entry. E.g., their speed of interaction with the system may be slower than the real user, or they are using keyboard shortcuts that the real user never used.

- **The Benefits and Risks of Biometric Authentication**

Identity insurance: Biometric identification provides the answers to “something a person has and is” and helps verify identity. Biometric authentication ensures increased levels of assurance to end-users. Its sophisticated software lets providers know that a person is who they claim to be through a tangible, real-world trait. Even if a cyber attacker knew of a user’s password or the answer to their security question, there’s no way they’d be able to duplicate a fingerprint or iris scan.

Ease of Use: While biometric authentication is more on the technical side in regards to its internal process, it’s generally easy and quick from a user’s point of view. By using either a fingerprint scanner to unlock an account or facial recognition, you’re cutting down the amount of times you have to log in using a long password that has multiple special characters in it that you will likely end up forgetting. Apple does a great job with biometric authentication, both fingerprint and facial, in their devices.

Fraud Detection: Biometrics are almost nearly impossible to replicate. They’re hard to replicate and steal, and only have about 1 in 64 billion chances that your fingerprint will match up exactly with someone else’s. It’s highly unlikely that a hacker will be able to access anything that’s secured with biometrics.

Risks

Being Hackable: Biometrics can still be hacked. Businesses and governments that collect and store users’ personal data are under constant threat from hackers. If however, they are a victim of a data breach, biometric data is irreplaceable, and organizations need to treat users’ biometrics with care and caution.

Partial Matches: Most common biometric authentication methods rely on partial information to authenticate a user’s identity. For example, during the enrollment process of registering your fingerprint, it will take data from your entire print and convert it into data. During future authentication however, it will only take partial fingerprint data to verify your identity so it’s faster and quicker.

Fail to Recognize a Valid User: When you register for facial recognition, you are registering a specific angle, and expression to your face. However, because the system only has the data from the enrollment process, anytime a user wears glasses, makeup, or even smiling, the facial recognition has a hard time recognizing the user, which could make the login process difficult.

Bias: Facial recognition systems may not recognize persons of color or non-cisgender people as accurately. Many biometric systems have been trained primarily using white or white male photos. This incorporates in them an inherent bias that results in difficulty recognizing women and people of color. Poor implementation of technology or deliberate misuse can result in discrimination and exclusion. Without a proven identity proofing solution, cross-demographic performance can be unreliable.

Fears of Sharing Biometric Data: Is it acceptable for companies to sell or provide their biometric data to others, such as law enforcement, immigration enforcement, or repressive foreign governments. These privacy concerns have caused many US states to enact biometric information privacy laws. When biometrics are converted into data and stored, particularly in places or countries that have large surveillance measures, a user runs the risk of leaving a permanent digital record that can be potentially tracked by nefarious actors.

Data Storage: Wherever biometric data is stored, it must be stored securely. Biometric data cannot be reset like a password. If biometric data is hacked, then there's really nothing a user can do as they cannot change their fingerprint or iris.

6.6] Key Distribution center

A Key Distribution Center (KDC) is a crucial element in a network encryption system responsible for assigning secret keys to authorized users. It operates in a system such as Kerberos, providing authentication and secret keys for users and services. Its primary function is to [enhance](#) communication security by preventing unauthorized access to transmitted data.

Key Distribution Center (KDC) is a crucial part of many cryptographic systems that provide a mechanism for secure distribution of encryption keys. It serves as a trusted third party that authenticates users in a network.

KDC works in several steps. Primarily, it verifies the user's identity, then it creates session keys and securely transmits these keys to the communicating parties. This process ensures secure, encrypted communication. The most common example of a system that uses KDC is the Kerberos protocol. Kerberos is a network authentication protocol designed to provide strong authentication for client/server applications.

Importance

The Key Distribution Center (KDC) is a crucial component in cryptography, particularly within the context of network [security systems](#) that use protocols like Kerberos. It is responsible for generating, storing, and managing secret encryption keys, which are used for secure communication between network entities.

Without a functioning KDC, these entities can't securely exchange their [data](#) since they wouldn't have access to shared, mutually trustworthy keys. Therefore, the KDC provides an essential service by distributing these keys securely, ensuring that each party involved in a communication process can trust and verify the authenticity and

confidentiality of the transmitted data. Its existence, operation, and proper management significantly enhance the security integrity of a network or system.

Explanation

The Key Distribution Center (KDC) is a critical feature of network security systems, particularly those that employ symmetric key encryption such as the Kerberos protocol. Its primary function is to securely distribute encryption and decryption keys to users within a network, thereby facilitating secure communication and data transfers. This is essential as encryption keys need to be dynamically assigned and updated to members of a network to maintain secure transmissions.

Therefore, a KDC helps in reducing the vulnerability of a system to key theft or unauthorized access. To further illuminate on its use, when a user requests access to a particular system or service within the network, the KDC is responsible for verifying their identity, and if authenticated, providing the necessary keys to enable the user to securely engage with the requested service.

This process ensures that only authorized individuals gain access to relevant information and resources, and any data exchanged remains confidential and protected from security threats. Hence, a Key Distribution Center is at the heart of managing and securing digital identities within a network.

Examples

1. Kerberos Authentication System: Kerberos, originally developed at MIT, is a widely used example of Key Distribution Center (KDC) technology. It is used to verify the identity of users and network services. In this system, the KDC is the trusted third party that distributes symmetric encryption keys to users and service providers in a network. After initial authentication, Kerberos issues 'tickets' for subsequent access to different network resources, minimizing repeated exposure of user credentials.

2. Wi-Fi Protected Access 2 (WPA2): In the context of a Wi-Fi network, WPA2 uses a central Key Distribution Center mechanism called the "Authentication Server", which distributes cryptographic keys to other devices on the network. WPA2's protocol relies on the KDC to authenticate devices and keep the network secure from unauthorized access.

3. Microsoft Active Directory: Active Directory (AD), Microsoft's network management technology, incorporates a KDC as part of the system's security infrastructure. In an Active Directory framework, the KDC distributes encryption keys to users, computers, and services within the network, securing the transmission of sensitive data and maintaining the integrity of network communications.

Frequently Asked Questions (FAQ)

Q: What is a Key Distribution Center (KDC)?

A: A Key Distribution Center is a part of a cryptographic system intended to reduce the risks inherent in exchanging keys. KDCs are part of the Kerberos protocol for network authentication.

Q: How does a Key Distribution Center work?

A: A Key Distribution Center securely holds encryption keys and controls their distribution among parties involved in communication. When a user logs in, the KDC verifies their identity and issues a ticket-granting ticket (TGT), using which the user can obtain service tickets for various services.

Q: What is the main purpose of a KDC?

A: The main purpose of a Key Distribution Center is to provide a secure method of sharing cryptographic keys, reducing the risk of keys being intercepted and potentially misused.

Q: When would you use a Key Distribution Center?

A: KDCs are often used in large networks where secure communication is required. For example, it is a critical part of the Kerberos authentication protocol, commonly used in Windows Active Directory Networks.

Q: Is Key Distribution Center safe?

A: As long as the KDC server is secure, and the system is properly configured and managed, KDC offers a high level of safety. However, if an unauthorized person gains access to the KDC, it can be compromised.

Q: What is the relationship between a KDC and Kerberos?

A: Kerberos is a network authentication protocol and KDC is an integral part of its system responsible for authenticating users. The KDC issues incident-specific cryptographic keys when a logon session is initiated, allowing secure authentication throughout the session.

Q: What information is stored in a KDC?

A: A KDC holds a database of secret keys; each entity on the network — whether a user or a service — shares a unique secret key with the KDC.

6.7] Security Handshake Pitfalls :

Security handshake protocols in cryptography are crucial for establishing a secure communication channel between two parties. However, there are potential pitfalls and vulnerabilities that can be exploited by attackers. Here are some common security handshake pitfalls in cryptography:

1. **Man-in-the-Middle Attacks:**

- Attackers may intercept the handshake process, pretending to be one of the communicating parties. This can lead to the exchange of encryption keys with the attacker instead of the intended recipient.

2. **Weak Key Exchange Algorithms:**

- If the key exchange algorithm used during the handshake is weak or vulnerable, it can be exploited by attackers to compromise the confidentiality of the communication.

3. **Insufficient Authentication:**

- Weak or inadequate authentication mechanisms during the handshake can allow attackers to impersonate one of the parties, leading to unauthorized access or data manipulation.

4. **Lack of Forward Secrecy:**

- Without forward secrecy, if a long-term key is compromised, all past communications encrypted with that key become vulnerable to decryption. Implementing forward secrecy helps mitigate this risk.

5. **Insecure Cipher Suites:**

- Choosing insecure or out dated cipher suites during the handshake can expose the communication to attacks. It's important to use strong, well-vetted encryption algorithms and avoid deprecated protocols.

6. **Certificate Issues:**

- If certificates are not properly validated during the handshake, it opens the door to various attacks. This includes issues such as expired certificates, incorrect certificate chains, or certificates signed by untrusted entities.

7. **Timing Attacks:**

- Timing attacks involve exploiting variations in the time taken to perform cryptographic operations. Poorly implemented algorithms or protocols may leak information through timing, compromising the security of the handshake.

8. **Denial of Service (DoS) Attacks:**

- Handshake processes can be targeted for DoS attacks. Attackers may flood the system with handshake requests, overwhelming resources and causing service disruptions for legitimate users.

9. **Insecure Random Number Generation:**

- Secure cryptographic protocols rely on strong random numbers for key generation. If the random number generator is weak or predictable, it can lead to the generation of weak keys, compromising the security of the handshake.

10. **Implementation Flaws:**

- Bugs or vulnerabilities in the implementation of cryptographic protocols can introduce weaknesses in the handshake process. Regular code audits and security testing are essential to identify and address these issues.

6.8] Single Sign on (SSO)

. Role of SSO :

The role of Single Sign-On (SSO) is to provide a seamless and convenient authentication experience for users while improving security and reducing administrative overhead for organizations.

- **Enhanced user experience:** Users can access multiple applications and systems with a single set of credentials, simplifying the login process and improving productivity.
- **Improved security:** SSO reduces the reliance on weak passwords and enables centralized access control, enforcing strong security measures consistently across systems.
- **Centralized authentication and access control:** SSO centralizes the authentication process, allowing organizations to manage user access and authentication policies in one place
- **Streamlined user provisioning and deprovisioning:** SSO simplifies the management of user access privileges, making it easier to on board and off board users.
- **Audit and compliance:** SSO provides better visibility and control over user access, facilitating monitoring, auditing, and compliance with regulatory requirements.

Script-Based SSO:

In script-based SSO, authentication and authorization are managed through scripts or code embedded within applications.

- Users authenticate themselves individually to each application, and the script within each application verifies the user's credentials.
- The script then establishes a session for the user, allowing them to access other applications without requiring further authentication.
- Example: An organization uses script-based SSO to enable its employees to access different internal systems such as email, document management, and HR portals. Once authenticated to one system, the user can seamlessly access other systems without entering credentials again.

Agent-Based SSO:

- Agent-based SSO relies on a dedicated software agent installed on the user's device or workstation.
- The agent intercepts authentication requests from applications and handles the authentication process on behalf of the user.
- Upon successful authentication, the agent provides a security token to the application, allowing the user to access multiple applications without additional login prompts.
- Example: A user logs into their computer and launches a web browser. The agent installed on the computer automatically authenticates the user to the browser, and subsequent requests to various web applications are handled by the agent, providing a seamless single sign-on experience.

Unit 3 : Assymetric Key Algorithm, Digital Signatures And RSA

3.1] Introduction

Assymmetric Key Algorithm :

Unlike symmetric encryption methods, asymmetric encryption methods use two separate keys for the encryption and decryption process. Asymmetric encryption is also referred to as public key encryption. A public key is used during the encryption process to encrypt the plaintext message and a private key is used during the decryption process to decrypt ciphertext.

Advantages	Disadvantages
Easier to deal with key distribution and ensure integrity of message	Slower due to multiple keys

If Alice wants to send a message that only Bob can read, Alice will use Bob's public key to encrypt the message. The only way the message can be decrypted is when Bob uses his private key.

Figure 2 – Asymmetric encryption

The most popular asymmetric algorithm is RSA which can provide key exchange, encryption, and digital signatures. The strength of this algorithm lies in the difficulty of calculating the prime factors of very large numbers. Other asymmetric algorithms include ElGamal, ECC, and Knapsack. Read the article, Asymmetric Encryption Algorithms, Diffie-Hellman, RSA, ECC, ElGamal, DSA, for a comparison of these algorithms.

❖ Digital Signature :

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is

very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

3.2] History And Overview Of Assymmetric Key Cryptography :

Asymmetric cryptography is a second form of cryptography. It is called a Public-key cryptography. There are two different keys including one key is used for encryption and only the other corresponding key should be used for decryption. There is no other key can decrypt the message and not even the initial key used for encryption. The style of the design is that every communicating party needs only a key pair for communicating with any number of other communicating parties.

Asymmetric cryptography is scalable for use in high and ever expanding environments where data are generally exchanged between different communication partners. Asymmetric cryptography is used to exchange the secret key to prepare for using symmetric cryptography to encrypt information.

In the case of a key exchange, one party produce the secret key and encrypts it with the public key of the recipient. The recipient can decrypt it with their private key. The remaining communication would be completed with the secret key being the encryption key. Asymmetric encryption is used in key exchange, email security, Web security, and some encryption systems that needed key exchange over the public network.

The main problem for asymmetric-key cryptography is proving that a public key is authentic, and has not been tampered with or restored by a malicious third party. The beneficial method to this problem is to use a Public-key Infrastructure (PKI), in which one or more third parties, called a certificate authorities, certify ownership of key pairs. Another method, used by PGP, is the “web of trust” method to provide authenticity of key pairs.

Asymmetric key cryptography is a cryptographic technique that uses a several verification key in the CAD than the proving key used by the smart card. This method is usually implemented using trapdoor one-way functions, in which smart card creates an electronic signature with its secret key and the CAD needs a public key to authenticate the signature.

The Rivest-Shamir-Adelman (RSA) public key cryptosystem is the most frequently implemented asymmetric key authentication method. It places the security of the algorithm in the complexity of factoring large prime numbers.

The Digital Signature Standard (DSS) developed by the National Institute of Standards and Testing (NIST) can be used to smart card authentication. This method was produced for making document signatures in the U.S. Government and can be a preferred choice for government

application authentication. Its strength lies in the complexity in computing discrete logarithms. It has an advantage that several intermediate values for an authentication can be "pre-evaluated" so that the authentication time can be decreased.

The crux of public key systems, particularly those that are used for authentication purposes, is key distribution. The certificate enables the smart cards to support the verification key to the CAD. The certificate is formed when the card is boot up using a system wide secret key.

3.3] The RSA Algorithm :

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption. It was invented by Rivest, Shamir and Adleman in year 1978 and hence name **RSA** algorithm.

Algorithm

The RSA algorithm holds the following features –

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.
- There are two sets of keys in this algorithm: private key and public key.

You will have to go through the following steps to work on RSA algorithm –

Step 1: Generate the RSA modulus

The initial procedure begins with selection of two prime numbers namely p and q , and then calculating their product N , as shown –

$$N = p * q$$

Here, let N be the specified large number.

Step 2: Derived Number (e)

Consider number e as a derived number which should be greater than 1 and less than $(p-1)$ and $(q-1)$. The primary condition will be that there should be no common factor of $(p-1)$ and $(q-1)$ except 1

Step 3: Public key

The specified pair of numbers n and e forms the RSA public key and it is made public.

Step 4: Private Key

Private Key d is calculated from the numbers p , q and e . The mathematical relationship between the numbers is as follows –

$$ed = 1 \bmod (p-1)(q-1)$$

The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

Encryption Formula:

Consider a sender who sends the plain text message to someone whose public key is (n,e) . To encrypt the plain text message in the given scenario, use the following syntax –

$$C = P^e \bmod n$$

Decryption Formula:

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver C has the private key d , the result modulus will be calculated as –

$$\text{Plaintext} = C^d \bmod n$$

3.4] Symmetric And Asymmetric Key Cryptography

❖ Symmetric Key Cryptography

Symmetric Key Cryptography, or Symmetric Encryption, uses a secret key for both encryption and decryption. This approach is the inverse of Asymmetric Encryption, which uses one key to encrypt and another to decrypt. Data is translated to a format that cannot be interpreted or inspected by someone who does not have the secret key used to encrypt it during this phase.

The strength of the random number generator used to generate the secret key determines the effectiveness of this method. Symmetric Key Cryptography, commonly used on the Internet today, comprises two kinds of algorithms: Block and Stream. The Advanced Encryption Standard (AES) and the Data Encryption Standard (DES) are two common encryption algorithms. This type of encryption is typically much faster than Asymmetric Encryption, but it allows the secret key to be held by both the sender and the data receiver.

Symmetric cryptography is based on a single shared key that all parties are aware of and can use to encrypt and decrypt data.

Secret-key, single-key, shared-key, one-key, and private-key encryption are other words for symmetric-key cryptography. The usage of the last and first words will lead to misunderstanding compared to the related language used in public-key cryptography.

Symmetric key encryption employs one of the following encryption techniques:

Stream ciphers: Encrypt a message's digits or letters one at a time.

Block ciphers: Encrypt a group of bits as a single entity, inserting the plaintext to make it a block size multiple. 64-bit blocks are widely used. The NIST-approved Advanced Encryption Standard (AES) algorithm and the GCM block cipher mode of operation all use 128-bit blocks.

What is the Purpose of Symmetric Encryption?

Although symmetric encryption is an older type of encryption, it is simpler and more effective than asymmetric encryption, which strains networks due to data size performance problems and heavy CPU usage.

Since symmetric encryption performs smoother and quicker than asymmetric encryption, it is commonly used for bulk encryption / encrypting massive volumes of data, such as database encryption. In a database, the secret key can be used only by the database to encrypt or decrypt data.

Here are a few instances of where symmetric cryptography is used:

- Payment applications, such as bank purchases, where personally identifiable information (PII) must be secured to avoid identity theft or fraudulent charges
- Validations are performed to ensure that the sender of a message is who he appears to be.
- Hashing or random number creation

Advantages

- A symmetric cryptosystem is more effective.
- Encrypted data can be transmitted over a network in Symmetric Cryptosystems even though it is certain that the data would be intercepted. Since no key is sent with the files, the chances of data decryption are zero.
- To confirm the receiver's existence, a symmetric cryptosystem employs password authentication.
- A message can only be decrypted by a device that has a hidden key.
- Prevents widespread message protection breaches. For communicating with each party, a separate secret key is used. Only communications from a specific pair of sender and recipient are impacted when a key is corrupted. Communication with others is always safe.
- This type of encryption is simple to implement. All users need to do is specify and exchange the secret key until they can begin encrypting and decrypting messages.
- Encrypt and decrypt your files. There is no need to build separate keys if you use encryption for messages or data that you wish to access just once. For this, single-key encryption is ideal.

- Symmetric key encryption is much faster than asymmetric key encryption.
- Uses fewer computer resources. As opposed to public-key encryption, single-key encryption needs fewer computing resources.

Disadvantages

- Key transportation is a concern in symmetric cryptosystems. The secret key must be sent to the receiving device before the final message is sent. Electronic communication is unreliable, and no one can guarantee the communication networks will not be tapped. As a result, the only safe method of sharing keys will be to do it in person.
- It is not possible to have digital signatures that cannot be revoked.
- The message's origin and validity cannot be assured. Messages cannot be proven to have originated from a specific person since both sender and recipient use the same key. If there is a disagreement, this may be a challenge.
- For communication between each different party, a new shared key must be created. This poses a challenge with handling and securing both of these keys.

❖ Asymmetric Key Cryptography

Asymmetric cryptography, better known as public-key cryptography, encrypts and decrypts a message using a pair of similar keys. In asymmetric key cryptography, the private key is kept by one public key and one private key — to prevent unauthorized entry or usage. Anybody can use a public key to encrypt a document so that only the expected receiver can decrypt it with their private key. A private key or secret key is only known to the key's generator.

When anyone tries to submit an encrypted message, they will use a shared directory to retrieve the recipient's public key and use it to encrypt the message until submitting it. The message will then be decrypted by the receiver using their associated private key.

However, when the sender encrypts the message using their private key, the message may only be decrypted using the sender's public key, thus authenticating the sender. These encryption and decryption procedures are automatic; users don't need to lock and unlock the message manually.

Numerous protocols, including the transport layer security (TLS) and safe sockets layer (SSL) protocols that allow HTTPS, depend on asymmetric cryptography. Encryption is often used in browsers that need to create a stable link over an unstable network, such as the Internet, or to verify a digital signature.

The key advantage of asymmetric cryptography is increased data security. Since users are never expected to disclose or exchange their private keys, the risks of cyber activity on a user's private key during transmission are reduced.

What is the Purpose of Asymmetric Encryption?

Asymmetric encryption uses: Asymmetric cryptography is often used to check the authenticity of data using digital signatures. A digital signature is a cryptographic technique for verifying the validity and credibility of a message, software, or digital record. It's the equivalent of an in-person signature or a sealed seal in digital form.

Digital signatures, which are based on asymmetric cryptography, may include proof of the origin, identification, and status of an electronic record, transaction, or post, as well as acknowledge the signer's informed consent.

You can also use asymmetric encryption in applications where many users can encrypt and decode messages, such as:

- Encrypted email entails using a public key to encrypt a file and a private key to decode it.
- Asymmetric encryption is often used in the SSL/TSL cryptographic protocols, which provide encrypted connections between websites and browsers.
- Asymmetric encryption is used for Bitcoin and other crypto currencies when consumers provide public keys that everybody will use and private keys held hidden. Bitcoin employs a cryptographic algorithm to guarantee that only the funds belonging to rightful owners can be spent.
- Each unspent transaction output (UTXO) in the Bitcoin ledger is usually associated with a public key. So, suppose user A needs to give user B some money and has a UTXO associated with his public key. In that case, he uses his private key to sign a contract that spends the UTXO and makes a new UTXO associated with user B's public key.

Advantages

- There is no need to exchange keys in asymmetric or public key cryptography, eliminating the key distribution issue.
- The main benefit of public-key cryptography is improved security: private keys are never exchanged or exposed to others.
- May provide digital signatures that can be revoked.
- Message verification is provided by public-key cryptography, which requires the use of digital signatures, which allows the receiver of a message to check that the message is actually from a specific sender.
- The usage of digital signatures in public-key cryptography helps the recipient to determine whether or not the message was altered during transit. A digitally signed message cannot be altered without rendering the signature null.

- Signing a message digitally is analogous to physically signing a document. Since it is an acknowledgement of the message, the sender cannot reject it.

Disadvantages

- One drawback of using public-key cryptography for encryption is the lack of speed. Popular secret-key encryption systems are substantially quicker than any commonly accessible public-key encryption technique.
- Authentication of public keys is recommended/required. No one can be certain that a public key corresponds to the individual it identifies, so everybody must verify that their public keys are theirs.
- It consumes more computer resources. It necessitates much more computing resources than single-key encryption.
- A widespread security breach is likely if an intruder obtains a person's private key and reads his or her entire message.
- The loss of a private key can be irreversible. When a private key is lost, all incoming messages cannot be decrypted.

Symmetric Key Cryptography	Asymmetric Key Cryptography
There is just one key (symmetric key) used, and it is the same key used to encrypt and decrypt the message.	For encryption and decryption, two different cryptographic keys (asymmetric keys), known as the public and private keys, are used.
Since it is a straightforward procedure, the encryption method can be completed easily.	It is a much more complex and time-consuming mechanism than symmetric key encryption.
Key lengths are usually 128 or 256 bits, depending on the security criteria.	The key length is even greater; for asymmetric encryption example, recommended RSA key size is 2048 bits or higher.
It is also called secret-key cryptography or private key cryptography.	Asymmetric key is also called a conventional cryptography system or public-key cryptography.
Represented mathematically as: $P=D(K,E(P))$	Represented mathematically as: $P=D(K_d,E(K_e,P))$
It uses fewer resources than an asymmetric key cipher uses.	It consumes more resources than symmetric key cryptography.
It is used where massive amounts of data may be transmitted.	It is mainly used in smaller transactions to authenticate and create a stable contact channel before data transfer.
The secret key is shared. As a result, the likelihood of compromise is increased.	Since the private key is not shared, the overall process is more reliable than symmetric encryption.
Symmetric encryption is an old technique.	Asymmetric encryption is relatively new.
RC4, AES, DES, 3DES, and other algorithms are	RSA, Diffie-Hellman, ECC, and other asymmetric encryption algorithms

examples.

are examples.

3.5] Digital Signature :

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

Model of Digital Signature :

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration – Diag

The following points explain the entire process in detail –

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.
- Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence **signing a hash is more efficient than signing the entire data.**

Importance of Digital Signature :

Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security.

Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature –

- **Message authentication** – When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.
- **Data Integrity** – In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.
- **Non-repudiation** – Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely – Privacy, Authentication, Integrity, and Non-repudiation.

3.5.1] Encryption with Digital Signature:

In many digital communications, it is desirable to exchange an encrypted messages than plaintext to achieve confidentiality. In public key encryption scheme, a public (encryption) key of sender is available in open domain, and hence anyone can spoof his identity and send any encrypted message to the receiver.

This makes it essential for users employing PKC for encryption to seek digital signatures along with encrypted data to be assured of message authentication and non-repudiation.

This can archived by combining digital signatures with encryption scheme. Let us briefly discuss how to achieve this requirement. There are **two possibilities, sign-then-encrypt** and **encrypt-then-sign**.

However, the crypto system based on sign-then-encrypt can be exploited by receiver to spoof identity of sender and sent that data to third party. Hence, this method is not preferred. The process of encrypt-then-sign is more reliable and widely adopted. This is depicted in the following illustration – Diag

The receiver after receiving the encrypted data and signature on it, first verifies the signature using sender's public key. After ensuring the validity of the signature, he then retrieves the data through decryption using his private key.

3.6] Knapsack And Other Algorithm

Knapsack Encryption Algorithm is the first general public key cryptography algorithm. It is developed by **Ralph Merkle** and **Mertin Hellman** in 1978. As it is a Public key cryptography, it needs two different keys. One is Public key which is used for Encryption process and the other one is Private key which is used for Decryption process. In this algorithm we will use two different knapsack problems in which one is easy and other one is hard. The easy knapsack is used as the private key and the hard knapsack is used as the public key. The easy knapsack is used to derived the hard knapsack.

For the easy knapsack, we will choose a **Super Increasing knapsack problem**. Super increasing knapsack is a sequence in which every next term is greater than the sum of all preceding terms.

Example –

{1, 2, 4, 10, 20, 40} is a super increasing as
 $1 < 2$, $1+2 < 4$, $1+2+4 < 10$, $1+2+4+10 < 20$ and $1+2+4+10+20 < 40$.

Derive the Public key

- **Step-1:**

Choose a super increasing knapsack {1, 2, 4, 10, 20, 40} as the private key.

- **Step-2:**

Choose two numbers n and m. Multiply all the values of private key by the number n and then find modulo m. The value of m must be greater than the sum of all values in private key, for example 110. And the number n should have no common factor with m, for example 31.

- **Step-3:**

Calculate the values of Public key using m and n.

$$1 \times 31 \bmod(110) = 31$$

$$2 \times 31 \bmod(110) = 62$$

$$4 \times 31 \bmod(110) = 14$$

$$10 \times 31 \bmod(110) = 90$$

$$20 \times 31 \bmod(110) = 70$$

$$40 \times 31 \bmod(110) = 30$$

- Thus, our public key is {31, 62, 14, 90, 70, 30}

And Private key is {1, 2, 4, 10, 20, 40}.

Now take an example for understanding the process of encryption and decryption.

Example –

Lets our plain text is 100100111100101110.

1. Encryption :

As our knapsacks contain six values, so we will split our plain text in a groups of six:

100100 111100 101110

Multiply each values of public key with the corresponding values of each group and take their sum.

100100 {31, 62, 14, 90, 70, 30}

$$1 \times 31 + 0 \times 62 + 0 \times 14 + 1 \times 90 + 0 \times 70 + 0 \times 30 = 121$$

111100 {31, 62, 14, 90, 70, 30}

$$1 \times 31 + 1 \times 62 + 1 \times 14 + 1 \times 90 + 0 \times 70 + 0 \times 30 = 197$$

101110 {31, 62, 14, 90, 70, 30}

$$1 \times 31 + 0 \times 62 + 1 \times 14 + 1 \times 90 + 1 \times 70 + 0 \times 30 = 205$$

So, our cipher text is 121 197 205.

2. Decryption :

The receiver receive the cipher text which has to be decrypt. The receiver also knows the values of m and n.

So, first we need to find the $n^{-1} \pmod{m}$, which is multiplicative inverse of n mod m i.e.,

$$n \times \text{mod}(m) = 131 \times \text{mod}(110) = 1 = 71$$

Now, we have to multiply 71 with each block of cipher text take modulo m.

$$121 \times 71 \pmod{110} = 11$$

Then, we will have to make the sum of 11 from the values of private key {1, 2, 4, 10, 20, 40} i.e.,

$$1 + 10 = 11 \text{ so make that corresponding bits 1 and others 0 which is } 100100.$$

Similarly,

$$197 \times 71 \pmod{110} = 17$$

$$1 + 2 + 4 + 10 = 17 = 111100$$

$$\text{And, } 205 \times 71 \pmod{110} = 35$$

$$1 + 4 + 10 + 20 = 35 = 101110$$

After combining them we get the decoded text.

100100111100101110 which is our plain text.

Example:

Here's a Python example of the Knapsack Encryption Algorithm with proper explanation and output:

- Python

```
import random
```

```
# Function to generate a super-increasing sequence for the public key
```

```
def generate_super_increasing_sequence(n):
```

```
    sequence = [random.randint(1, 100)]
```

```
    while len(sequence) < n:
```

```
        next_element = sum(sequence) + random.randint(1, 10)
```

```
        sequence.append(next_element)
```

```
    return sequence
```

```
# Function to generate the private key from the public key
```

```
def generate_private_key(public_key, q, r):
```

```
    private_key = [(r * element) % q for element in public_key]
```

```
    return private_key
```

```
# Function to encrypt the plaintext using the public key
```

```
def knapsack_encrypt(plaintext, public_key):
```

```
    encrypted_message = sum(public_key[i] for i in range(len(plaintext)) if plaintext[i] == '1')
```

```
    return encrypted_message
```

```
# Function to decrypt the ciphertext using the private key
```

```
def knapsack_decrypt(ciphertext, private_key, q):
```

```
r_inverse = pow(r, -1, q) # Modular multiplicative inverse of r
```

```
decrypted_message = ""
```

```
for element in reversed(private_key):
```

```
    if (ciphertext * r_inverse) % q >= element:
```

```
        decrypted_message = '1' + decrypted_message
```

```
        ciphertext -= element
```

```
    else:
```

```
        decrypted_message = '0' + decrypted_message
```

```
return decrypted_message
```

```
# Example usage
```

```
if __name__ == "__main__":
```

```
    n = 8 # Number of elements in the super-increasing sequence
```

```
    q = 103 # Modulus (should be greater than the sum of the super-increasing sequence)
```

```
    r = 3 # Multiplier for generating private key
```

```
# Generate the public key and private key
```

```
public_key = generate_super_increasing_sequence(n)
```

```
private_key = generate_private_key(public_key, q, r)
```

```
plaintext = "11001010"

ciphertext = knapsack_encrypt(plaintext, public_key)

decrypted_message = knapsack_decrypt(ciphertext, private_key, q)

print("Original Message:", plaintext)

print("Encrypted Ciphertext:", ciphertext)

print("Decrypted Message:", decrypted_message)
```

Explanation:

1. The 'generate_super_increasing_sequence()' function generates a super-increasing sequence, which serves as the public key. The function starts with a random element and adds random values to it until the sequence reaches the desired length n .
2. The 'generate_private_key()' function generates the private key from the public key using the formula: $\text{private_key}[i] = (r * \text{public_key}[i]) \% q$.
3. The 'knapsack_encrypt()' function takes the plaintext and the public key as input. It converts the plaintext to binary and encrypts it by summing the elements of the public key corresponding to '1' bits in the binary representation of the plaintext.
4. The 'knapsack_decrypt()' function takes the ciphertext, private key, and modulus q as input. It uses the private key to reconstruct the original binary representation of the plaintext by finding '1' bits in the ciphertext and subtracting the corresponding elements from the private key.

Output:

Original Message: 11001010

Encrypted Ciphertext: 426

Decrypted Message: 11001010

In this example, the plaintext "11001010" is encrypted using the Knapsack Encryption Algorithm, resulting in the ciphertext "426". The ciphertext is then decrypted back to the original plaintext, which demonstrates the correctness of the algorithm.

Unit 4 – Digital Certificate And Public Key Infrastructure (PKI)

4.1] Introduction

Digital certificates and PKI

Digital certificates facilitate secure electronic communication and data exchange between people, systems, and devices online. They are issued by Certificate Authorities (CAs) and perform two primary functions:

- Verifying the identity of the sender/receiver of an electronic message
- Providing the means to encrypt/decrypt messages between sender and receiver (i.e., binding and entity to their public key)

There are three basic types of digital signature certificates:

- **Individual digital signature certificates (signing certificates):** These certificates are used to identify a person and include personal information. They can be used to sign electronic documents (i.e., to provide electronic signatures) and emails, and to implement access control mechanisms for sensitive or valuable information.
- **Server certificates:** These certificates identify a server (computer) and contain the host name or IP address. They are used for one- or two-layer SSL to ensure secure communication of data over a network.
- **Encryption certificates:** These certificates are used to encrypt a message using the public key of the recipient to ensure data confidentiality during transmission. Different signatures for encryption and digital signatures are available from different CAs. (adapted from [Government of India 2010](#))
- **A system—including policies, institutions, and technologies—that manages the distribution, authentication, and revocation of digital certificates is often referred to as public-key infrastructure (PKI).** Because digital certificates are standard in data exchange and security protocols for digital ID systems (including the TLS encryption measures described above, as well as smartcard- and mobile-based authentication), a country's PKI landscape is a common building block for many ID systems.

For example, when a smartcard or SIM card that uses PKI for authentication and digital signatures is personalized, it is issued with a private key and digital certificate signed by a CA that attests to the authenticity of the credential and provides the public-key necessary for other devices (e.g., card readers, servers, etc.) to verify the authenticity and integrity of the card.

4.2] Digital Certificates

Digital Certificates are a standard of security for establishing an encrypted link between a server and a client. Generally, between a mail server or a webserver, which protects data in transitions by encrypting them. A Digital Certificate is also a Digital ID or a passport which is issued by a Third Party Authority which verifies the identity of the server's owner and not claiming a false identity. (Diag)

Components of a Digital Certificate :

All these following components can be found in the certificate details –

- **Serial Number** – Used to uniquely identify the certificate.
- **Subject** – The person, or entity identified.
- **Signature Algorithm** – The algorithm used to create the signature.
- **Signature** – The actual signature to verify that it came from the issuer.
- **Issuer** – The entity that verified the information and issued the certificate.
- **Valid-From** – The date the certificate is first valid from.
- **Valid-To** – The expiration date.
- **Key-Usage** – Purpose of the public key (For example: encipherment, signature, certificate signing...).
- **Public Key** – The public key.
- **Thumbprint Algorithm** – The algorithm used to hash the public key certificate.
- **Thumbprint** – The hash itself, used as an abbreviated form of the public key certificate.

Levels of Validations

In this section we will discuss the various levels of validations of an SSL (Secure Sockets Layer) Certificate. Some of the most important ones are as follows –

- **Domain Validation SSL Certificate** – It validates the domain that is registered by a system administrators and they have administrator rights to approve the certificate request, this validation generally is done by email request or by a DNS record.
- **Organization Validated SSL Certificates** – It validates the domain ownership and also the business information like the Official Name, City, Country, etc. This validation is done by email or DNS record entering and the certificate authority would also need some genuine documents to verify the Identity.
- **Extended Validation SSL Certificates** – It validates domain ownership and organization information, plus the legal existence of the organization. It also validates that the organization is aware of the SSL certificate request and approves it. The validation requires documentation to certify the company identity plus a set of additional steps and checks. The Extended Validation SSL Certificates are generally identified with a green address bar in the browser containing the company name.

4.3] Private Key Management

A private key, also known as a *secret key*, is a variable in cryptography that is used with an algorithm to encrypt and decrypt data. Secret keys should only be shared with the key's generator or parties authorized to decrypt the data. Private keys play an important role in symmetric cryptography, asymmetric cryptography and [crypto currencies](#).

A private key is typically a long, randomly or pseudo-randomly generated sequence of bits that cannot be easily guessed. The complexity and length of the private key determine how easily an attacker can execute a [brute-force attack](#), where they try out different keys until the right one is found.

How does a private key work?

Private key encryption is also referred to as [symmetric encryption](#), where the same private key is used for both encryption and decryption. In this case, a private key works as follows:

- **Generating a new private key.** Prior to encryption, generate a new key that is as random as possible; encryption software is typically used to generate private keys.
- **Securely storing the private key.** Once generated, the private key must be stored securely. Depending on the application, keys may be stored offline or on the computer used to generate, encrypt and decrypt data. Private keys may be protected with a password, encrypted or hashed for security -- or all three.
- **Key exchange.** The private key is used to decrypt, as well as to encrypt, so using it for symmetric encryption requires a [key exchange](#) to share that key securely with trusted parties authorized to exchange secured data. Cryptographic software is usually used to automate this process.
- **Key management.** Private key management is required to prevent any individual key from being used for too long. It helps to securely retire keys after their useful lifetime is reached.

A private key is also used in [asymmetric cryptography](#), which is also known as [public key cryptography](#). In this case, the private key refers to the secret key of a public key pair. In public key cryptography, the private key is used for encryption and digital signatures. It works as follows for asymmetric cryptography:

- **Generating a public-private key pair.** Randomness is even more important for this process. Encryption application software is usually used to generate key pairs. It should require a source of randomness, such as mouse movement.
- **Securely storing the private key.** Once generated, the private key must be stored securely. Like the symmetric cryptography process, keys may be stored offline or on the computer used to generate, encrypt and decrypt data. Here, too, private keys should be protected with a password, encrypted or hashed for security.
- **Key exchange.** The private key of a public key pair should almost never be shared with others. Public key cryptography, including digital signatures, is typically used to securely share session keys used for symmetric encryption. However, other protocols for [public key infrastructure](#) are used to authoritatively share public keys between cooperating parties.
- **Using the private key.** The owner of a public key pair uses their private key for decrypting data that has been encrypted with the public key of the pair. Only the holder of the private key should be able to decrypt data encrypted with the public key. For digital signatures, the owner of the key pair uses their private key to encrypt the signature. In this way, anyone with access to the public key can decrypt the signature and verify that it was signed by the private key owner.
- **Key management.** Public key pairs are often generated with expiration dates, and key management is vital to maintaining access to data protected with a key pair. For example, an expired public key certificate, which depends on a public key pair, may cause browsers to flag access to a website as insecure. Secret keys should be stored with the highest security, and public key pairs should be managed to avoid compromise or issues related to key pair expiration.

Advantages of private encryption keys:

Private key encryption provides several useful features. They include the following four benefits:

1. **More secure.** Private keys that are longer and have greater [entropy](#), or randomness, are more secure from brute-force or dictionary attacks.
2. **Faster.** Symmetric key encryption is faster computationally than asymmetric encryption with its public-private key pairs.
3. **Best for encryption.** Most cryptographic processes use private key encryption to encrypt data transmissions. They typically use a public key algorithm to securely share secret keys.
4. **Work for stream and block ciphers.** Secret key [ciphers](#) -- the algorithm for encrypting and decrypting data -- generally fall into one of two categories: stream ciphers or [block ciphers](#). A block cipher applies a private key and algorithm to a block of data simultaneously, whereas a stream cipher applies the key and algorithm one bit at a time.

Private keys can be shared securely using the Diffie-Hellman public key exchange protocol.(Diag)

Challenges of private encryption key management :

The security of encryption keys depends on choosing a strong encryption algorithm and maintaining high levels of operational security. [Encryption key management](#) is necessary for any organization using encryption to protect its data. That goes for symmetric, as well as asymmetric, encryption.

While private key encryption can ensure a high level of security, the following key management challenges must be considered:

- **Overall management.** Encryption key management is necessary to protect cryptographic keys from loss, corruption or unauthorized access.
- **Continual updating.** Private keys used to encrypt sensitive data should be changed regularly to minimize exposure should they be leaked or stolen.

- **Recoverability and loss.** If an encryption key becomes inaccessible, data encrypted with that key will be unrecoverable and lost.

Private keys vs. public keys

Asymmetric cryptography, also known as *public key cryptography*, uses pairs of public and private keys. These two different but mathematically linked keys are used to transform plaintext into encrypted [ciphertext](#) or encrypted text back to plaintext.

When the public key is used to encrypt ciphertext, that text can only be decrypted using the private key. This approach enables anyone with access to the public key to encrypt a message, and only the private key holder will be able to decrypt it.

4.4] The PKIX Model

As we know, the X.509 standard defines the digital-certificate structure, format and fields.

It also specifies the procedure for distributing the public keys.

In order to extend such standards and make them universal, the Internet Engineering Task Force (IETF) formed the Public Key Infrastructure X.509(PKIX) working group.

This extends the basic philosophy of the X.509 standard, and specifies how the digital certificates can be deployed in the world of the Internet.

Services of PKIX :

PKIX identifies the primary goals of a PKI infrastructure in the form of the following broad level services:

Registration -It is the process where an end-entity (subject) makes itself known to a CA. Usually, this is via an RA.

Initialization– This deals with the basic problems, such as the methodology of verifying that the end-entity is talking to the right CA.

Certification—In this step, the CA creates a digital certificate for the end-entity and returns it to the end-entity, maintains a copy for its own records, and also copies it in public directories, if required.

Key-Pair Recovery Keys— used for encryption may be required to be recovered at a later date for decrypting some old documents. Key archival and recovery services can be provided by a CA or by an independent key-recovery system.

Key Generation PKIX specifies that the end-entity should be able to generate private-and public-key pairs, or the CA/RA should be able to do this for the end-entity (and then distribute these keys securely to the end-entity).

Key Update

This allows a smooth transition from one expiring key pair to a fresh one, by the automatic renewal of digital certificates. However, there is a provision for manual digital certificate renewal request and response.

Cross-certification

Helps in establishing trust models, so that end-entities that are certified by different CAs can cross-verify each other.

Revocation PKIX provides support for the checking of the certificate status in two modes: online (using OCSP) or offline (using CRL).

OR

PKIX stands for Public Key Infrastructure X.509. The X.509 standard defines the structure, format, and fields for digital certificates, it also specifies the procedures for distributing public keys. To extend these standards and make them universal IETF (Internet Engineering Task Force) forms the PKIX working group. PKIX extends the basic philosophy of the X.509 standard and specifies the implementation of digital certificates in the world of the Internet. Additionally, other PKI models have been defined for application use in various domains. For example, ANSI, ASC X9F standards are used by financial organizations.

PKIX Services :

Services provided by are as follows:

- **Registration:** It is a process where end-entity registers itself to a CA. Usually, the registration is done via the RA.
- **Initialization:** This deals with basic problems such as the methodology of verifying that the end entity is talking to the right CA.
- **Certification:** It is a process where CA creates a digital certificate for end-entity and returns it to the end entity. CA also maintain a copy for certificate for its records. If required, CA also copied it in public directories.
- **Key pair recovery:** Keys which are used for encrypting documents may be required to be recovered later for decrypting the same old documents. Key archival and recovery services can be provided by CA or by an independent key recovery system.
- **Key generation:** PKIX model specifies that the end entity should be able to generate the public key and private key pairs or CA should be able to do this for the end entity.
- **Key update:** It is a process where the expired key of the digital certificate is automatically renewed and replaced with a new key pair. However, there is a provision for manual digital certificate renewal requests and responses.
- **Cross certification:** It is a process where end entities that re-certified by different CA, can cross verify each other. It helps in establishing trust models.
- **Revocation:** PKIX model provides support for checking certificate status in two modes, online using OCSP and offline using CRL.

PKIX Architectural Model

PKIX has developed a document that describes five areas of its architectural model. These areas are as follows:

1. 509 V3 certificate and V2 certificate revocation list profiles

X.509 standard allows the use of various options while describing the extension of digital certificates. PKIX has grouped all options that are deemed fit for internet users. It calls this group of options as the profile of internet users. This profile is described in RFC2459 and specifies which attributes must/may/may not be supported. Appropriate value ranges for the values used in each extension category are also provided. For instance, the X.509 standard does not specify the instruction codes when the certificate is suspended. PKIX defines them.

2. *Operational protocols*

These define the underlying protocols that provide the transport mechanism for delivering certificates. CRLs and other management and status information to PKI users. Since each of these requirements demands a different way of service, how to use HTTP, LDAP, FTP, X.500, etc. are defined for this purpose.

3. Management Protocols

These protocols enable exchange information between various PKI entities. For example, how to carry registration request revocation status or cross-certification request and response. The management protocol specifies the structure of the message that floats between the entities. They also specify what details are required to process these messages. Examples of management protocols include CMP (Certificate Management Protocol) for requesting a certificate.

4. Policy outlines

PKIX defines the outlines for CP (Certificate Policies) and CPS (Certificate Practice Statements) in RFC2527. These define the policies for the creation of a document such as certificate policies which determine what considerations are important when choosing a type of certificate for a particular application domain.

5. Timestamp and data certification service

Time stamping service is provided by a trusted third party which is called Time Stamp Authority. The main purpose of this service is to sign a message to guarantee that it existed before a specific date and time. This helps deal with non-repudiation claims. DCS (Data certification Service) is a trusted third party's service that verifies the correctness of the data that it receives. This is similar to the notary service in real life, where for instance, it can use it for getting one's property certified.

Conclusion

PKIX stands for Public Key Infrastructure X.509 standard is a model which deals with the issue related to PKI technology i.e. Public Key Infrastructure. In this article, we have discussed the concept of PKIX with its working, Services, and architecture.

4.5] Public Key Cryptography Standards (PKCS)

Public key cryptography standards may not be in the everyday lexicon of a business, but every business in every industry runs on it. Large technology companies make a concerted effort to integrate these standards into their infrastructure. Companies in sectors such as healthcare, education, and even government unwittingly subscribe to these standards when they incorporate various [SaaS](#) solutions and cloud-based infrastructure decisions into their systems.

At an individual level, every website and application accessed from a personal device is also subject to PKCS standards. It makes sense, considering how much personal information flows into an online shopping or dating app.

The reasons that PKCS are so essential are:

1. Security.

There isn't a single business in any industry today that is not connected to the internet. Even basic mailing software is subject to cyber-attacks. According to the Cost of a Data Breach 2022 Report by IBM, the average cost of a [data breach](#) in the United States is \$9.44 million.

The PKCS standards ensure no gaps while implementing encryption across these different applications consumed. They provide an encryption checklist for [cyber security](#) personnel.

2. Compliance

With so much private and sensitive data doing the rounds, governing bodies across all industries have created regulations to control how this data is transmitted and stored. Encrypted data is usually the first mandate in all these regulations.

For example, HIPAA regulations in the healthcare industry require that patients' protected health information (PHI) and electronic PHI (ePHI) must be encrypted while stored and in transit. Failing to comply with HIPAA can cost around \$50,000 per violation.

3. Versatility

New devices and exceptional hardware capabilities are surfacing every day. The concept of stand-alone hardware is moot now, with every device recording, transmitting, or transmitting some form of data.

The [Internet-of-things](#) (IoT) has caught on, with data constantly being uploaded and used to make operations more efficient. An architecture that complies with the PKCS standards makes it easier to adapt these new technologies without necessarily uprooting the sections of the existing system.

4. Interoperability

The most significant advantage of using PKCS is the ability to securely allow various hardware and software forms to communicate without many development overheads. This makes it easier to explore different solutions and vendors. It also works well in the current cloud-driven infrastructure models.

The 15 PKCS Specifications

Each PKCS standard deals with a specific aspect of the PKI. They are reviewed and maintained individually and updated as necessary. The 15 standards are:

1. PKCS #1 – RSA cryptography standard

Description: This is the base standard that defines the syntax of the public and private keys, the format of the messages and digital signatures, the different algorithms to be used for each level of encryption and decryption, and the format of the digital certificate.

Besides the syntax of the key, it also specifies which mathematical requirements the keys must satisfy to be unbreachable for [brute-force attacks](#). Some libraries implementing PKCS#1 are BSAFE, cryptlib, OpenSSL, and Nettle.

Is this standard still maintained? Yes. It was also republished as RFC 8017 in 2012. RFC is a formal document maintained by the Internet engineering task force (IETF) that contains specifications and notes for everything related to computing and networking.

Current version: 2.2

2. PKCS #2 – RSA message digest standard

Description: The PKCS #2 standard defines how message digests must be created and encrypted.

Is this standard still maintained? No, this standard was withdrawn in 2010 and merged with PKCS #1.

Current version: Nil

3. PKCS #3 – Diffie-Helman key agreement standard

Description: This standard is named after Diffie and Helman, who first developed the public/private key mode of encryption and decryption. It defines how the receiver and sender can establish a secret key to envelope private communication.

This key is assumed to be created and shared in a non-secure [network](#). Many key establishment schemes, such as ANSI X9.63 and IEEE 1363a, come into play here. Organizations such as Oracle have their own implementation of this standard.

Is this standard still maintained? Yes, it is still being used.

Current version: 1.4

4. PKCS #4 – RSA key syntax standard

Description: This standard covers the key syntax to be used by RSA.

Is this standard still maintained? No. PKCS #4 was withdrawn in 2010 and merged with the first standard.

Current version: Nil

5. PKCS #5 – Password-based cryptography standard

Description: PKCS #5 explores the usage of passwords in the key agreement phase. It specifies how one should apply pseudo-random functions such as HMAC to the password along with a salt value. This text derived from the password now forms the secret key. This additional layer of encryption makes up for the pitfalls of using plain, user-fed passwords during communication.

Password-Based Key Derivation Function 2 (PBKDF2) is the recommended implementation by PKCS #5's version 2.1. It involves hashing the password repeatedly to derive the key. The number of repetitions is based on the CPU speed. The more repetitions, the harder it is to break. However, newer and more robust implementations such as Argon2 and SCrypt are being used today.

Is this standard still maintained? Yes, this standard still plays a crucial role in encrypted communication. It has also been published as Internet Engineering Task Force's RFC 2898.

Current version: 2.1

6. PKCS #6 – Extended-certificate syntax standard

Description: PKCS #6 was meant to provide the syntax for extending and modifying the attributes established in a digital certificate.

Is this standard still maintained? No, this standard was made obsolete by the third version of X.509. X.509 is a different standard that defines the format of public key certificates.

Current version: Nil

7. PKCS #7 – Cryptographic message syntax standard

Description: Cryptographic message syntax (CMS) standard specifies the syntax of stored, encrypted data. PKCS#7 is used by certificate authorities to store digital certificates that they've issued.

It also specifies how to handle certificates that have been invalidated for reasons other than expiry. Any storage of digital signatures should also adhere to PKCS #7. One implementation of this standard can be seen in [Single sign-on](#) applications.

Is this standard still maintained? This standard forms the basis for RFC 5652, and while still being implemented, it is being taken over by RFC 3369.

Current version: 1.5

8. PKCS #8 – Private key information syntax standard

Description: As the name suggests, this standard defines the syntax for private keys. It explains which algorithms and attributes suit the best to create a private key. It also looks into how users must store private keys.

Usually, this standard is used in conjunction with PKCS #5, using a passcode and a salt to store private keys.

Is this standard still maintained? This standard is maintained and available as RFC 5208.

Current version: 1.2

9. PKCS #9 – Selected attribute types standard

Description: PKCS #9 defines the data type, length, and other details of attributes necessary for certificates, signatures, and private keys. It does not give the format of the certificate or the key itself. It deals with the attributes or details necessary to create one.

For example, a business name and URL are necessary attributes for certain digital certificates.

Is this standard still maintained? PKCS #9 is still being supported.

Current version: 2.0

10. PKCS #10 – Certification request syntax standard

Description: PKCS #10 specifies the format of the messages sent to a certificate authority (CA) to request a digital certificate. This is called a certificate signing request.

These messages usually contain the public key chosen by a business, domain name or other identifying details, and proof of authenticity.

Is this standard still maintained? Yes, this standard is still being maintained.

Current version: 1.7

11. PKCS #11 – Cryptographic token interface standard

Description: The cryptographic token interface standard is also known as Cryptoki. It describes a platform-independent [application programming interface](#) (API) that can be used by cryptography token generators such as smart cards and key fobs.

The API is a generic interface that allows the generation, modification, and deletion of the different types of keys and certificates used by security hardware.

This standard ensures that encryption is platform-agnostic and allows various devices to communicate without extensive workarounds.

Cryptoki serves well in scenarios that require single sign-on and disk encryption systems. Certificate authorities use it to access the CA's signing key and to generate new certificates. Organizations such as Oracle Solaris and Red Hat have their own implementations for proprietary use.

Is this standard still maintained? PKCS #11 is still being supported, with RSA Security turning it over to the OASIS PKCS 11 technical committee for all further work.

Current version: 3.0

12. PKCS #12 – Personal information exchange syntax standard

Description: PKCS #12 defines the file format in which the private key issued with the public key is stored. It also specifies how to store various other private information, such as the digital certificate.

More often than not, all the private encryption information is bundled and stored in internal containers called Safe Bags. Some Safe Bags have a predefined structure that enables the storage of certificate requests, X.509 certificates, and private keys. Some Safe Bags are more flexible and allow the storage of custom, private information.

Besides storage, PKCS #12 also provides the mechanism for transferring this data across a network, along with the necessary syntax. PKCS #12 is built on PKCS #8 which only deals with private keys. It adds an extra layer to check for the integrity of this data both in transit and storage.

Is this standard still maintained? Yes, PKCS #12 is still being maintained.

Current version: 1.1

13. PKCS #13 – Elliptic curve cryptography standard

Description: The elliptic curve cryptography (ECC) standard was a type of cryptography that used modified mathematical methods in encryption.

The current public key algorithm is the RSA algorithm, created by Rivest, Shamir, and Adelman. It involves the modular exponentiation of the product of two large prime numbers.

ECC tries to find a distinct logarithm within a random elliptic curve. This yields more secure keys than RSA but is still in the exploration phase.

Is this standard still maintained? While proposed initially in 1998, there is no documentation of this standard available.

Current version: Nil

14. PKCS #14 – Pseudo-random number generation standard

Description: Random number generation plays a significant role in PKI, particularly with enveloping and hashing. It makes sense to standardize this aspect of encrypted communication. However, progress has yet to be made on this front.

Is this standard still maintained? No, there is no documentation of PKCS #14 that is available as of today.

Current version: Nil

15. PKCS #15 – Cryptographic token information format standard

Description: While all these standards have been established keeping interoperability in mind, it also makes sense to specify a standard that will allow for customization of certain aspects of encryption.

For example, integrated circuit cards are cryptographic tokens used across various devices. PKCS #15 specified the format of the credentials required by these tokens to identify themselves to the user. This is akin to digital certificates but at a token level.

Is this standard still maintained? PKCS #15 is still supported, though some nuances have been handed over to ISO/IEC 7816-15.

Current version: 1.1

4.6] XML

XML encryption is the process of encrypting and decrypting digital XML content, using certain syntax and algorithms. The basic concept of cryptography remains much the same, but the difference is in adopting a standard format for representing and exchanging encrypted XML data.

This standard format, often known as the XML encryption specification, includes a standard syntax for representing the encrypted content within the XML, as well as information required to decrypt its contents at the receiving end.

The W3C XML Encryption Working Group—a collaborative effort between the [W3C](#) (World Wide Web Consortium) and the [IETF](#) (Internet Engineering Task Force)—advocates XML encryption standards and specifications. (W3C and IETF are two of the frontline organizations involved in defining interoperable standards and specifications for XML/Web services technologies.)

As of March 2002, the Working Group has released the [Candidate Recommendation Specification](#) for XML encryption, which includes the following:

- XML Encryption Requirements
- XML Encryption Syntax and Processing

Inspirations behind the XML Encryption specification effort, can be summarized as follows:

- It is good to have standard element tags for representing encrypted elements within the XML documents. This will enable parsers to better understand encrypted elements and data during the validation process.
- It is necessary to provide means for encrypting only the desired elements within an XML document instead of encrypting the whole document. This will pave the way for incorporating several confidential data elements that are intended for different recipients within a single XML document.

- There should be standard mechanisms for exchanging the secret keys used for encryption and decryption processes.
- The standard should allow encryption of different parts of the document with different keys, so that multiple recipients can decrypt only those portions that are intended for them.
- The standards should be adaptable to both ASCII and binary data.
- The standards should be adaptable to different cryptographic algorithms.
- The standards should work along with other XML security standards and specifications

4.7] PKI And Security

Public key infrastructure or PKI is the governing body behind issuing digital certificates. It helps to protect confidential data and gives unique identities to users and systems. Thus, it ensures security in communications.

The public key infrastructure uses a pair of keys: the public key and the private key to achieve security. The public keys are prone to attacks and thus an intact infrastructure is needed to maintain them.

Managing Keys in the Cryptosystem:

The security of a cryptosystem relies on its keys. Thus, it is important that we have a solid key management system in place. The 3 main areas of key management are as follows:

- A cryptographic key is a piece of data that must be managed by secure administration.
- It involves managing the key life cycle which is as follows:
 - **Keeping the private key secret:** Only the owner of a private key is authorized to use a private key. It should thus remain out of reach of any other person.
 - **Assuring the public key:** Public keys are in the open domain and can be publicly accessed. When this extent of public accessibility, it becomes hard to know if a key is correct and what it will be used for. The purpose of a public key must be explicitly defined.

PKI or public key infrastructure aims at achieving the assurance of public key.

Public Key Infrastructure:

Public key infrastructure affirms the usage of a public key. PKI identifies a public key along with its purpose. It usually consists of the following components:

- A digital certificate also called a public key certificate
- Private Key tokens
- Registration authority
- Certification authority
- CMS or Certification management system

Working on a PKI:

Let us understand the working of PKI in steps.

- **PKI and Encryption:** The root of PKI involves the use of [cryptography](#) and [encryption](#) techniques. Both [symmetric](#) and [asymmetric encryption](#) uses a public key. The challenge here is – “how do you know that the public key belongs to the right person or to the person you think it belongs to?”. There is always a risk of MITM(Man in the middle). This issue is resolved by a PKI using digital certificates. It gives identities to keys in order to make the verification of owners easy and accurate.
- **Public Key Certificate or Digital Certificate:** Digital certificates are issued to people and electronic systems to uniquely identify them in the digital world. Here are a few noteworthy things about a digital certificate. Digital certificates are also called X.509 certificates. This is because they are based on the ITU standard X.509.
 - The Certification Authority (CA) stores the public key of a user along with other information about the client in the digital certificate. The information is signed and a digital signature is also included in the certificate.
 - The affirmation for the public key then thus be retrieved by validating the signature using the public key of the Certification Authority.
- **Certifying Authorities:** A CA issues and verifies certificates. This authority makes sure that the information in a certificate is real and correct and it also digitally signs the certificate. A CA or ***Certifying Authority performs these basic roles:***
 - Generates the key pairs – This key pair generated by the CA can be either independent or in collaboration with the client.
 - Issuing of the digital certificates – When the client successfully provides the right details about his identity, the CA issues a certificate to the client. Then CA further signs this certificate digitally so that no changes can be made to the information.
 - Publishing of certificates – The CA publishes the certificates so that the users can find them. They can do this by either publishing them in

an electronic telephone directory or by sending them out to other people.

- Verification of certificate – CA gives a public key that helps in verifying if the access attempt is authorized or not.
- Revocation – In case of suspicious behavior of a client or loss of trust in them, the CA has the power to revoke the digital certificate.

Classes of a Digital Certificate:

A digital certificate can be divided into four broad categories. These are :

- Class 1: These can be obtained by only providing the email address.
- Class 2: These need more personal information.
- Class 3: This first checks the identity of the person making a request.
- Class 4: They are used by organizations and governments.

Process of creation of certificate:

The creation of a certificate takes place as follows:

- Private and public keys are created.
- CA requests identifying attributes of the owner of a private key.
- Public key and attributes are encoded into a CSR or Certificate Signing Request.
- Key owner signs that CSR to prove the possession of a private key.
- CA signs the certificate after validation.

Creation of Trust layers among CA Hierarchies:

Each CA has its own certificate. Thus, trust is built hierarchically where one CA issues certificates to other CAs. Moreover, there is a root certificate that is self-signed. For a root CA, the issuer and the subject are not two separate parties but a single party.

Security of Root CA:

As you saw above, the ultimate authority is the root CA. Hence, the security of root CA is of huge importance. If the private key of a root CA is not taken care of, then it might turn into a catastrophe. This is because anyone disguised as the root CA can then issue certificates. To meet security standards, a root CA should be offline 99.9% of the time. However, it does need to come online to create public and private keys and to issue new certificates. Ideally, these activities should be performed 2-4 times a year.

Use of PKI in Today's Digital Age:

Today, there are an enormous number of applications that need require authentication. Certifications are needed at millions of places. This can not be done without a Public key infrastructure. The importance of PKI, depending on the use case and needs, has evolved over time. Here is a part of that track.

- For the very first time during the period of 1995 to 2002, the use of PKI was limited to the most important and high-value certificates. This included the certificates of eCommerce websites that enabled them to display the lock icon in the search bar. The goal was to make consumers confident about the security and authenticity of various websites.
- The second episode of PKI emerged around 2003 to 2010 when enterprises came into the picture. It was at this time that employees received laptops and the use of mobile phones was rising. Thus, employees needed access to the organization's assets even outside the office. That is when the use of PKI looked like the best way for authentication.
- The third phase started in 2011 and is continuing to date. With the advent of new technologies like IoT(Internet of Things) and need the to scale PKI, the use, as well as the challenges in using PKI, have increased tremendously. Today, millions of certificates are issued to authenticate mobile workforces. However, managing this huge number of certificates is quite challenging.
- S/MIME, Document Signing, code or app signing also uses PKI.

Challenges that a PKI Solves:

PKI owes its popularity to the various problems its solves. Some use cases of PKI are:

- Securing web browsers and communicating networks by SSL/TLS certifications.
- Maintaining Access Rights over Intranets and VPNs.
- Data Encryption
- Digitally Signed Software
- Wi-fi Access Without Passwords

Other than these, one of the most important use cases of PKI is based around IoT(Internet of Things). Here are two industries that are using PKI for IoT devices:

- **Auto Manufacturers:** Cars these days have features like GPS, call for services, assistants, etc. These require communication paths where a lot of data is passed. Making these connections secure is very important to avoid malicious parties hacking into the cars. This is where PKI comes in.
- **Medical device Manufacturers:** Devices like surgical robots require high security. Also, FDA mandates that any next-generation medical device must be updatable so that bugs can be removed and security issues can be dealt with. PKI is used to issues certificates to such devices.

Disadvantages of PKI:

- **Speed:** Since PKI uses super complex algorithms to create a secure key pair. So it eventually slows down the process and data transfer.
- **Private Key Compromise:** Even though PKI can't be hacked very easily but a private key can be hacked by a professional hacker, since PKI uses Public and Private key to encrypt and decrypt data so with user's private key in hand and public key which is easily available the information can be decrypted easily.

4.8] Creating Digital Certificate

The steps required to create a [digital certificate](#) involves three parties first the end user, second the registration authority and third is certificate authority. The end user request for a digital certificate and the request goes to the registration authority(RA) which then assist the certificate authority(CA) to create the digital certificate. Registration authority act as a intermediate between end user and the certificate authority. It also assist in day to day task of certificate authority.

Services of Registration Authority:

- Accepting and verifying the details of new user's registration.
- User key generation.
- Backups and recovery of key.
- Certificate cancellation.

Steps for Digital Certificate Creation:

- **Step-1:** Key generation is done by either user or registration authority. The public key which is generated is sent to the registration authority and private key is kept secret by user.
- **Step-2:** In the next step the registration authority registers the user.
- **Step-3:** Next step is verification which is done by registration authority in which the user's credentials are being verified by registration authority. It also checks that the user who send the public key have corresponding private key or not.
- **Step-4:** In this step the details are sent to certificate authority by registration authority who creates the digital certificate and give it to users and also keeps a copy to itself.

Unit 5 – Internet Security Protocols

5.1] Introduction And Concepts

Internet security refers to securing communication over the internet. It includes specific security protocols such as:

- Internet Security Protocol (IPSec)
- Secure Socket Layer (SSL)

❖ Internet Security Protocol (IPSec)

It consists of a set of protocols designed by Internet Engineering Task Force (IETF). It provides security at network level and helps to create authenticated and confidential packets for IP layer.

❖ Secure Socket Layer (SSL)

It is a security protocol developed by Netscape Communications Corporation.). It provides security at transport layer. It addresses the following security issues:

- Privacy
- Integrity
- Authentication

❖ Threats

Internet security threats impact the network, data security and other internet connected systems. Cyber criminals have evolved several techniques to threat privacy and integrity of bank accounts, businesses, and organizations.

Following are some of the internet security threats:

- Mobile worms
- Malware
- PC and Mobile ransomware
- Large scale attacks like Stuxnet that attempts to destroy infrastructure.
- Hacking as a Service
- Spam
- Phishing

❖ Email Phishing

Email phishing is an activity of sending emails to a user claiming to be a legitimate enterprise. Its main purpose is to steal sensitive information such as usernames, passwords, and credit card details.

Such emails contains link to websites that are infected with malware and direct the user to enter details at a fake website whose look and feels are same to legitimate one.

- What a phishing email may contain?

Following are the symptoms of a phishing email:

Spelling and bad grammar

Most often such emails contain grammatically incorrect text. Ignore such emails, since it can be a spam.

Beware of links in email

Don't click on any links in suspicious emails.

Threats

Such emails contain threat like “your account will be closed if you didn't respond to an email message”.

Spoofing popular websites or companies

These emails contain graphics that appear to be connected to legitimate website but they actually are connected to fake websites.



Internet Protocol Security (IPSec) is a framework of open standards for ensuring private, secure communications over Internet Protocol (IP) networks, through the use of [cryptographic security](#) services. IPSec is a suite of cryptography-based protection services and security protocols. Because it requires no changes to programs or protocols, you can easily deploy IPSec for existing networks.

The driving force for the acceptance and deployment of secure IP is the need for business and government users to connect their private WAN/LAN infrastructure to the Internet for providing access to Internet

services and use of the Internet as a component of the WAN transport system. As we all know, users need to isolate their networks and at the same time send and receive traffic over the Internet. The authentication and privacy mechanisms of secure IP provide the basis for a security strategy for us.

IPsec protects one or more paths between a pair of hosts, a pair of security gateways, or a security gateway and a host. A security gateway is an intermediate device, such as a switch or firewall, that implements IPsec. Devices that use IPsec to protect a path between them are called peers.

IPsec requires a PCI Accelerator Card (PAC) to provide hardware data compression and encryption. A PAC is a hardware processing unit the switch's CPU controls.

IPsec provides the following security services for traffic at the IP layer:

- Data origin authentication—identifying who sent the data.
- Confidentiality (encryption)—ensuring that the data has not been read en route.
- Connectionless integrity—ensuring the data has not been changed en route.
- Replay protection—detecting packets received more than once to help protect against denial of service attacks.

Applications of IPsec

As we all know to help in the security of a network the Internet community has done lot of work and developed application-specific security mechanisms in numerous application areas, including electronic mail (*Privacy Enhanced Mail*, *Pretty Good Privacy* [PGP]), network management (*Simple Network Management Protocol Version 3* [SNMPv3]), Web access (Secure HTTP, *Secure Sockets Layer* [SSL]), and others. However, users have some security concerns that cut across protocol layers. For example, an enterprise can run a secure, private TCP/IP network by disallowing links to untrusted sites, encrypting packets that leave the premises, and authenticating packets that enter the premises. By implementing security at the IP level, an organization can ensure secure networking not only for applications that have security mechanisms but also for the many security-ignorant applications.

Benefits of IPSec :

- When IPSec is implemented in a firewall or router, it provides strong security whose application is to all traffic crossing this perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.
- IPSec is below the transport layer (TCP, UDP), and is thus transparent to applications. There is no need to change software on a user or server system when IPSec is implemented in the firewall or router.
- Even if IPSec is implemented in end systems, upper layer software, including applications is not affected. IPSec can be transparent to end users.
- There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization. IPSec can provide security for individual users if needed. This feature is useful for offsite workers and also for setting up a secure virtual sub network within an organization for sensitive applications.

5.2] Secure Socket Layer

[Secure Socket Layer \(SSL\)](#) provides security to the data that is transferred between web browser and server. SSL encrypts the link between a web server and a browser which ensures that all data passed between them remain private and free from attack.

Secure Socket Layer Protocols:

- SSL record protocol
- Handshake protocol
- Change-cipher spec protocol
- Alert protocol

SSL Protocol Stack: Diag

SSL Record Protocol:

SSL Record provides two services to SSL connection.

- Confidentiality
- Message Integrity

In the SSL Record Protocol application data is divided into fragments. The fragment is compressed and then encrypted MAC (Message Authentication Code) generated by algorithms like SHA (Secure Hash Protocol) and MD5 (Message Digest) is appended. After that encryption of the data is done and in last SSL header is appended to the data. (Diag)

Handshake Protocol:

Handshake Protocol is used to establish sessions. This protocol allows the client and server to authenticate each other by sending a series of messages to each other.

Handshake protocol uses four phases to complete its cycle.

- **Phase-1:** In Phase-1 both Client and Server send hello-packets to each other. In this IP session, cipher suite and protocol version are exchanged for security purposes.
- **Phase-2:** Server sends his certificate and Server-key-exchange. The server ends phase-2 by sending the Server-hello-end packet.
- **Phase-3:** In this phase, Client replies to the server by sending his certificate and Client-exchange-key.
- **Phase-4:** In Phase-4 Change-cipher suite occurs and after this the Handshake Protocol ends.

SSL Handshake Protocol Phases diagrammatic representation

Change-cipher Protocol:

This protocol uses the SSL record protocol. Unless Handshake Protocol is completed, the SSL record Output will be in a pending state. After the handshake protocol, the Pending state is converted into the current state.

Change-cipher protocol consists of a single message which is 1 byte in length and can have only one value. This protocol's purpose is to cause the pending state to be copied into the current state.

Alert Protocol:

This protocol is used to convey SSL-related alerts to the peer entity. Each message in this protocol contains 2 bytes.

The level is further classified into two parts:

Warning (level = 1):

This Alert has no impact on the connection between sender and receiver. Some of them are:

Bad certificate: When the received certificate is corrupt.

No certificate: When an appropriate certificate is not available.

Certificate expired: When a certificate has expired.

Certificate unknown: When some other unspecified issue arose in processing the certificate, rendering it unacceptable.

Close notify: It notifies that the sender will no longer send any messages in the connection.

Unsupported certificate: The type of certificate received is not supported.

Certificate revoked: The certificate received is in revocation list.

Fatal Error (level = 2):

This Alert breaks the connection between sender and receiver. The connection will be stopped, cannot be resumed but can be restarted. Some of them are :

Handshake failure: When the sender is unable to negotiate an acceptable set of security parameters given the options available.

Decompression failure: When the decompression function receives improper input.

Illegal parameters: When a field is out of range or inconsistent with other fields.

Bad record MAC: When an incorrect MAC was received.

Unexpected message: When an inappropriate message is received.

The second byte in the Alert protocol describes the error.

Salient Features of Secure Socket Layer:

- The advantage of this approach is that the service can be tailored to the specific needs of the given application.
- Secure Socket Layer was originated by Netscape.
- SSL is designed to make use of TCP to provide reliable end-to-end secure service.
- This is a two-layered protocol.

Versions of SSL:

SSL 1 – Never released due to high insecurity.

SSL 2 – Released in 1995.

SSL 3 – Released in 1996.

TLS 1.0 – Released in 1999.

TLS 1.1 – Released in 2006.

TLS 1.2 – Released in 2008.

TLS 1.3 – Released in 2018.

SSL (Secure Sockets Layer) certificate is a digital certificate used to secure and verify the identity of a website or an online service. The certificate is issued by a trusted third-party called a Certificate Authority (CA), who verifies the identity of the website or service before issuing the certificate.

The SSL certificate has several important characteristics that make it a reliable solution for securing online transactions:

1. **Encryption:** The SSL certificate uses encryption algorithms to secure the communication between the website or service and its users. This ensures that the sensitive information, such as login credentials and credit card information, is protected from being intercepted and read by unauthorized parties.
2. **Authentication:** The SSL certificate verifies the identity of the website or service, ensuring that users are communicating with the intended party and not with an impostor. This provides assurance to users that their information is being transmitted to a trusted entity.
3. **Integrity:** The SSL certificate uses message authentication codes (MACs) to detect any tampering with the data during transmission. This ensures that the data being transmitted is not modified in any way, preserving its integrity.
4. **Non-repudiation:** SSL certificates provide non-repudiation of data, meaning that the recipient of the data cannot deny having received it. This is important in situations where the authenticity of the information needs to be established, such as in e-commerce transactions.
5. **Public-key cryptography:** SSL certificates use public-key cryptography for secure key exchange between the client and server. This allows the client and

server to securely exchange encryption keys, ensuring that the encrypted information can only be decrypted by the intended recipient.

6. **Session management:** SSL certificates allow for the management of secure sessions, allowing for the resumption of secure sessions after interruption. This helps to reduce the overhead of establishing a new secure connection each time a user accesses a website or service.
7. **Certificates issued by trusted CAs:** SSL certificates are issued by trusted CAs, who are responsible for verifying the identity of the website or service before issuing the certificate. This provides a high level of trust and assurance to users that the website or service they are communicating with is authentic and trustworthy.

5.3] Transport Layer Security

Transport Layer Securities (TLS) are designed to provide security at the transport layer. TLS was derived from a security protocol called [Secure Socket Layer \(SSL\)](#). TLS ensures that no third party may eavesdrop or tamper with any message. There are several benefits of TLS:

- **Encryption:**
TLS/SSL can help to secure transmitted data using encryption.
- **Interoperability:**
TLS/SSL works with most web browsers, including Microsoft Internet Explorer and on most operating systems and web servers.
- **Algorithm flexibility:**
TLS/SSL provides operations for authentication mechanism, encryption algorithms and hashing algorithm that are used during the secure session.
- **Ease of Deployment:**
Many applications TLS/SSL temporarily on a windows server 2003 operating systems.
- **Ease of Use:**
Because we implement TLS/SSL beneath the application layer, most of its operations are completely invisible to client.

Working of TLS:

The client connect to server (using [TCP](#)), the client will be something. The client sends number of specification:

1. Version of SSL/TLS.
2. which cipher suites, compression method it wants to use.

The server checks what the highest SSL/TLS version is that is supported by them both, picks a cipher suite from one of the clients option (if it supports one) and optionally picks a compression method. After this the basic setup is done, the server provides its certificate. This certificate must be trusted either by the client itself or a party that the client trusts. Having verified the certificate and being certain this server really is who he claims to be (and not a man in the middle), a key is

exchanged. This can be a public key, “PreMasterSecret” or simply nothing depending upon cipher suite.

5.4] Secure Hypertext Transport Protocol

Secure Hypertext Transfer Protocol (S-HTTP) is a protocol used for encrypting [HTTP traffic](#), enhancing the security of data as it travels across the [World Wide Web](#). Essentially, S-HTTP is designed to secure individual messages transmitted between clients and servers, differentiating itself from other security protocols by its granularity and message-oriented approach.

- **Basic Functionality and Principles**

The primary functionality of S-HTTP is to provide end-to-end encryption of web communications. It achieves this by encrypting the data before transmission and decrypting it upon receipt. This process ensures confidentiality and protects against eavesdropping and tampering. S-HTTP supports various encryption algorithms, enabling flexible security arrangements based on the needs of the communication parties. Furthermore, it provides options for authentication, ensuring that both sender and receiver can verify each other’s identity, adding an additional layer of security to web interaction. S-HTTP is also a server software.

The Secure Hypertext Transfer Protocol is not as widely implemented as Secure Sockets Layer ([SSL](#)), which is independent of protocol and works with HTTP, Simple Mail Transfer Protocol (SMTP), [Network News Transfer Protocol](#) (NNTP), and other Internet protocols.

S-HTTP is also the name given to World Wide Web (WWW) server software that implements the S-HTTP protocol. This software was developed by Enterprise Integrations Technologies (EIT), the National Center for Supercomputing Applications (NCSA), and RSA Security.

2. The Creation of S-HTTP

Historical Context and Origin

S-HTTP was developed in the early [1990s](#), a time when the internet was rapidly expanding, and the need for secure communication methods was becoming increasingly apparent. The protocol was developed to address the lack of inherent security in the HTTP protocol, which is the foundational technology for data communication on the World Wide Web.

The development of S-HTTP was primarily led by R. T. Fielding and J. C. Mogul. Their work was instrumental in the early stages of defining and standardizing the protocol. The key milestones in the development of S-HTTP include:

1. **Initial Proposal:** The concept of Secure Hypertext Transfer Protocol was proposed to address the growing concerns over web security and privacy.
2. **Collaboration and Standardization:** Collaborative efforts among various internet security experts and organizations led to the refinement and standardization of the protocol.
3. **RFC Publication:** The formal documentation of S-HTTP, known as RFC 2660, was published in 1999, detailing the specifications and use cases of the protocol. (see the [references section](#))

The creation of S-HTTP marked a significant step in the evolution of web security, laying the groundwork for future developments in secure web communication

3. How S-HTTP Works

Technical Mechanism and Encryption Process

S-HTTP operates by encrypting individual HTTP messages, unlike other protocols that encrypt the entire communication session. This mechanism provides flexibility in securing specific parts of the data being transmitted. Each message, whether it's a request from the client or a response from the server, is encrypted independently.

The encryption process involves several key steps:

1. **Encryption Initiation:** When a message is ready for transmission, S-HTTP initiates encryption using a selected algorithm.
2. **Key Exchange:** S-HTTP supports various key exchange mechanisms. These keys are used to encrypt and decrypt messages at both ends.
3. **Data Encryption:** The actual data in the HTTP message is encrypted using the agreed-upon encryption standards, rendering it unreadable to unauthorized parties.
4. **Transmission and Decryption:** Once the encrypted message reaches its destination, it is decrypted using the corresponding decryption key.

Setting Up and Configuring S-HTTP

To set up S-HTTP:

1. **Server and Client Configuration:** Both the server and client must support S-HTTP. This involves configuring the web server and client applications to handle S-HTTP requests and responses.
2. **Certificate Management:** Obtain and install [digital certificates](#) to facilitate authentication and key exchange.
3. **Selecting Encryption Algorithms:** Choose suitable encryption algorithms based on the required level of security and the capabilities of the server and client.

4. S-HTTP vs. HTTPS: Understanding the Differences

Comparative Analysis of Protocols

While both S-HTTP and [HTTPS](#) are designed to secure HTTP communications, they differ fundamentally in their approach:

- **Encryption Scope:** Secure-HTTP encrypts individual messages, whereas HTTPS encrypts the entire communication session.
- **Flexibility:** SHTTP offers more flexibility in terms of what parts of the communication are encrypted. HTTPS, on the other hand, provides a uniform layer of encryption over all data exchanged.
- **Compatibility:** The Secure Hypertext Transfer Protocol can coexist with regular HTTP on a single server, while HTTPS typically requires a dedicated port.

Use Cases and Suitability

- **S-HTTP:** Best suited for scenarios where selective encryption of messages is needed, such as in applications that only require certain parts of the communication to be secured.
- **HTTPS:** Ideal for general web browsing and transactions where uniform security is required for all data exchanges, like in e-commerce or online banking.

Understanding these differences is crucial for selecting the right protocol based on specific security requirements and the nature of the data being transmitted.

[Back to Index](#)

5. Current Relevance of S-HTTP

Is S-HTTP Still Used Today?

S-HTTP, despite its initial promise, has seen limited adoption in modern web communications. The protocol, while innovative, has been overshadowed by the widespread acceptance and implementation of HTTPS. Today, S-HTTP is rarely used in standard web applications or by major [web browsers](#), making it more of a historical footnote in the evolution of secure web communication protocols.

Modern Implementations and Legacy

Although S-HTTP is not prevalent in current web applications, its legacy persists in the form of ideas and methodologies that have influenced modern encryption techniques. The concept of selectively encrypting web messages has been integrated into other technologies and protocols, albeit in different forms. S-HTTP's emphasis on flexible encryption and message-specific security remains a relevant concept in the design of secure communication systems.

[Back to Index](#)

6. Challenges and Limitations of S-HTTP

Technical Constraints

One of the major technical challenges facing S-HTTP was the complexity of its implementation. The protocol's message-specific encryption approach required more processing power and presented challenges in efficiently managing encryption keys. Moreover, the necessity for both client and server to support S-HTTP added to the complexity, making it less attractive compared to more straightforward alternatives like HTTPS.

Adoption and Compatibility Issues

S-HTTP's adoption was hindered by several factors:

- **Limited Browser and Server Support:** Many web browsers and servers did not adopt S-HTTP, favoring the more straightforward HTTPS protocol.
- **Compatibility with Web Infrastructure:** The existing web infrastructure, including caching mechanisms and proxy servers, was optimized for HTTP and HTTPS, leading to compatibility issues with S-HTTP.
- **Emergence of HTTPS as a Standard:** The rapid adoption of HTTPS as the de facto standard for secure web communication further marginalized S-HTTP, as developers and organizations preferred the more universally accepted protocol.

These challenges limited the widespread adoption of S-HTTP, relegating it to a niche role in the history of web security protocol.

7. Future Outlook and Evolution of Secure Web Protocols

Emerging Trends in Web Security

The future of web security is likely to be shaped by several emerging trends:

1. **Advanced Encryption Techniques:** The development of more sophisticated encryption methods to counteract evolving cyber threats.
2. **Quantum Computing:** The advent of [quantum computing](#) poses new challenges to current encryption standards, driving the need for quantum-resistant protocols.
3. **AI and Machine Learning:** The integration of AI and [machine learning](#) for predictive threat detection and automated security responses.
4. **Increased Focus on End-User Privacy:** Growing concerns over data privacy are prompting the development of protocols that enhance user anonymity and data protection.

Predictions for the Future of Secure HTTP and Related Technologies

Finally, while S-HTTP itself may not see a revival, its principles could influence future protocols:

- **Selective Encryption:** The concept of selective encryption, central to Secure-HTTP, may find relevance in specialized applications requiring flexible security measures.
- **Incorporation into Hybrid Protocols:** Elements of the Secure Hypertext Transfer Protocol might be integrated into new or existing protocols to enhance their security capabilities, especially in environments where message-level encryption is crucial.

5.5] Time Stamping Protocol

Timestamping is a fundamental aspect of end-to-end digitization that enables the verification of the order of events in a digital system. In simpler terms, timestamping aids in maintaining the record when a document has been **digitally signed**. It is a critical tool in ensuring the accuracy and integrity of various digital transactions, including financial transactions, legal documents, and **digitally signed documents**. Needless to say, **digital signing** is a growing trend and will continue to do so. This shifting impetus towards **digital signing solutions** has created a corresponding demand for timestamping.

Despite its importance, enterprises are yet to understand its role in ensuring integrity and confidentiality in the digital ecosystem. A lack of **timestamping protocol** in your **enterprise workflow** can have significant consequences, including data tampering, data loss, and legal disputes.

Therefore, in this blog, we aim to explain timestamping by providing a comprehensive guide to its mechanisms and applications. We will delve into the technical aspects of timestamping, exploring how it functions.

What is Timestamping?

Timestamping is a process of assigning a unique identifier to a specific event or transaction that occurs in a digital system. It involves recording the time and date when an event or transaction occurred, along with additional metadata that may be necessary to verify its authenticity and integrity.

The **primary purpose of timestamping** is to ensure the order of events. It plays a critical part in the workflow, where the accuracy and integrity of data are of utmost importance. For instance, in financial transactions, timestamping is used to provide evidence of the timing and order of trades, helping to prevent fraud and ensure fair and transparent markets. In legal transactions, timestamping is often used to authenticate documents and establish a chain of custody, helping to prevent tampering and provide evidence in legal disputes.

The Need for Timestamping

Timestamping plays a critical role in **ensuring trust and credibility** in the digital ecosystem, but the question remains, what is the need for assigning and recording time to digitally signed documents? Let us delve a little deeper into the growing need for timestamping.

To understand the **need for timestamping** let us recap the concept of digital signatures. **Digital signatures** are equivalent to traditional handwritten signatures, used to signify the signer's intent to agree to the terms and conditions of a document or transaction. Employing a **Public Key Infrastructure (PKI)**-based digital signature to digitally execute a document ensures minimal manual intervention. While it undoubtedly has provided ease of doing business, implementing such a process requires addressing the issue of non-repudiation.

For instance, when a document is signed with a **PKI-based certificate**, it captures the individual's information and the signature's date and time. However, the date and time depend on the computer's local time, which is prone to tampering. One could change the computer's time to make an expired or revoked certificate appear valid. Therefore, ensuring the authenticity of the digital signature requires additional measures.

This is where timestamping comes to play. Enterprises can resolve the issue at hand by employing a **Time Stamping Authority (TSA)**, which leverages the **RFC 3161 Time-Stamp Protocol (TSP)**. This protocol substantiates that the data has not been subjected to any form of unauthorized modification, thereby ensuring the integrity of the information in question. By integrating the **use of a TSA** in the process of document or code signing, any attempts to tamper with the timestamp will be effectively prevented.

Trusted Timestamping

A **trusted timestamp** is a **digital timestamp** that is created and verified by a trusted **third-party timestamping authority (TSA)** using secure **FIPS-compliant hardware**. It is used to establish the authenticity and integrity of a digital document or message by providing a reliable record of the time at which it was created, sent, or received. It employs a **cryptographic algorithm** that generates unique code associated with a document or message.

The code is further encrypted using a **PKI certificate** which ensures that the timestamp is authentic and has not been tampered with. It is important to note that the reliability of a trusted timestamp depends on the TSA that creates and verifies the timestamp. Therefore, it is essential to choose a **reputable and trustworthy TSA, like eMudhra** to ensure that the trusted timestamp is reliable and can be used to establish the authenticity and integrity of the digital document or message.

How does Timestamping Work in Digital Documents?

The process of timestamping a digitally signed document can be concluded in the following steps:

- A hash code is created for the data or the document that is to be timestamped.
- The hash code is sent to a TSA (time stamping authority) which adds time to the data.
- Now, the hash value and the time are hashed together (typically a SHA-1/2 hash value) to create a unique hash value.
- The newly created unique hash value is then encrypted with TSA's private key using PKI technology.
- The digital signature of the document concatenated with the time stamp is sent to the original owner of the document.
- When the document is accessed, the encrypted code and timestamp can be decrypted using the PKI certificate to confirm that the document has not been altered since the timestamp was applied.

This provides a reliable record of the **document's authenticity and integrity** and can be used to demonstrate compliance with legal and regulatory requirements.

Timestamping in Network Security

In **network security**, **timestamping** protocols are used to provide a reliable and accurate way of recording the occurrence of events in a networked environment. The primary function of timestamping in this context is to establish a chronological order of events and to provide evidence of the occurrence of a particular event at a specific time.

One of the **primary applications of timestamping in network security** is in the prevention of replay attacks, where an attacker intercepts and replays a legitimate message to gain unauthorized access to a system. Timestamping can be used to prevent such attacks by ensuring that each message is unique and only valid for a specific time.

Furthermore, timestamping can also be used to provide evidence of compliance with legal or regulatory requirements, such as data retention laws. By accurately timestamping data, organizations can prove when specific events occurred and demonstrate that they have met their legal and regulatory obligations.

Overall, timestamping protocols play a critical role in network security by providing an accurate and reliable way of recording events in a networked environment. By ensuring the integrity and authenticity of data, timestamping helps to prevent attacks and provides a basis for legal and regulatory compliance.

Legal Framework of Timestamping Across the Globe

In India, the **use of timestamping** is recognized under the **Information Technology Act, of 2000**. The act defines **electronic signatures** and provides for their legal recognition, which includes the use of timestamping to establish the authenticity and integrity of electronic documents. The Indian government has also established a framework for **digital signatures and certificates**, which includes the use of a trusted third-party timestamping authority.

In the United States, it is governed by **federal and state laws**, as well as industry-specific regulations. The Electronic Signatures in **Global and National Commerce Act (ESIGN)** and the **Uniform Electronic Transactions Act (UETA)** provide legal recognition for electronic signatures, including the use of timestamping.

In Europe, the legality is governed by the **eIDAS Regulation**, which provides a framework for **electronic identification, authentication, and trust services**. The regulation recognizes its use as a means of establishing the authenticity and

integrity of documents and recognizes the use of a trusted third-party timestamping authority.

The **use of timestamping in digital transactions** is becoming increasingly important as more industries shift to digital platforms. While the legal framework varies across different countries and regions, there is a growing recognition of the **importance of timestamping** in ensuring the authenticity and integrity of documents.

5.5] Secure Electronic Transaction

Secure Electronic Transaction or SET is a system that ensures the security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied to those payments. It uses different encryption and hashing techniques to secure payments over the internet done through credit cards. The SET protocol was supported in development by major organizations like Visa, Mastercard, and Microsoft which provided its Secure Transaction Technology (STT), and Netscape which provided the technology of Secure Socket Layer (SSL).

SET protocol restricts the revealing of credit card details to merchants thus keeping hackers and thieves at bay. The SET protocol includes Certification Authorities for making use of standard Digital Certificates like X.509 Certificate.

Before discussing SET further, let's see a general scenario of electronic transactions, which includes client, payment gateway, client financial institution, merchant, and merchant financial institution. (Diag)

Requirements in SET: The SET protocol has some requirements to meet, some of the important requirements are:

- It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is an intended user or not, and merchant authentication.
- It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.
- It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.
- SET also needs to provide interoperability and make use of the best security mechanisms.

Participants in SET: In the general scenario of online transactions, SET includes similar participants:

1. **Cardholder** – customer
2. **Issuer** – customer financial institution

3. **Merchant**
4. **Acquirer** – Merchant financial
5. **Certificate authority** – Authority that follows certain standards and issues certificates (like X.509V3) to all other participants.

SET functionalities:

- **Provide Authentication**
 - **Merchant Authentication** – To prevent theft, SET allows customers to check previous relationships between merchants and financial institutions. Standard X.509V3 certificates are used for this verification.
 - **Customer / Cardholder Authentication** – SET checks if the use of a credit card is done by an authorized user or not using X.509V3 certificates.
- **Provide Message Confidentiality:** Confidentiality refers to preventing unintended people from reading the message being transferred. SET implements confidentiality by using encryption techniques. Traditionally DES is used for encryption purposes.
- **Provide Message Integrity:** SET doesn't allow message modification with the help of signatures. Messages are protected against unauthorized modification using RSA digital signatures with SHA-1 and some using HMAC with SHA-1,

Dual Signature: The dual signature is a concept introduced with SET, which aims at connecting two information pieces meant for two different receivers :

Order Information (OI) for merchant

Payment Information (PI) for bank

You might think sending them separately is an easy and more secure way, but sending them in a connected form resolves any future dispute possible. Here is the generation of dual signature: Diag.

Where,

PI stands for payment information

OI stands for order information

PIMD stands for Payment Information Message Digest

OIMD stands for Order Information Message Digest

POMD stands for Payment Order Message Digest

H stands for Hashing

E stands for public key encryption

KPc is customer's private key

|| stands for append operation

Dual signature, $DS = E(KPc, [H(H(PI)||H(OI))])$

Purchase Request Generation: The process of purchase request generation requires three inputs:

- Payment Information (PI)
- Dual Signature
- Order Information Message Digest (OIMD)

The purchase request is generated as follows:

Here,

PI, OIMD, OI all have the same meanings as before.

The new things are :

EP which is symmetric key encryption

Ks is a temporary symmetric key

KUbank is public key of bank

CA is Cardholder or customer Certificate

Digital Envelope = $E(KUbank, Ks)$

Purchase Request Validation on Merchant Side: The Merchant verifies by comparing POMD generated through PIMD hashing with POMD generated through decryption of Dual Signature as follows: Diag

Since we used Customer's private key in encryption here we use KUC which is the public key of the customer or cardholder for decryption 'D'.

Payment Authorization and Payment Capture: Payment authorization as the name suggests is the authorization of payment information by the merchant which ensures payment will be received by the merchant. Payment capture is the process by which a merchant receives payment which includes again generating some request blocks to gateway and payment gateway in turn issues payment to the merchant.

The disadvantages of Secure Electronic Exchange: At the point when SET was first presented in 1996 by the SET consortium (Visa, Mastercard, Microsoft, Verisign, and so forth), being generally taken on inside the following couple of years was normal. Industry specialists additionally

anticipated that it would immediately turn into the key empowering influence of worldwide internet business. Notwithstanding, this didn't exactly occur because of a few serious weaknesses in the convention.

The security properties of SET are better than SSL and the more current TLS, especially in their capacity to forestall web based business extortion. Be that as it may, the greatest downside of SET is its intricacy. SET requires the two clients and traders to introduce extraordinary programming – – card perusers and advanced wallets – – implying that exchange members needed to finish more jobs to carry out SET. This intricacy likewise dialed back the speed of web based business exchanges. SSL and TLS don't have such issues.

5.6] SSL Vs SET

S. No.	Secure Socket Layer	Secure Electronic Transaction
	<p style="text-align: center;">Basics-</p> <p>1. SSL is an encryption mechanism for order taking, queries, and other applications and is available on the customer's browser. It does not protect against all security hazards and is naturally simple and widely used. SSL is a protocol for general-purpose secure message exchange. SSL protocol may use a certificate, but the payment gateway is not available. So, the merchant needs to receive both the ordering information and credit card information because the capturing process should be generated by the merchant. SSL protocol has been the industry standard for securing internet communication.</p>	<p style="text-align: center;">Basics-</p> <p>SET is a very comprehensive protocol. It provides privacy, integration, and authenticity. It is not used frequently due to its complexity and the need for a special card reader by the user. It may be abandoned if it is not simplified. SET is tailored to the credit card payment to the merchant. SET protocols hide the customer's credit card information from merchant and also hides the order information from banks to protect privacy called a dual signature. The SET protocol is complex and more secure.</p>
2.	<p style="text-align: center;">Developed by-</p> <p>SSL protocol was developed by Netscape for the secure online</p>	<p style="text-align: center;">Developed by-</p> <p>The SET protocol was jointly developed by MasterCard and visa to secure web browsers for a bank</p>

S. No.	Secure Socket Layer	Secure Electronic Transaction
	transaction.	card transaction.
3.	Working- SSL uses a combination of public-key and symmetric-key encryption to safeguard data transactions. The handshake technique is used by the SSL protocol, which permits the server to verify its identity to the client. In case of unsuccessful authentication, the connection will not be formed.	Working- The dual signature mechanism is deployed by SET to safeguard a transaction. To use an e-commerce site, SET requires the purchase of software. The design of the protocol necessitates the client's installation of an e-wallet.
4.	Integrity- The technique of Hash functions is used for this purpose.	Integrity- The technique of digital signatures is used for this purpose.
5.	Acceptability- Its acceptability is more as compared to SET.	Acceptability- SET acceptability is less because it's necessary to build an open PKI.
6.	Functionality- The Secure Sockets Layer (SSL) is not a payment protocol. SSL encrypts the communication channel between the cardholder and the merchant website and is not backed by any financial institution. As a result, SSL is unable to ensure the security of a transaction.	Functionality- SET was created with the sole purpose of securing and ultimately guaranteeing a payment transaction. For example, increase in the possibilities for online retail growth only when consumer confidence grows in online shopping.
7.	Encryption- The purpose of SSL lies in prevention of data tampering in client/server applications and has considerably weaker encryption, with a maximum of 128-bit	Encryption- SET, which was created expressly to address the security of all parties involved in an electronic payment transaction, uses 1024-bit encryption throughout the

S. No.	Secure Socket Layer	Secure Electronic Transaction
	encryption.	transaction.
8.	Authentication- SSL certificates are not endorsed by any financial institution or payment brand association, so they cannot effectively validate all parties.	Authentication- Here, all parties get authentication to the transaction because SET's certificates are backed not just by a Certificate Authority, but also by financial institutions and MasterCard International.
9.	Security- SSL only protects the cardholder and the merchant, which is insufficient to prevent fraud. SSL transactions, in other words, are never assured.	Security- SET enables transaction security from the cardholder's desktop to the merchant via bank approvals and back through the gateway, leaving an indisputable audit trail and, as a result, a guaranteed transaction.

5.7] 3-D Secure protocol

What is 3D Secure?

3D Secure is a security protocol that is designed to provide an additional layer of security for online credit card and debit card transactions. This protocol was created by Arcot Systems (now CA Technologies) and was first used by Visa to provide improved security for Internet payments.

It's likely that you have heard of programs such as Verified by Visa, MasterCard Secure Code, J/Secure, and American Express Safe Key. These payment security services are based on 3D Secure.

How Does 3D Secure Work?

The name 3D comes from the three-domain model used to provide the additional layer of secure authentication between the financial authorization process and online authentication process. The three domains used to provide this security are:

- **Acquirer Domain:** The bank and the merchant receiving the transaction payment.

- **Issuer Domain:** The bank that issued the credit or debit card used for the transaction.
- **Interoperability Domain:** The infrastructure provided for the card that's used to support the 3D Secure protocol.

3D Secure uses XML messaging and SSL communication to secure and authenticate transactions.

As a merchant, it's important that you understand how 3D Secure impacts the customer transaction process.

1.
 1. After completing the checkout process, customers paying with Visa or MasterCard are prompted to enter their [Verified by Visa](#) or [MasterCard Secure Code](#)
 2. The customer is then either redirected to the issuer's website for authorization, or the authorization is completed within your payment solution.

If your customer is using an American Express, Discover card, or gift card, they are not prompted to enter a password.

If your customer is not enrolled in Verified by Visa or MasterCard Secure Code, the customer is prompted to enroll and create a password. The cardholder can decline registration, but when the customer reaches the maximum number of opt-out occurrences, the customer is no longer given the option to opt-out. The maximum number of opt-out occurrences is up to the credit card issuer, and the actions taken after the maximum is reached are also up to the credit card issuer.

It's important to note that as a merchant you can decide which transactions require 3D Secure authentication. For example, you might identify high-risk transactions with your payment solution's rules engine and then decide to further secure these transaction with 3D Secure.

What Are the Benefits of Using 3D Secure?

3D Secure offers merchants the following benefits:

1. **Liability shift:** As you know, when a chargeback occurs, you're typically liable for this transaction. With 3D Secure, the liability is shifted from you onto your issuing bank. Note: it's important you review all documentation provided by your issuer to ensure you understand the rules and regulations regarding 3D Secure implementation.
2. **Chargeback protection:** When you use Verified by Visa, you're guaranteed to never receive a chargeback. This protection helps secure you against friendly fraud or chargeback fraud. Note: MasterCard does not have a similar policy.
3. **Interchange benefits:** You may be able to access lower interchange fees and benefit from longer payment terms with your acquirer.
4. **Customer confidence:** As a CNP merchant, customer confidence is critical. Many customers will feel more confident knowing there is an extra level of security in place to protect their data.

What Should I Tell Customers about 3D Secure?

It's important to remember that while customers do want assurances that their transaction and data are secure, they also do not like extra confirmation steps and passwords. This means you must make sure you clearly explain that you are using 3D Secure (Verified by Visa or MasterCard Secure Code) to provide them with extra protection.

Provide some clear information in the early stages of the checkout process that explains the 3D Secure process in simple terms, to allow the customer to be prepared for the extra authentication step.

Remind the customer that this extra security is free of charge and that there aren't any additional fees hidden within the overall transaction. Provide a link that gives the customer more information about the program. Additionally, make sure the [Verified by Visa](#) or [MasterCard Secure Code](#) logo is clearly displayed.

How Can I Learn More about 3D Secure?

Read the documentation provided to you by your credit card issuer, and do not hesitate to ask them any questions you have.

Also, we suggest you browse the Verify website and learn more about [new device technology](#) and the future of [password authentication](#).

We welcome you to [contact us](#) with your questions about 3D Secure, and how to best ensure that your business and customers are protected from malicious and fraudulent activity.

5.8] Electronic Money

Electronic money is the currency that exists in banking computer systems that may be used to facilitate [electronic transactions](#). Although its value is backed by fiat currency and may be exchanged into a physical, tangible form, electronic money is primarily used for electronic transactions.

KEY TAKEAWAYS

- Electronic money is currency stored in banking computer systems.
- Electronic money is backed by fiat currency and the central banking system.
- Various companies allow for transactions with electronic money, such as Square or PayPal.

How eMoney Works

Electronic money is used for transactions globally and is commonly accessed through electronic banking systems and monitored through electronic processing. eMoney can be exchanged between individuals and businesses electronically using digital apps.

Electronic money is backed and controlled by [central banks](#). The [U.S. Federal Reserve](#) and 12 supporting banks manage the fiat currency in physical form and control the money supply through monetary policies and open market operations.

Central banks are exploring the use of a [Central Bank Digital Currency](#). In the United States, there is a physical currency issued by the Federal Reserve and digital balances held by commercial banks at the Federal Reserve. A CBDC would be a liability of the Federal Reserve, not of a commercial bank.¹

Electronic Payment Processing

Americans process transactions electronically by receiving pay checks through [direct deposits](#), moving money from one account to another via electronic fund transfers, or spending money with credit cards and debit cards.

The U.S. financial market has a robust infrastructure for transacting electronic money, which includes payment processing networks, such as Visa and MasterCard. Prepaid cards and digital wallets like [PayPal](#) and [Square](#) likewise allow users to deposit fiat currency for electronic money. Electronic money allows for immediate transactions through e-commerce.

eMoney vs. Cryptocurrency

Electronic money may be exchanged for fiat currency and is represented within the regulated banking system. [Cryptocurrencies](#) are privately issued and are not backed by financial assets. Cryptocurrency is created using block chain technology without government control or regulation.

What Is a Disadvantage of Using Electronic Money?

Fraud may be an issue when money can be transferred from one party to another without the necessity for verification of the original owner's true identity.

Is Cash Losing Popularity as the Use of eMoney Rises?

In 2022, only 18% of all payments were made with cash in the United States.²

What Is an Electronic Wallet or Digital Wallet?

A [digital wallet](#) or electronic wallet is a financial transaction application that runs on any connected device. It securely stores payment information and passwords in the cloud. Digital wallets may be accessible from a computer; [mobile wallets](#), which are a subset, are primarily used on mobile devices.

The Bottom Line

Electronic money or eMoney is currency stored in banking computer systems and backed by fiat currency. Individuals process and receive electronic money through paycheck direct deposits, electronic fund transfers, or online payments and purchases.

5.9] Email Security

E-mail Hacking

Email hacking can be done in any of the following ways:

- Spam
- Virus
- Phishing

#Spam

E-mail spamming is an act of sending **Unsolicited Bulk E-mails (UBE)** which one has not asked for. Email spams are the junk mails sent by commercial companies as an advertisement of their products and services.

#Virus

Some emails may incorporate with files containing malicious script which when run on your computer may lead to destroy your important data.

#Phishing

Email phishing is an activity of sending emails to a user claiming to be a legitimate enterprise. Its main purpose is to steal sensitive information such as usernames, passwords, and credit card details.

Such emails contains link to websites that are infected with malware and direct the user to enter details at a fake website whose look and feels are same to legitimate one.

E-mail Spamming and Junk Mails

Email spamming is an act of sending Unsolicited Bulk E-mails (UBI) which one has not asked for. Email spams are the junk mails sent by commercial companies as an advertisement of their products and services.

Spams may cause the following problems:

- It floods your e-mail account with unwanted e-mails, which may result in loss of important e-mails if inbox is full.
- Time and energy is wasted in reviewing and deleting junk emails or spams.
- It consumes the bandwidth that slows the speed with which mails are delivered.
- Some unsolicited email may contain virus that can cause harm to your computer.

Blocking Spams

Following ways will help you to reduce spams:

- While posting letters to newsgroups or mailing list, use a separate e-mail address than the one you used for your personal e-mails.
- Don't give your email address on the websites as it can easily be spammed.
- Avoid replying to emails which you have received from unknown persons.
- Never buy anything in response to a spam that advertises a product.

E-mail Clean-up and Archiving

In order to have light weighted Inbox, it's good to archive your inbox from time to time. Here I will discuss the steps to clean up and archive your Outlook inbox.

- Select **File** tab on the mail pane.
- Select **Cleanup Tools** button on account information screen.
- Select **Archive** from cleanup tools drop down menu.
- Select **Archive this folder and all subfolders** option and then click on the folder that you want to archive. Select the date from the **Archive items older than:** list. Click **Browse** to create new **.pst** file name and location. Click **OK**.

5.10] Wireless Application Protocol(WAP)

WAP is a collection of communication protocols meant to standardise how wireless devices, such as mobile phones and radio transceivers, may access the Internet and its services, such as email and newsgroups. Prior to the advent of WAP, Internet connection was available, but

various manufacturers utilised different technologies; WAP promised compatibility across these technologies.

WAP was created in 1997 at the WAP Forum by Ericsson, Motorola, Nokia, and Unwired Planet (now Phone.com). The WAP Forum was renamed the Open Mobile Alliance in 2002. (OMA).

Operation of the WAP:

WAP is an acronym for a collection of protocols. WAP hardware, such as WAP-enabled web browsers and network technologies, and WAP software, such as WAP-enabled web browsers, are designed to be compatible with this standard. These standards enhance mobile experiences that were previously limited by the capabilities of portable devices and wireless networks. WAP does this through the use of the following methods –

- Extensible Markup Language (XML), user datagram protocol (UDP), and Internet Protocol (IP) are efficient for wireless settings and are based on standards such as HTML, HTTP; the WML format for pages that may be sent via WAP.
- The WAP paradigm works in the same way as the traditional client-server model, but an additional WAP gateway acts as a middleman between the client and the server. This gateway transforms a micro browser's WAP device request into an HTTP URL request and sends it across the internet to the server. When the server answers, the WAP gateway analyses the response and sends the webpage as a WML file compatible with micro browsers to the WAP mobile device.

The Protocol Stack for WAP:

Here's a breakdown of the WAP standard's protocol stack that allows WAP devices to communicate with each other.

- In addition to WAE, WTP (Wireless Transaction Protocol) provides transaction functionality for server requests and answers.
- In order to protect privacy and data integrity, WTLS uses public-key cryptography.
- *Wireless Datagram Protocol* (WDP)– This protocol describes how data travels from the receiver to the sender and vice versa.

Why Would You Want to Use WAP?

WAP offered benefits to wireless network operators, content providers, and end-users when it was first presented in 1999.

- WAP was intended to improve existing wireless data services like voicemail while also allowing developers to create new mobile apps. These applications may be set up without the need for extra infrastructure or phone updates.
- WAP provides a market for third-party app developers to create new apps and mobile phone features. WML application development is introduced as an alternative programming language that allows developers to build effective mobile device apps.

5.11] Security In GSM

GSM is the most secured cellular telecommunications system available today. GSM has its security methods standardized. GSM maintains end-to-end security by retaining the confidentiality of calls and anonymity of the GSM subscriber.

Temporary identification numbers are assigned to the subscriber's number to maintain the privacy of the user. The privacy of the communication is maintained by applying encryption algorithms and frequency hopping that can be enabled using digital systems and signalling.

This chapter gives an outline of the security measures implemented for GSM subscribers.

Mobile Station Authentication

The GSM network authenticates the identity of the subscriber through the use of a challenge-response mechanism. A 128-bit Random Number (RAND) is sent to the MS. The MS computes the 32-bit Signed Response (SRES) based on the encryption of the RAND with the authentication algorithm (A3) using the individual subscriber authentication key (Ki). Upon receiving the SRES from the subscriber, the GSM network repeats the calculation to verify the identity of the subscriber.

The individual subscriber authentication key (Ki) is never transmitted over the radio channel, as it is present in the subscriber's SIM, as well as the AUC, HLR, and VLR databases. If the received SRES agrees with the calculated value, the MS has been successfully authenticated and may continue. If the values do not match, the connection is terminated and an authentication failure is indicated to the MS.

The calculation of the signed response is processed within the SIM. It provides enhanced security, as confidential subscriber information such as the IMSI or the individual subscriber authentication key (Ki) is never released from the SIM during the authentication process.

Signalling and Data Confidentiality

The SIM contains the ciphering key generating algorithm (A8) that is used to produce the 64-bit ciphering key (Kc). This key is computed by applying the same random number (RAND) used in the authentication process to ciphering key generating algorithm (A8) with the individual subscriber authentication key (Ki).

GSM provides an additional level of security by having a way to change the ciphering key, making the system more resistant to eavesdropping. The ciphering key may be changed at regular intervals as required. As in case of the authentication process, the computation of the ciphering key (Kc) takes place internally within the SIM. Therefore, sensitive information such as the individual subscriber authentication key (Ki) is never revealed by the SIM.

Encrypted voice and data communications between the MS and the network is accomplished by using the ciphering algorithm A5. Encrypted communication is initiated by a ciphering mode request command from the GSM network. Upon receipt of this command, the mobile station begins encryption and decryption of data using the ciphering algorithm (A5) and the ciphering key (Kc).

Subscriber Identity Confidentiality

To ensure subscriber identity confidentiality, the Temporary Mobile Subscriber Identity (TMSI) is used. Once the authentication and encryption procedures are done, the TMSI is sent to the mobile station. After the receipt, the mobile station responds. The TMSI is valid in the location area in which it was issued. For communications outside the location area, the Location Area Identification (LAI) is necessary in addition to the TMSI.

5.11] Security In 3G :

Mobile 3G network attacks are attacks that are specifically targeted at 3G mobile networks. 3G, or third-generation, mobile networks are the third generation of mobile communication technology and are used to provide wireless data and voice services to mobile devices. There are several types of attacks that can be launched against 3G mobile networks, including eavesdropping, man-in-the-middle attacks, denial of service (DoS) attacks, malware infections, and interference with communication signals. It's important for mobile network operators to implement security measures to protect against these types of attacks and keep their networks secure.

Types of 3G Mobile Attack

There are several types of attacks that can be launched against mobile networks –

- **Eavesdropping** – This type of attack involves intercepting and listening to the communication between two parties without their knowledge.
- **Man-in-the-middle attack** – In this type of attack, an attacker intercepts the communication between two parties and poses as one of the parties, allowing them to alter the communication and potentially steal sensitive information.
- **Denial of service (DoS) attack** – A DoS attack involves flooding a network or device with traffic, making it unavailable to legitimate users.
- **Malware** – Malware, or malicious software, can be used to compromise the security of a mobile network and steal sensitive information.
- **Interference** – Attackers can interfere with the communication between two parties by disrupting the signals used to transmit data.
- **Rogue base stations** – An attacker can set up a fake base station to intercept communication between mobile devices and the legitimate network.
- **SMS spoofing** – Attackers can send text messages from a fake or spoofed sender, potentially tricking users into divulging sensitive information or downloading malware.
- **Location tracking** – An attacker can track a mobile device's location by accessing the location data transmitted by the device.

It's important for mobile network operators to implement security measures to protect against these types of attacks and keep their networks secure.

How to secure Mobile 3G network traffic

There are several measures that can be taken to secure 3G mobile network traffic –

- **Encryption** – Encrypting the communication between devices and the network can help prevent eavesdropping and man-in-the-middle attacks.
- **Virtual private networks (VPNs)** – VPNs can be used to secure the communication between devices and the network by creating a secure, encrypted connection.
- **Firewalls** – Firewalls can be used to block unwanted traffic and protect against DoS attacks.
- **Antivirus software** – Antivirus software can be used to detect and prevent malware infections on mobile devices.
- **Network segmentation** – Segmenting the network into different sections can help prevent an attacker from gaining access to the entire network if they are able to compromise one section.
- **Regular software updates** – Keeping software up to date can help fix known vulnerabilities and prevent attacks.

- **User education** – Educating users on how to identify and avoid potential threats, such as phishing attacks, can help prevent them from inadvertently compromising the security of the network.

How to protect mobile data from network attacks

Here are some steps you can take to protect your mobile data from network attacks –

- Use a strong, unique password for your device and don't share it with anyone.
- Enable two-factor authentication whenever possible.
- Use a virtual private network (VPN) to encrypt your internet connection and secure your data when using public Wi-Fi.
- Keep your device's operating system and antivirus software up to date to fix known vulnerabilities.
- Avoid accessing sensitive information on public or untrusted networks.
- Be cautious when clicking on links or downloading attachments, as they could contain malware.
- Don't jailbreak or root your device, as doing so can remove important security measures.
- Consider using a mobile security app to provide additional protection for your device and data.

Unit 6 - User Authentication And kerberos

6.1] Introduction

User authentication verifies the identity of a user attempting to gain access to a network or computing resource by authorizing a human-to-machine transfer of credentials during interactions on a network to confirm a user's authenticity. The term contrasts with [machine authentication](#), which is an automated authentication method that does not require user input.

[Authentication](#) helps ensure only authorized users can gain access to a system by preventing unauthorized users from gaining access and potentially damaging systems, stealing information or causing other problems. Almost all human-to-computer interactions -- other than guest and automatically logged-in accounts -- perform a user authentication. It authorizes access on both wired and wireless networks to enable access to networked and internet-connected systems and resources.

A straightforward process, user authentication consists of three tasks:

1. **Identification.** Users have to prove who they are.
2. **Authentication.** Users have to prove they are who they say they are.
3. **Authorization.** Users have to prove they're allowed to do what they are trying to do.

User [authentication](#) can be as simple as requiring a user to type a unique identifier, such as a user ID, along with a [password](#) to access a system. It can also be more complex, however -- for example, requiring a user to provide information about physical objects or the environment or even take actions, such as placing a finger on a fingerprint reader.

User authentication methods

The main factors used in user authentication include the following:

- **Knowledge factors** include all things users must know in order to log in to gain access to a system. Usernames, IDs, passwords and personal identification numbers (PINs) all fall under this category.
- **Possession factors** consist of anything users must have in their possession in order to log in. This category includes one-time password tokens, [key fobs](#), [smartphone](#) apps, and employee ID cards.
- **Inherence factors** include characteristics inherent to individuals that confirm their identity. This category includes the scope of biometrics, such as [retina scans](#), fingerprint scans, [facial recognition](#) and Voice authentication.

❖ Kerberos :

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. In Kerberos Authentication server and database is used for client authentication. Kerberos runs as a third-party trusted server known as the Key Distribution Centre (KDC). Each user and service on the network is a principal. The main components of Kerberos are:

- **Authentication Server (AS):**
The Authentication Server performs the initial authentication and ticket for Ticket Granting Service.
- **Database:**
The Authentication Server verifies the access rights of users in the database.
- **Ticket Granting Server (TGS):**
The Ticket Granting Server issues the ticket for the Server

Kerberos Overview: Diag

- **Step-1:**
User login and request services on the host. Thus user requests for ticket-granting service.
- **Step-2:**
Authentication Server verifies user's access right using database and then gives ticket-granting-ticket and session key. Results are encrypted using the Password of the user.
- **Step-3:**
The decryption of the message is done using the password then send the ticket to Ticket Granting Server. The Ticket contains authenticators like user names and network

addresses.

- **Step-4:**
Ticket Granting Server decrypts the ticket sent by User and authenticator verifies the request then creates the ticket for requesting services from the Server.
- **Step-5:**
The user sends the Ticket and Authenticator to the Server.
- **Step-6:**
The server verifies the Ticket and authenticators then generate access to the service. After this User can access the services.

Kerberos Limitations

- Each network service must be modified individually for use with Kerberos
- It doesn't work well in a timeshare environment
- Secured Kerberos Server
- Requires an always-on Kerberos server
- Stores all passwords are encrypted with a single key
- Assumes workstations are secure
- May result in cascading loss of trust.
- Scalability

Is Kerberos Infallible?

No security measure is 100% impregnable, and Kerberos is no exception. Because it's been around for so long, hackers have had the ability over the years to find ways around it, typically through forging tickets, repeated attempts at password guessing (brute force/credential stuffing), and the use of malware, to downgrade the encryption.

Despite this, Kerberos remains the best access security protocol available today. The protocol is flexible enough to employ stronger encryption algorithms to combat new threats, and if users employ good password-choice guidelines, you shouldn't have a problem!

What is Kerberos Used For?

Although Kerberos can be found everywhere in the digital world, it is commonly used in secure systems that rely on robust authentication and auditing capabilities. Kerberos is used for Posix, Active Directory, NFS, and Samba authentication. It is also an alternative authentication system to SSH, POP, and SMTP.

Applications

- **User Authentication:** User Authentication is one of the main applications of Kerberos. Users only have to input their username and password once with Kerberos to gain

access to the network. The Kerberos server subsequently receives the encrypted authentication data and issues a ticket granting ticket (TGT).

- **Single Sign-On (SSO):** Kerberos offers a Single Sign-On (SSO) solution that enables users to log in once to access a variety of network resources. A user can access any network resource they have been authorized to use after being authenticated by the Kerberos server without having to provide their credentials again.
- **Mutual Authentication:** Before any data is transferred, Kerberos uses a mutual authentication technique to make sure that both the client and server are authenticated. Using a shared secret key that is securely kept on both the client and server, this is accomplished. A client asks the Kerberos server for a service ticket whenever it tries to access a network resource. The client must use its shared secret key to decrypt the challenge that the Kerberos server sends via encryption. If the decryption is successful, the client responds to the server with evidence of its identity.
- **Authorization:** Kerberos also offers a system for authorization in addition to authentication. After being authenticated, a user can submit service tickets for certain network resources. Users can access just the resources they have been given permission to use thanks to information about their privileges and permissions contained in the service tickets.
- **Network Security:** Kerberos offers a central authentication server that can regulate user credentials and access restrictions, which helps to ensure network security. In order to prevent unwanted access to sensitive data and resources, this server may authenticate users before granting them access to network resources.

6.2] Password Authentication Networks

A password is a secret word or phrase or code that you need to know in order to have access to a place or system. In technical terms, it is a series of letters or numbers that you must type into a computer or computer system in order to be able to use it. A password is a real-life implementation of challenge-response authentication (a set of protocols to protect digital assets and data).

A string of characters i.e letters, numbers, special characters, used to verify the identity of a user during the authentication process is known as password.

Password Management:

Since passwords are meant to keep the files and data secret and safe so it is prevented the unauthorized access, password management refers to the practices and set of rules or principles or standards that out must follow or at least try to seek help from in order to be a good/strong password and along with its storage and management for the future requirements.

Issues Related to Managing Passwords:

The main problem with password management is that it is not safe to use the same password for multiple sites, therefore having different passwords for different sites and on top of that remembering them is quite difficult. As per the statistics, more

than 65% of people reuse passwords across accounts and the majority do not change them, even after a known breach. Meanwhile, 25% reset their passwords once a month or more because they forgot them.

To escape from this situation people often tend to use password managers (A password manager is a computer program that allows users to store, generate, and manage their passwords for local applications and online services.). Password managers to a certain extent reduce the problem by having to remember only one “master password” instead of having to remember multiple passwords. The only problem with having a master password is that once it is out or known to an attacker, the rest of all the passwords become available.

The main issues related to managing passwords are as follows:

- Login spoofing
- Sniffing attack
- [Brute force attack](#)
- [Shoulder surfing attack](#)
- Data breach

Methods to Manage Password:

There are a lot of good practices that we can follow to generate a strong password and also the ways to manage them.

- **Strong and long passwords:** A minimum length of 8 to 12 characters long, also it should contain at least three different character sets (e.g., uppercase characters, lowercase characters, numbers, or symbols)
- **Password Encryption:** Using irreversible end-to-end encryption is recommended. In this way, the password remains safe even if it ends up in the hands of cybercriminals.
- **Multi-factor Authentication (MFA):** Adding some security questions and a phone number that would be used to confirm that it is indeed you who is trying to log in will enhance the security of your password.
- **Make the password pass the test:** Yes, put your password through some testing tools that you might find online in order to ensure that it falls under the strong and safe password category.
- **Avoid updating passwords frequently:** Though it is advised or even made mandatory to update or change your password as frequently as in 60 or 90 days.

6.3] Authentication Tokens :

Password is the most common way of authentication. However, they are not as cheap and simple as we think, especially from the organization’s point of view because they have to provide an id and password to each resource they use as well as they’ve to manage the passwords of so many people using their portal. Also, it becomes cumbersome for the users to remember the passwords of every website

they log into. So we have an alternative to the passwords and that is Authentication Tokens.

Authentication Token is an immensely useful alternative to passwords. It is a small device that generates a new random value every time it is used. This random value is used for authenticating the users.

Components of Authentication Token

- Processor
- LCD for displaying output (Liquid Crystal Display)
- Battery
- A small keypad for entering information
- A real-time clock

Every such device(authentication token) is pre-programmed with a unique number called **random seed or seed**. This seed ensures that the output generated by the authentication token (the device) is unique. An authentication token is an example of 2-factor authentication because the token itself is protected with some PIN.

Working of Authentication Token

1. Creation of Token

When an authentication token is created, the corresponding random seed is generated for the token by the authentication server. This seed is automatically used by the authentication token due to which the value of the seed is not known by the user. This seed is pre-programmed inside the token, as well as its entry is made against that user's record in the user database.

2. Use of Token

An authentication token automatically generates pseudo-random numbers, called one-time passwords or one-time passcode (these codes/passwords can be used only once). Once they are used, they cannot be reused. This one time password is basically a 4 digit PIN. Below are some important points to use this one time password.

1. The user being authenticated will enter his/her id and one-time password which goes to the server.
2. The server receives the seed corresponding to the user id from the user database, using a *seed-retrieval program*.
3. The server gives the seed and one-time passcode to a *Password Validation Program*.
4. This program checks if the one-time password and seed are related to each other.

3. Server responds

The server finally responds back with a suitable message based on the output (success/failure) of the previous step.

Types of Authentication Token:

Response/Challenge Token

1. User sends a login request by providing only his user id and not the one-time password.
2. Server checks whether the user id is valid. If it is not valid, it responds with an error message otherwise if it is valid, then the server creates a random challenge. Then sends the random challenge to the user.
3. User receives the random challenge. Open the authentication token using the PIN and keys in the random challenge using the small keypad.
4. The seed of the token encrypts the random challenge which is then entered by the user in the password section of the login request.
5. Server verifies the encrypted random challenge received by the user which can be done in two ways-
 - The server can decrypt the encrypted random challenge received from the user with the seed value for the user, which is available to the server via the user database. If this decryption matches the original random challenge available on the server, the authentication is successful.
 - The server can encrypt its own version of the random challenge, which was sent earlier to the user, with the seed for the user. If this encryption matches with the encrypted random challenge received from the user, the authentication is successful.

Time-Based Token

In a time-based token, the server need not send any random challenge to the user. The token need not have a keypad for entry. In fact, it uses time in place of a random challenge. The tokens automatically generate a password every 60 seconds and display the latest password on the LCD output for the user.

For generating the password, the time-based tokens use seed and current system time.

- When a user wants to log in he/she enters the password displayed on the LCD of the token and uses it to login along with their user id.
- The server receives the password and performs an independent cryptographic function on the user's seed value and the current system time to generate its version of the password. If the two values match, it considers the user as a valid one.

- Finally, the server sends an appropriate message back to the user based on the result of the previous step.

6.4] Certificate Based Authentication

- Certificate-based authentication (CBA) has been a staple of government agencies and other high security environments for decades, long before the invention of [FIDO U2F](#) and [FIDO2](#), mostly due to its reliability and effectiveness in physical environments.
- To this day, it remains a favourite amongst many security experts, and is still being deployed across countless industries for a variety of scenarios.
- A couple of notable examples of its ubiquity include the humble smart card often used for access to offices and other buildings, the widely used SSL/TLS protocol featured in web browsers, and CBA is even found in any implementation of Public-Key Infrastructure (PKI).
- Generally speaking, client certificate-based authentication refers to an end user's device proving its own identity by providing a digital certificate that can be verified by a server in order to gain access to a network or other resources.
- In a nutshell, certificate-based authentication (CBA) uses a digital certificate derived from cryptography to identify a user, device or machine, before granting access to an application, network or other resource. Unlike some authentication solutions that are targeted at humans, such as one time passwords (OTP) and biometrics, certificate-based authentication can be adopted for all endpoints, including servers, personal computers, e-passports and literally anything that may be classified under the Internet of Things (IoT).
- In general, client certificate-based authentication and other methods where the secret is never exposed to even the user, is preferable to password-based authentication. Username and password authentication is based only on what the user knows (the password), but certificate-based client authentication also leverages what the user has (the private key), which cannot be phished, guessed or socially engineered.

But it is also important to highlight some of the conditions that help to maintain this level of control:

- no unauthorized users have gained access to the private-key underlying the digital certificate
- the lifecycle of distributed certificates is properly managed, including registration, renewal and revocation
- the proper infrastructure is set up to support the sending and validation of certificates

Users must be especially vigilant in protecting private keys in addition to the physical security of their devices generally speaking, but parties responsible for the infrastructure to support CBA also have their part to play in the process.

How Does Certificate-Based Authentication Work?

A digital identity certificate is an electronic document used to prove private key ownership. Certificate-based authentication uses the information within said document to verify the user, device or machine, in contrast to the classic username and password combination which is strictly limited to verifying only those who are in possession, i.e. potentially not just the user who should have access. The increased sophistication of attacks, coupled with the exploding number of attack surfaces as more and more devices access the network, make this additional capability more necessary than ever.

A digital certificate contains a number of important details, depending on the standard in use. For example, the popular X.509 certificate includes the below elements:

- The public key
- The user or device's name
- The name of the Certificate Authority (CA) that issued the certificate
- The date from which the certificate is valid
- The expiry date of the certificate
- The version number of the certificate data
- A serial number

As shown above here are the relevant details of the certificate based authentication flow:

- Client will initiate connection to the server.
- The server will respond and provide the server's public certificate to the client.
- The client will perform some validation to make sure the server's public certificate is trusted.
- The server requests the certificate from the client.
- A nonce is signed by the client using the client's private key, and is returned to the server and also includes the client's public certificate.
- The server receives both the certificate and the signed nonce. Using the client's public certificate the server will verify that the nonce was signed by the client.
- The server checks that the certificate has not expired and that it also has not been revoked.
- If all the verifications have passed the server may map attributes of the certificate to a user in their system to identify and authenticate the user.

There are usually even additional layers of verification in addition to the above process, such as examining the signature of the CA that issued the certificate, and climbing that hierarchy of signatures all the way to the root CA, to verify that it is known and trusted by the server.

6.5] Biometric Authentication

Biometric authentication refers to a cyber security process that verifies a user's identity using their unique biological traits such as fingerprints, voices, retinas, and facial features. Biometric authentication systems store this information in order to verify a user's identity when that user accesses their account. This type of authentication is usually more secure than traditional forms of multi-factor authentication.

- **Types of Authentication Methods**

Biometric authentication refers to a cyber security process that verifies a user's identity using their unique biological traits such as fingerprints, voices, retinas, and facial features. Biometric authentication systems store this information in order to verify a user's identity when that user accesses their account. This type of authentication is usually more secure than traditional forms of multi-factor authentication.

Types of Authentication Methods

The following are a few common authentication methods used for network security designed to beat cybercriminals and some of the biometric authentication technologies below are ones that you might use daily.

Facial recognition: These systems use a person's unique facial features to identify them. It's used in a variety of places such as smartphones, credit card payments, and law enforcement.

Fingerprint Recognition: Fingerprint authentication uses a person's unique fingerprint to verify their identity. It can be used to secure everything from mobile devices to automobiles, even buildings, making it the most widespread biometric authentication technology.

Eye Recognition: Eye recognition uses the unique pattern of someone's iris or retina to identify them. Because this type of biometric authentication is harder to implement, it's less common than the other types of biometric authentication options. An iris scan requires an infrared light source, a camera that can see IR, and minimal light pollution in order to ensure accuracy. Although it poses its challenges, it is one of the most accurate biometric authentication systems available when those conditions are met. Eye recognition is generally used in situations where security is most critical such as nuclear research facilities, etc.

Voice Recognition: Voice recognition uses the tone, pitch, and frequencies that are unique to an individual to authenticate them. This is the most commonly used biometric to verify users when they contact a call center for customer service support (for example, online banking)

Retina/Iris Recognition: Retina or also known as iris recognition, uses the pattern of someone's iris or retina to identify them. This type of biometric authentication is less common as it is harder to implement. It requires the implementation of an infrared light source, a camera that can see IR, and minimal light pollution to ensure accuracy. However, it happens to be one of the most accurate biometric authentication methods when those conditions are met. So it's typically used in situations where security is most critical (nuclear research facilities, for instance).

Gait Recognition: Gait recognition authenticates using the way someone walks to identify them. Each person walks a little differently, so the way a person puts one foot in front of the other is an effective way to verify their identity. As of now, it's not a common form of authentication but it's expected to become more common as future forms of authentication become more popular.

Vein Recognition: Vein recognition uses the pattern of blood vessels in a person's hand or finger to identify them. This type of biometric authentication uses infrared light to map the veins under the skin in your hands or fingers. Vein recognition is extremely accurate, more than retina/iris recognition.

What is Multimodal Biometric Authentication?

First, we need to understand what a unimodal biometric authentication system is. Essentially it's a system that verifies only one distinct characteristic, e.g face, retina. However, this system is very susceptible to spoofing.

This is where multimodal biometric authentication comes into play. It's an approach in which various biometrics are checked during identity verification. This makes it harder for a malicious hacker to spoof.

An example of multimodal authentication: A hacker may be able to find a person's photo on the internet, which they then use to successfully trick a facial recognition system into thinking it's the actual user. If the system just had the one authentication, then the user's accounts would be hacked. However, if the system requires the user to provide additional authentication such as a video of the person saying their password, then the hacker is very unlikely to find it.

By combining physical and behavioral authentication, you can enhance your security posture. Even if a malicious actor manages to spoof a fingerprint, the system can detect change in behaviour and deny entry. E.g., their speed of interaction with the system may be slower than the real user, or they are using keyboard shortcuts that the real user never used.

- **The Benefits and Risks of Biometric Authentication**

Identity insurance: Biometric identification provides the answers to “something a person has and is” and helps verify identity. Biometric authentication ensures increased levels of assurance to end-users. Its sophisticated software lets providers know that a person is who they claim to be through a tangible, real-world trait. Even if a cyber attacker knew of a user’s password or the answer to their security question, there’s no way they’d be able to duplicate a fingerprint or iris scan.

Ease of Use: While biometric authentication is more on the technical side in regards to its internal process, it’s generally easy and quick from a user’s point of view. By using either a fingerprint scanner to unlock an account or facial recognition, you’re cutting down the amount of times you have to log in using a long password that has multiple special characters in it that you will likely end up forgetting. Apple does a great job with biometric authentication, both fingerprint and facial, in their devices.

Fraud Detection: Biometrics are almost nearly impossible to replicate. They’re hard to replicate and steal, and only have about 1 in 64 billion chances that your fingerprint will match up exactly with someone else’s. It’s highly unlikely that a hacker will be able to access anything that’s secured with biometrics.

Risks

Being Hackable: Biometrics can still be hacked. Businesses and governments that collect and store users’ personal data are under constant threat from hackers. If however, they are a victim of a data breach, biometric data is irreplaceable, and organizations need to treat users’ biometrics with care and caution.

Partial Matches: Most common biometric authentication methods rely on partial information to authenticate a user’s identity. For example, during the enrollment process of registering your fingerprint, it will take data from your entire print and convert it into data. During future authentication however, it will only take partial fingerprint data to verify your identity so it’s faster and quicker.

Fail to Recognize a Valid User: When you register for facial recognition, you are registering a specific angle, and expression to your face. However, because the system only has the data from the enrollment process, anytime a user wears glasses, makeup, or even smiling, the facial recognition has a hard time recognizing the user, which could make the login process difficult.

Bias: Facial recognition systems may not recognize persons of color or non-cisgender people as accurately. Many biometric systems have been trained primarily using white or white male photos. This incorporates in them an inherent bias that results in difficulty recognizing women and people of color. Poor implementation of technology or deliberate misuse can result in discrimination and exclusion. Without a proven identity proofing solution, cross-demographic performance can be unreliable.

Fears of Sharing Biometric Data: Is it acceptable for companies to sell or provide their biometric data to others, such as law enforcement, immigration enforcement, or repressive foreign governments. These privacy concerns have caused many US states to enact biometric information privacy laws. When biometrics are converted into data and stored, particularly in places or countries that have large surveillance measures, a user runs the risk of leaving a permanent digital record that can be potentially tracked by nefarious actors.

Data Storage: Wherever biometric data is stored, it must be stored securely. Biometric data cannot be reset like a password. If biometric data is hacked, then there's really nothing a user can do as they cannot change their fingerprint or iris.

6.6] Key Distribution center

A Key Distribution Center (KDC) is a crucial element in a network encryption system responsible for assigning secret keys to authorized users. It operates in a system such as Kerberos, providing authentication and secret keys for users and services. Its primary function is to [enhance](#) communication security by preventing unauthorized access to transmitted data.

Key Distribution Center (KDC) is a crucial part of many cryptographic systems that provide a mechanism for secure distribution of encryption keys. It serves as a trusted third party that authenticates users in a network.

KDC works in several steps. Primarily, it verifies the user's identity, then it creates session keys and securely transmits these keys to the communicating parties. This process ensures secure, encrypted communication. The most common example of a system that uses KDC is the Kerberos protocol. Kerberos is a network authentication protocol designed to provide strong authentication for client/server applications.

Importance

The Key Distribution Center (KDC) is a crucial component in cryptography, particularly within the context of network [security systems](#) that use protocols like Kerberos. It is responsible for generating, storing, and managing secret encryption keys, which are used for secure communication between network entities.

Without a functioning KDC, these entities can't securely exchange their [data](#) since they wouldn't have access to shared, mutually trustworthy keys. Therefore, the KDC provides an essential service by distributing these keys securely, ensuring that each party involved in a communication process can trust and verify the authenticity and

confidentiality of the transmitted data. Its existence, operation, and proper management significantly enhance the security integrity of a network or system.

Explanation

The Key Distribution Center (KDC) is a critical feature of network security systems, particularly those that employ symmetric key encryption such as the Kerberos protocol. Its primary function is to securely distribute encryption and decryption keys to users within a network, thereby facilitating secure communication and data transfers. This is essential as encryption keys need to be dynamically assigned and updated to members of a network to maintain secure transmissions.

Therefore, a KDC helps in reducing the vulnerability of a system to key theft or unauthorized access. To further illuminate on its use, when a user requests access to a particular system or service within the network, the KDC is responsible for verifying their identity, and if authenticated, providing the necessary keys to enable the user to securely engage with the requested service.

This process ensures that only authorized individuals gain access to relevant information and resources, and any data exchanged remains confidential and protected from security threats. Hence, a Key Distribution Center is at the heart of managing and securing digital identities within a network.

Examples

1. Kerberos Authentication System: Kerberos, originally developed at MIT, is a widely used example of Key Distribution Center (KDC) technology. It is used to verify the identity of users and network services. In this system, the KDC is the trusted third party that distributes symmetric encryption keys to users and service providers in a network. After initial authentication, Kerberos issues 'tickets' for subsequent access to different network resources, minimizing repeated exposure of user credentials.

2. Wi-Fi Protected Access 2 (WPA2): In the context of a Wi-Fi network, WPA2 uses a central Key Distribution Center mechanism called the "Authentication Server", which distributes cryptographic keys to other devices on the network. WPA2's protocol relies on the KDC to authenticate devices and keep the network secure from unauthorized access.

3. Microsoft Active Directory: Active Directory (AD), Microsoft's network management technology, incorporates a KDC as part of the system's security infrastructure. In an Active Directory framework, the KDC distributes encryption keys to users, computers, and services within the network, securing the transmission of sensitive data and maintaining the integrity of network communications.

Frequently Asked Questions (FAQ)

Q: What is a Key Distribution Center (KDC)?

A: A Key Distribution Center is a part of a cryptographic system intended to reduce the risks inherent in exchanging keys. KDCs are part of the Kerberos protocol for network authentication.

Q: How does a Key Distribution Center work?

A: A Key Distribution Center securely holds encryption keys and controls their distribution among parties involved in communication. When a user logs in, the KDC verifies their identity and issues a ticket-granting ticket (TGT), using which the user can obtain service tickets for various services.

Q: What is the main purpose of a KDC?

A: The main purpose of a Key Distribution Center is to provide a secure method of sharing cryptographic keys, reducing the risk of keys being intercepted and potentially misused.

Q: When would you use a Key Distribution Center?

A: KDCs are often used in large networks where secure communication is required. For example, it is a critical part of the Kerberos authentication protocol, commonly used in Windows Active Directory Networks.

Q: Is Key Distribution Center safe?

A: As long as the KDC server is secure, and the system is properly configured and managed, KDC offers a high level of safety. However, if an unauthorized person gains access to the KDC, it can be compromised.

Q: What is the relationship between a KDC and Kerberos?

A: Kerberos is a network authentication protocol and KDC is an integral part of its system responsible for authenticating users. The KDC issues incident-specific cryptographic keys when a logon session is initiated, allowing secure authentication throughout the session.

Q: What information is stored in a KDC?

A: A KDC holds a database of secret keys; each entity on the network — whether a user or a service — shares a unique secret key with the KDC.

6.7] Security Handshake Pitfalls :

Security handshake protocols in cryptography are crucial for establishing a secure communication channel between two parties. However, there are potential pitfalls and vulnerabilities that can be exploited by attackers. Here are some common security handshake pitfalls in cryptography:

1. **Man-in-the-Middle Attacks:**

- Attackers may intercept the handshake process, pretending to be one of the communicating parties. This can lead to the exchange of encryption keys with the attacker instead of the intended recipient.

2. **Weak Key Exchange Algorithms:**

- If the key exchange algorithm used during the handshake is weak or vulnerable, it can be exploited by attackers to compromise the confidentiality of the communication.

3. **Insufficient Authentication:**

- Weak or inadequate authentication mechanisms during the handshake can allow attackers to impersonate one of the parties, leading to unauthorized access or data manipulation.

4. **Lack of Forward Secrecy:**

- Without forward secrecy, if a long-term key is compromised, all past communications encrypted with that key become vulnerable to decryption. Implementing forward secrecy helps mitigate this risk.

5. **Insecure Cipher Suites:**

- Choosing insecure or out dated cipher suites during the handshake can expose the communication to attacks. It's important to use strong, well-vetted encryption algorithms and avoid deprecated protocols.

6. **Certificate Issues:**

- If certificates are not properly validated during the handshake, it opens the door to various attacks. This includes issues such as expired certificates, incorrect certificate chains, or certificates signed by untrusted entities.

7. **Timing Attacks:**

- Timing attacks involve exploiting variations in the time taken to perform cryptographic operations. Poorly implemented algorithms or protocols may leak information through timing, compromising the security of the handshake.

8. **Denial of Service (DoS) Attacks:**

- Handshake processes can be targeted for DoS attacks. Attackers may flood the system with handshake requests, overwhelming resources and causing service disruptions for legitimate users.

9. **Insecure Random Number Generation:**

- Secure cryptographic protocols rely on strong random numbers for key generation. If the random number generator is weak or predictable, it can lead to the generation of weak keys, compromising the security of the handshake.

10. **Implementation Flaws:**

- Bugs or vulnerabilities in the implementation of cryptographic protocols can introduce weaknesses in the handshake process. Regular code audits and security testing are essential to identify and address these issues.

6.8] Single Sign on (SSO)

• Role of SSO :

The role of Single Sign-On (SSO) is to provide a seamless and convenient authentication experience for users while improving security and reducing administrative overhead for organizations.

- **Enhanced user experience:** Users can access multiple applications and systems with a single set of credentials, simplifying the login process and improving productivity.
- **Improved security:** SSO reduces the reliance on weak passwords and enables centralized access control, enforcing strong security measures consistently across systems.
- **Centralized authentication and access control:** SSO centralizes the authentication process, allowing organizations to manage user access and authentication policies in one place
- **Streamlined user provisioning and deprovisioning:** SSO simplifies the management of user access privileges, making it easier to on board and off board users.
- **Audit and compliance:** SSO provides better visibility and control over user access, facilitating monitoring, auditing, and compliance with regulatory requirements.

Script-Based SSO:

In script-based SSO, authentication and authorization are managed through scripts or code embedded within applications.

- Users authenticate themselves individually to each application, and the script within each application verifies the user's credentials.
- The script then establishes a session for the user, allowing them to access other applications without requiring further authentication.
- Example: An organization uses script-based SSO to enable its employees to access different internal systems such as email, document management, and HR portals. Once authenticated to one system, the user can seamlessly access other systems without entering credentials again.

Agent-Based SSO:

- Agent-based SSO relies on a dedicated software agent installed on the user's device or workstation.
- The agent intercepts authentication requests from applications and handles the authentication process on behalf of the user.
- Upon successful authentication, the agent provides a security token to the application, allowing the user to access multiple applications without additional login prompts.
- Example: A user logs into their computer and launches a web browser. The agent installed on the computer automatically authenticates the user to the browser, and subsequent requests to various web applications are handled by the agent, providing a seamless single sign-on experience.

Unit 1: Introduction to Big Data

Business Intelligence (BI)

Definition

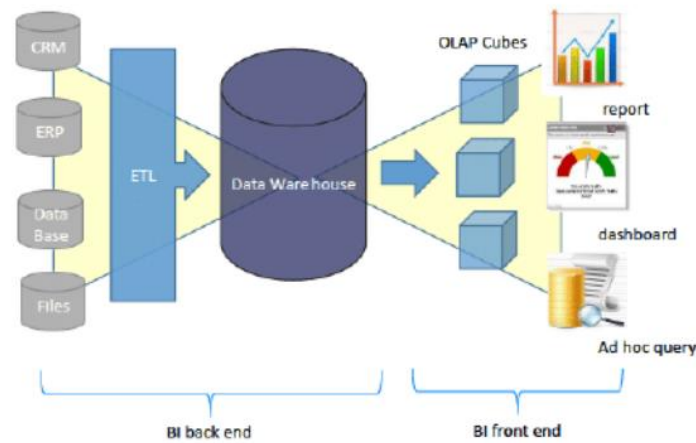
Business Intelligence (BI) is a technology-driven process for analyzing data and delivering actionable insights to help businesses make informed decisions. It combines tools, methodologies, and practices to transform raw data into meaningful information, enabling stakeholders to identify trends, uncover opportunities, and solve business challenges effectively.

Key Components of BI

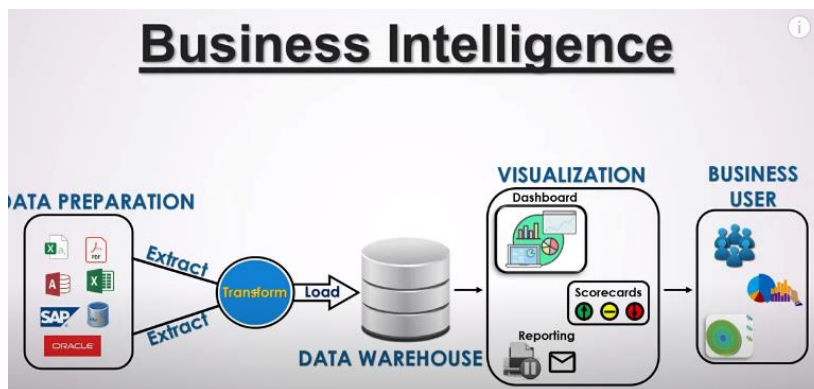
1. **Data Sources:**
BI collects data from diverse sources, such as:
 - **Internal systems:** ERP (Enterprise Resource Planning) systems, CRM (Customer Relationship Management) systems.
 - **External systems:** Market data, social media analytics, industry reports.
 2. **Data Integration and Transformation (ETL):**
 - **Extract:** Data is pulled from various sources.
 - **Transform:** Data is cleaned, standardized, and formatted for analysis.
 - **Load:** Processed data is loaded into a central repository, typically a data warehouse.
 3. **Data Warehousing:**
A central repository that stores integrated, historical data for querying and analysis.
 4. **Analytical Tools:**
These tools analyze the data and present insights. Examples include:
 - **Dashboards:** Visualize key performance indicators (KPIs).
 - **Reports:** Detailed summaries of business performance.
 - **Data Mining:** Uncovers hidden patterns and correlations in data.
 5. **Presentation Layer:**
Insights are presented in a user-friendly way through:
 - Interactive dashboards.
 - Visualizations such as bar charts, pie charts, and heat maps.
-

Benefits of BI

- **Improved Decision-Making:** Provides data-driven insights to make strategic and tactical decisions.
- **Enhanced Efficiency:** Automates reporting processes and reduces manual work.
- **Customer Understanding:** Helps analyze customer behavior and preferences.
- **Competitive Advantage:** Identifies market trends and competitor performance.
- **Real-Time Monitoring:** Dashboards provide real-time tracking of key metrics.



1: Business Intelligence Architecture



Business Intelligence in Action: Real-Time Example

Industry: Retail

Example: Walmart's BI System

- **Challenges:** Walmart, one of the largest global retailers, needed a system to manage its massive volume of transactional data to optimize operations and enhance customer satisfaction.
- **Implementation:**
 - Walmart implemented BI tools to collect data from millions of daily transactions across thousands of stores worldwide.
 - Data was stored in a centralized data warehouse for analysis.
- **Usage:**
 1. **Inventory Management:** BI tools track real-time sales and stock levels, allowing Walmart to predict demand and ensure optimal inventory.
 2. **Customer Insights:** By analyzing purchase patterns, Walmart identifies trending products and tailors promotions accordingly.
 3. **Operational Efficiency:** Store performance is monitored in real time, enabling Walmart to address issues such as underperforming locations or supply chain delays.
- **Outcomes:**
 - Reduced inventory costs by optimizing supply chain logistics.
 - Increased customer satisfaction through personalized offers and promotions.
 - Improved profitability by identifying and focusing on high-margin products.

Tools for BI

- **Tableau:** Interactive dashboards and visualizations.
 - **Microsoft Power BI:** Intuitive analysis with integration into the Microsoft ecosystem.
 - **QlikView:** Self-service analytics and discovery.
 - **Looker:** Advanced analytics for data exploration.
-

Emerging Trends in BI

1. **AI and Machine Learning:** Enhancing predictive analytics and automating data insights.
 2. **Self-Service BI:** Empowering non-technical users to explore data without relying on IT teams.
 3. **Cloud BI:** Scalable and accessible analytics hosted on cloud platforms.
 4. **Mobile BI:** Real-time access to analytics via smartphones and tablets.
-

OR

Business Intelligence is a collection of procedures, mechanisms, and technologies that modifies raw data into significant information that drives cost-effective business services.

-It is a collection of software and services to modify data into actionable intelligence and recognition. **Business Intelligence** (BI) has an unambiguous impact on an organization's methods, tactical and operational business decisions. BI supports fact-based decision-making using historical data rather than assumptions and gut feelings.

-BI tools implement data analysis and make documents, summaries, dashboards, maps, graphs, and charts to support users with detailed intelligence about the feature of the business.

-Business Intelligence is one of the most dynamic tools many organizations use to know their user base and industry better. It defines the business methodology in which the raw information is transformed into useful data which support decision making.

-Business intelligence has wide software, and if talking about the advantage of business intelligence in the retail sector, current business intelligence tools allow organizations to take benefit of information not only to consider current sales but also to estimate future potential, patterns, trends and understand the demand of the user on a deeper level.

Methods of Business Intelligence:

The methods of business intelligence are as follows –

Data analysis visualization – Data analysis visualization is all about how it visualizes the data. It presents records on dashboards and uses customized metrics associated with the business to create better decisions based on facts.

Reporting – Business intelligence tools are used for reporting information gathering from all the sources and processing it to enable better reporting and financial decision making with a rational mind.

Predictive Analytics – Predictive analytics is that how do you learn an action will work? The fact is you don't learn, and if you learn, not 100 percent. However, with business intelligence, it can make an evidence-based decision to drive business further. Business intelligence allows us to create a reasonable prediction of the current trends and user behaviors that impact the organization's complete development.

Data Mining – Data mining is a computer-supported technique to reveal previously anonymous or unnoticed relations between data entities. Data mining is the procedure of discovering useful new correlations, designs, and trends by sharing through a high amount of data saved in the warehouse, using pattern recognition technologies such as statistical and numerical approaches.

2. Decision Support Systems (DSS)

Definition

A Decision Support System (DSS) is an interactive, computer-based system designed to assist decision-makers in solving semi-structured or unstructured problems by providing data, models, and tools for analysis. Unlike traditional systems that automate routine tasks, a DSS focuses on improving decision quality by offering insights and alternatives.

Key Characteristics of DSS

1. **Support for Decision-Making:**
 - Assists in complex decision-making processes where data analysis is critical.
 - Provides recommendations, forecasts, or insights rather than automating decisions.
2. **Interactive:**
 - Users can input scenarios, change parameters, and view the outcomes dynamically.
3. **Flexible:**

- Adaptable to a variety of business problems and industries.
 - 4. **Data-Driven and Model-Driven:**
 - Combines raw data with analytical models to provide deeper insights.
 - 5. **User-Focused:**
 - Designed with a user-friendly interface to enable non-technical users to access and utilize the system effectively.
-

Components of DSS

1. **Database Management System (DBMS):**
 - Stores and organizes data required for decision-making.
 - Example: Historical sales data, inventory levels, or weather reports.
 2. **Model Management System (MMS):**
 - Provides tools to create, modify, and apply analytical models.
 - Models include statistical analysis, optimization, forecasting, and simulation.
 3. **User Interface (UI):**
 - Allows decision-makers to interact with the system, visualize data, and test scenarios.
 4. **Knowledge Base:** Contains domain-specific rules, facts, and insights to guide decisions.
-

Types of DSS

1. **Data-Driven DSS:**
 - Focuses on accessing and manipulating large datasets for analysis.
 - Example: Business performance dashboards.
 2. **Model-Driven DSS:**
 - Utilizes mathematical, statistical, or simulation models to support decision-making.
 - Example: Optimization of delivery routes in logistics.
 3. **Knowledge-Driven DSS:**
 - Relies on expert systems and artificial intelligence to provide recommendations.
 - Example: Clinical decision support in healthcare.
 4. **Communication-Driven DSS:**
 - Focuses on collaboration among multiple stakeholders.
 - Example: Video conferencing tools integrated with data sharing.
-

Benefits of DSS

- **Improves Decision Quality:** Provides comprehensive insights and data-driven alternatives.
 - **Saves Time:** Streamlines analysis and automates parts of the decision-making process.
 - **Enhances Efficiency:** Helps manage complex problems and reduces errors.
 - **Promotes Collaboration:** Encourages teamwork through shared data and models.
-

Real-Time Example: Agricultural Decision Support System

Scenario:

Farmers face challenges such as unpredictable weather, market price fluctuations, and pest outbreaks. A DSS tailored for agriculture helps optimize decisions.

Implementation:

An agricultural DSS integrates weather data, soil conditions, crop models, and market trends to support farmers.

Usage:

1. **Crop Selection:**
 - Suggests the best crops to plant based on soil type, weather patterns, and historical yield data.
 - **Example:** A farmer in India uses DSS to choose drought-resistant crops during a low rainfall season.
2. **Irrigation Management:**
 - Monitors soil moisture levels and weather forecasts to recommend optimal irrigation schedules.
 - **Example:** DSS alerts the farmer to reduce irrigation ahead of expected rainfall, conserving water.
3. **Market Forecasting:**
 - Analyzes market trends to predict crop demand and price fluctuations.
 - **Example:** The system advises delaying the sale of wheat until prices peak in the next quarter.

Outcomes:

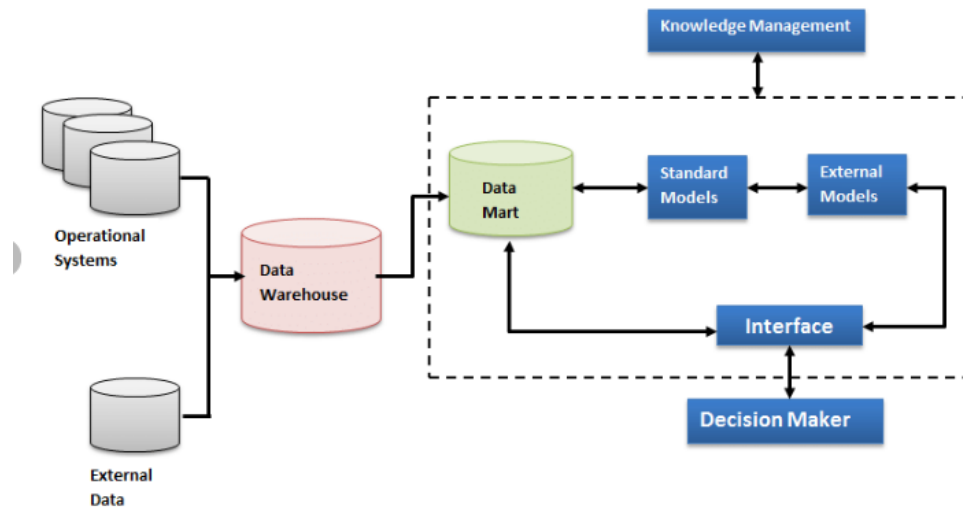
- Increased crop yield by making data-driven planting and harvesting decisions.
 - Reduced costs through efficient resource management (e.g., water, fertilizers).
 - Improved profitability by targeting high-demand crops at the right time.
-

Other Real-Time DSS Examples

1. **Healthcare DSS:**
 - Used in hospitals to assist doctors with diagnosis and treatment plans.
 - Example: A clinical DSS suggests treatments based on a patient's symptoms and medical history, reducing the likelihood of errors.
2. **Logistics DSS:**
 - Optimizes delivery routes to save time and fuel.
 - Example: A courier company uses DSS to reroute deliveries dynamically based on traffic data.
3. **Retail DSS:**
 - Tracks inventory levels and predicts demand to avoid stockouts or overstocking.
 - Example: Walmart uses DSS to optimize restocking based on past sales trends and customer behavior.

A DSS bridges the gap between raw data and actionable decisions, offering a competitive edge in today's data-driven landscape. Would you like further insights or examples tailored to a specific industry?

3. Data Warehousing



Decision Support System (DSS) architecture

A **data warehouse** is a centralized repository designed to store large volumes of structured and semi-structured data from multiple sources. It is optimized for querying and analysis rather than transaction processing. The primary goal is to support business intelligence (BI) activities, such as reporting, analysis, and decision-making.

Characteristics of a Data Warehouse

1. **Subject-Oriented:**
 - Focuses on specific business areas like sales, finance, or customer service rather than day-to-day operations.
 - Example: A sales data warehouse might consolidate data on customer purchases, regional sales, and product performance.
2. **Integrated:**
 - Combines data from various sources (e.g., databases, ERP systems, and third-party sources) into a consistent format.
 - Integration involves cleaning, transforming, and standardizing data to ensure uniformity.
3. **Time-Variant:**
 - Stores historical data to allow trend analysis over time.
 - Example: A marketing data warehouse retains campaign performance data from the past five years to analyze seasonal trends.
4. **Non-Volatile:**
 - Once data is entered into the warehouse, it does not change. This stability ensures that historical analysis is accurate.

- Example: A transaction record from five years ago remains intact for future reference.
-

Components of a Data Warehouse

1. **Data Sources:**
 - Include operational databases (e.g., CRM, ERP), web logs, and external data providers.
 2. **ETL Process (Extract, Transform, Load):**
 - **Extract:** Data is gathered from various sources.
 - **Transform:** Data is cleaned and formatted to ensure consistency.
 - **Load:** Processed data is loaded into the data warehouse.
 3. **Data Storage:**
 - Data is organized in schemas (star schema, snowflake schema) or tables optimized for querying.
 4. **Metadata:**
 - Provides information about the data (e.g., source, format, relationships) to help users understand its context.
 5. **Access Tools:**
 - Enable users to query and visualize data through dashboards, reports, or analytical tools.
-

Benefits of Data Warehousing

1. **Single Source of Truth:**
 - Consolidates data from multiple systems, providing a consistent and reliable data source.
 2. **Efficient Querying:**
 - Optimized for running complex queries and large-scale analyses without impacting operational systems.
 3. **Improved Decision-Making:**
 - Enables historical analysis and trend forecasting for informed decision-making.
 4. **Scalability:**
 - Modern data warehouses can scale to accommodate growing data volumes and increasing user demands.
-

Data Warehousing in Action: Real-Time Example

Industry: Retail

Example: Amazon's Data Warehouse System

- **Challenges:**
 - Amazon needed a system to manage its massive data volume generated by millions of daily transactions, customer preferences, and supplier interactions.

- Data from operational systems was siloed, making it difficult to perform cross-functional analysis.
 - **Implementation:**
 - Amazon built a scalable data warehouse that integrates data from various sources, including website logs, customer transactions, and inventory databases.
 - The ETL process ensures data consistency and quality before it is loaded into the warehouse.
 - **Usage:**
 1. **Personalized Recommendations:**
 - Amazon's data warehouse analyzes customer browsing and purchasing patterns to recommend products.
 2. **Inventory Management:**
 - Tracks inventory levels and predicts restocking needs to avoid stockouts or overstocking.
 3. **Pricing Strategies:**
 - Analyzes competitor pricing, customer demand, and seasonal trends to dynamically adjust prices.
 - **Outcomes:**
 - Enhanced customer experience through personalized interactions.
 - Optimized operations, reducing inventory costs and delivery delays.
 - Increased revenue by targeting promotions and adjusting prices effectively.
-

Emerging Trends in Data Warehousing

1. **Cloud Data Warehousing:**
 - Platforms like Snowflake, Amazon Redshift, and Google BigQuery offer scalable and cost-effective cloud-based solutions.
 - Example: A startup can scale its data warehouse without investing heavily in on-premise infrastructure.
 2. **Real-Time Data Warehousing:**
 - Incorporates real-time data streams to enable instant analysis and decision-making.
 - Example: A stock trading platform uses real-time warehousing to analyze market trends and execute trades promptly.
 3. **AI-Powered Data Warehousing:**
 - Leverages machine learning to automate data cleaning, integration, and anomaly detection.
 4. **Data Lakes vs. Warehouses:**
 - **Data Lakes** store raw, unstructured data.
 - **Data Warehouses** store structured, processed data for analysis.
-

Other Real-Time Examples

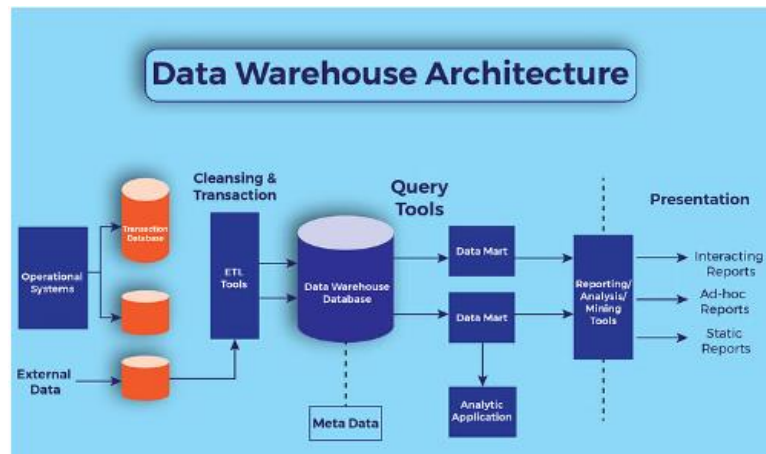
1. **Healthcare:**
 - Hospitals use data warehouses to integrate patient records, lab results, and treatment histories for predictive analytics and personalized care.
 - Example: A healthcare provider analyzes patient readmission rates to improve care quality.

2. Telecommunications:

- Telecom companies analyze customer call records and network usage to detect fraud and optimize network performance.
- Example: Verizon uses a data warehouse to monitor network performance and predict service outages.

3. Banking and Finance:

- Financial institutions analyze transaction data to detect fraudulent activities and offer personalized financial products.
- Example: A bank uses historical loan data from its warehouse to refine credit risk models.



4. Definition of Big Data

Definition:

Big data refers to extremely large datasets that traditional methods cannot efficiently handle due to their size, complexity, and rapid generation.

Sources of Big Data:

- **Transactional Data:** Generated by business transactions (e.g., sales records).
- **Sensor Data:** From IoT devices like smart meters or GPS trackers.
- **Social Media Data:** Posts, likes, comments, and shares.
- **Logs:** Server logs, clickstreams, and website activity.

Example:

Streaming platforms like Netflix analyze massive amounts of user activity data to personalize recommendations.

Characteristics of Big Data (5 Vs)

Big Data is defined by specific characteristics that make it challenging and unique to manage using traditional data processing methods. The five primary characteristics are:

1. Volume

- **Explanation:**
Volume refers to the enormous size of data generated every second. The data is typically measured in terabytes (TB), petabytes (PB), or even exabytes (EB). Traditional storage solutions are inadequate for managing such massive datasets.
 - **Real-Time Example:**
Social media platforms like Facebook generate over 500 terabytes of data daily from user posts, photos, videos, and interactions.
-

2. Velocity

- **Explanation:**
Velocity is the speed at which data is generated, collected, and processed. Some applications, like real-time analytics, demand instantaneous or near-instantaneous processing of streaming data.
 - **Real-Time Example:**
In financial trading, stock market data is generated and processed in milliseconds to enable algorithmic trading. Streaming services like Netflix also process real-time viewing data to provide personalized recommendations.
-

3. Variety

- **Explanation:**
Variety refers to the different types of data formats. Big data comes in:
 - **Structured Data:** Organized data in rows and columns (e.g., databases).
 - **Unstructured Data:** Data without a predefined format (e.g., videos, emails).
 - **Semi-Structured Data:** Partially organized data (e.g., JSON files, XML).
 - **Real-Time Example:**
E-commerce platforms like Amazon collect structured data (customer purchases), unstructured data (product reviews), and semi-structured data (JSON logs from website activity).
-

4. Veracity

- **Explanation:**
Veracity addresses the quality, reliability, and accuracy of the data. Poor-quality data with errors, biases, or inconsistencies can lead to incorrect insights and decisions.

- **Challenges:**
 - Managing incomplete or noisy data.
 - Resolving conflicts between different data sources.
 - **Real-Time Example:**

In healthcare, patient data may come from various sources (hospital records, wearable devices, and surveys). Ensuring this data is accurate and consistent is critical for effective diagnosis and treatment planning.
-

5. Value

- **Explanation:**

Value refers to the ability to extract meaningful and actionable insights from data. Without deriving value, big data is just a storage and processing burden.
 - **Real-Time Example:**

Retail chains like Walmart use big data analytics to optimize inventory management, predict customer purchasing behavior, and develop targeted marketing campaigns. This enhances customer satisfaction and increases revenue.
-

Interplay of Characteristics in Real-Time Example:

Example: Smart Cities

- **Volume:** Sensors in smart cities generate massive data about traffic, air quality, and energy usage.
 - **Velocity:** Traffic monitoring systems must process data in real-time to manage signals dynamically and reduce congestion.
 - **Variety:** Data includes structured formats (vehicle counts), unstructured formats (video footage), and semi-structured formats (sensor logs).
 - **Veracity:** Ensuring data accuracy from thousands of devices is critical to prevent faulty decisions like incorrect traffic routing.
 - **Value:** The processed data helps improve urban planning, reduce pollution, and enhance citizens' quality of life.
-

6. Considerations for Big Data

- **Scalability:** Use scalable technologies (e.g., cloud computing) to manage growing datasets.
- **Data Privacy:** Protect sensitive information and adhere to regulations like GDPR or HIPAA.
- **Processing Power:** Invest in high-performance computing systems to process data efficiently.
- **Cost:** Manage the costs associated with storage, processing, and analysis of big data.

7. Introduction to Hadoop

Definition

Hadoop is an open-source framework developed by the Apache Software Foundation that allows for distributed storage and processing of large datasets across clusters of computers. It is designed to handle massive amounts of structured, semi-structured, and unstructured data with high scalability, fault tolerance, and cost efficiency.

Core Components of Hadoop

1. HDFS (Hadoop Distributed File System):

- **Purpose:** A distributed file system designed to store large datasets across multiple nodes.
- **Features:**
 - Breaks data into blocks (default size: 128 MB) and stores them on different nodes.
 - Provides fault tolerance by replicating data blocks (default replication: 3).
- **Example:** A 1 GB file is divided into eight 128 MB blocks and distributed across different nodes in the cluster.

2. YARN (Yet Another Resource Negotiator):

- **Purpose:** Manages resources in the Hadoop cluster and schedules tasks.
- **Features:**
 - Dynamically allocates resources for various applications.
 - Enables multiple data processing engines (e.g., Spark, Hive) to run simultaneously.

3. MapReduce:

- **Purpose:** A programming model for processing and generating large datasets in parallel.
- **Features:**
 - **Map Function:** Processes input data and converts it into key-value pairs.
 - **Reduce Function:** Aggregates and processes the key-value pairs to generate final output.
- **Example:** Counting the occurrences of words in a document. The **Map** function generates a count for each word, and the **Reduce** function aggregates these counts.

4. Hadoop Ecosystem Tools:

- **Hive:** A data warehouse tool for querying and analyzing data using SQL-like syntax.
 - **Pig:** A scripting platform for processing and analyzing large datasets.
 - **HBase:** A NoSQL database for real-time read/write access.
 - **Sqoop:** A tool for importing/exporting data between Hadoop and relational databases.
 - **Flume:** Used for collecting and ingesting streaming data into Hadoop.
-

Key Features of Hadoop

1. **Scalability:**
 - Can scale horizontally by adding more commodity hardware to the cluster.
 2. **Fault Tolerance:**
 - Automatically replicates data across nodes, ensuring no data is lost if a node fails.
 3. **Cost-Effective:**
 - Uses inexpensive commodity hardware instead of specialized high-end servers.
 4. **Flexibility:**
 - Capable of handling diverse data types (text, images, videos, logs).
 5. **Distributed Computing:**
 - Enables parallel processing of data, significantly speeding up large-scale computations.
-

Benefits of Hadoop

1. **Handling Big Data:**
 - Designed to process massive datasets that traditional systems cannot handle efficiently.
 2. **Processing Efficiency:**
 - Processes data in parallel across multiple nodes, reducing computation time.
 3. **Integration:**
 - Works seamlessly with other big data tools like Spark, Kafka, and NoSQL databases.
 4. **Community Support:**
 - Large and active community constantly improving the framework.
-

Real-Time Example: Hadoop in Action

Industry: E-commerce

Example: How Amazon Uses Hadoop

- **Challenges:**
 - Amazon handles vast amounts of transactional data, customer reviews, clickstreams, and product inventory across its global platform. Traditional systems were insufficient for storing and analyzing this massive data efficiently.
- **Implementation:**
 - Amazon uses Hadoop to store and process customer interactions, purchasing patterns, and inventory data in a distributed environment.
 - **HDFS** stores raw data (e.g., clickstream logs, transaction records).
 - **MapReduce** processes this data to analyze customer behavior, detect purchasing trends, and optimize logistics.
- **Usage:**
 1. **Personalized Recommendations:**
 - Hadoop processes billions of data points from past purchases, product views, and search queries to suggest personalized products for each user.
 2. **Inventory Management:**

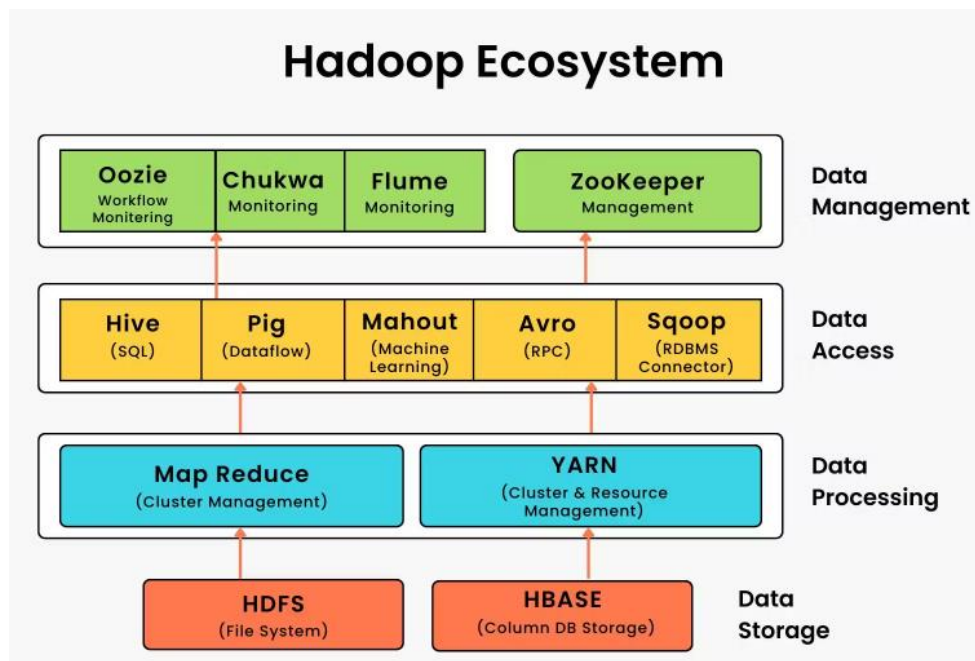
- Analyzes sales patterns to predict demand and manage stock levels efficiently.
 - 3. **Fraud Detection:**
 - Detects suspicious patterns in transaction logs to flag potential fraud cases.
 - **Outcomes:**
 - Increased customer satisfaction through personalized shopping experiences.
 - Reduced operational costs by optimizing inventory and supply chain processes.
 - Improved security through real-time fraud detection.
-

Other Real-Time Examples

1. **Healthcare:**
 - Hospitals use Hadoop to process and analyze large volumes of patient records, genomic data, and medical imaging.
 - **Example:** A research center uses Hadoop to identify genetic markers associated with diseases by analyzing terabytes of genomic data.
 2. **Telecommunications:**
 - Telecom companies process call records, network usage, and customer feedback to improve service.
 - **Example:** Verizon uses Hadoop to analyze network data and predict service outages.
 3. **Finance:**
 - Banks analyze transaction data to detect fraudulent activities and optimize investment strategies.
 - **Example:** A bank uses Hadoop to process millions of transactions daily to identify suspicious activity patterns.
 4. **Entertainment:**
 - Streaming platforms like Netflix analyze user viewing habits to recommend content.
 - **Example:** Hadoop processes clickstream data to identify trending shows and improve content recommendations.
-

Emerging Trends in Hadoop

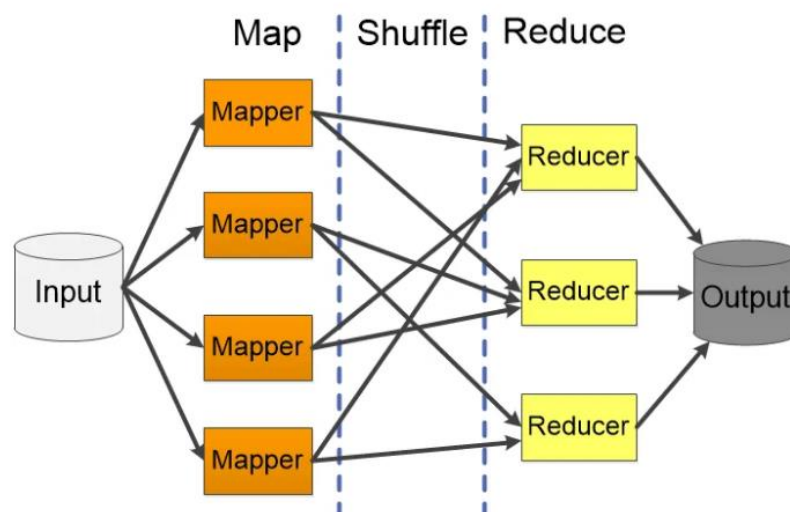
1. **Integration with Cloud:**
 - Hadoop is increasingly being integrated with cloud platforms like AWS, Azure, and Google Cloud for scalability and cost efficiency.
2. **Real-Time Processing:**
 - Tools like Apache Kafka and Spark are complementing Hadoop for real-time data processing.
3. **AI and ML Integration:**
 - Machine learning frameworks are being integrated with Hadoop to derive predictive insights.
4. **Hadoop-as-a-Service (HaaS):**
 - Managed Hadoop services reduce the complexity of setting up and maintaining clusters.



****Map Reduce:**

A MapReduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce: Simplified Data Processing on Large Clusters," published by Google.

The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.



The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes.

Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial.

But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change.

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
 - **Map stage** – The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
 - **Reduce stage** – This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

Terminology:

- **PayLoad** – Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** – Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** – Node that manages the Hadoop Distributed File System (HDFS).
- **DataNode** – Node where data is presented in advance before any processing takes place.
- **MasterNode** – Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** – Node where Map and Reduce program runs.
- **JobTracker** – Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** – Tracks the task and reports status to JobTracker.
- **Job** – A program is an execution of a Mapper and Reducer across a dataset.
- **Task** – An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** – A particular instance of an attempt to execute a task on a SlaveNode.

Unit 2: Big Data Analytics

2.1) Big Data Analytics

Big Data Analytics is all about crunching massive amounts of information to uncover hidden trends, patterns, and relationships. It's like sifting through a giant mountain of data to find the gold nuggets of insight.

Here's a breakdown of what it involves:

- **Collecting Data:** Such data is coming from various sources such as social media, web traffic, sensors and customer reviews.
- **Cleaning the Data:** Imagine having to assess a pile of rocks that included some gold pieces in it. You would have to clean the dirt and the debris first. When data is being cleaned, mistakes must be fixed, duplicates must be removed and the data must be formatted properly.
- **Analyzing the Data:** It is here that the wizardry takes place. Data analysts employ powerful tools and techniques to discover patterns and trends. It is the same thing as looking for a specific pattern in all those rocks that you sorted through.



How does big data analytics work?

Big Data Analytics is a powerful tool which helps to find the potential of large and complex datasets. To get better understanding, let's break it down into key steps:

- **Data Collection:** Data is the core of Big Data Analytics. It is the gathering of data from different sources such as the customers' comments, surveys, sensors, social media, and so on. The primary aim of data collection is to compile as much accurate data as possible. The more data, the more insights.
- **Data Cleaning (Data Preprocessing):** The next step is to process this information. It often requires some cleaning. This entails the replacement of missing data, the correction of inaccuracies, and the removal of duplicates. It is like sifting through a treasure trove, separating the rocks and debris and leaving only the valuable gems behind.
- **Data Processing:** After that we will be working on the data processing. This process contains such important stages as writing, structuring, and formatting of data in a way it will be usable for the analysis. It is like a chef who is gathering the ingredients before cooking. [Data processing](#) turns the data into a format suited for analytics tools to process.
- **Data Analysis:** [Data analysis](#) is being done by means of statistical, mathematical, and machine learning methods to get out the most important findings from the processed

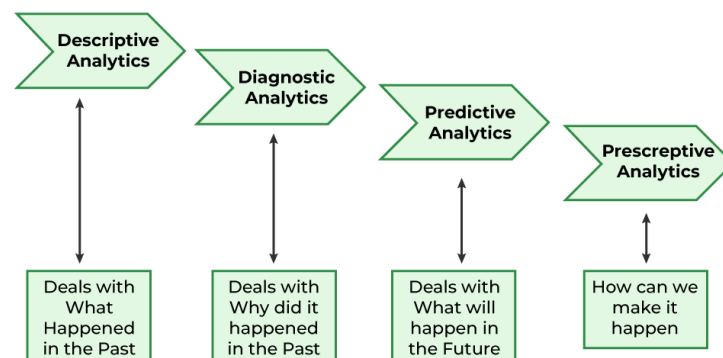
data. **For example**, it can uncover customer preferences, market trends, or patterns in healthcare data.

- **Data Visualization:** Data analysis usually is presented in visual form, for illustration – charts, graphs and interactive dashboards. The visualizations provided a way to simplify the large amounts of data and allowed for decision makers to quickly detect patterns and trends.
- **Data Storage and Management:** The stored and managed analyzed data is of utmost importance. It is like digital scrapbooking. May be you would want to go back to those lessons in the long run, therefore, how you store them has great importance. Moreover, data protection and adherence to regulations are the key issues to be addressed during this crucial stage.
- **Continuous Learning and Improvement:** Big data analytics is a continuous process of collecting, cleaning, and analyzing data to uncover hidden insights. It helps businesses make better decisions and gain a competitive edge.

Types of Big Data Analytics

Big Data Analytics comes in many different types, each serving a different purpose:

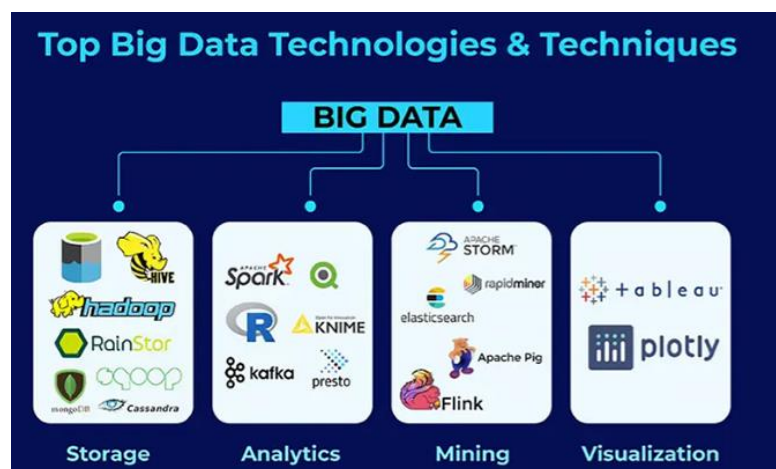
1. **Descriptive Analytics:** This type helps us understand past events. In social media, it shows performance metrics, like the number of likes on a post.
2. **Diagnostic Analytics:** In Diagnostic analytics delves deeper to uncover the reasons behind past events. In healthcare, it identifies the causes of high patient re-admissions.
3. **Predictive Analytics:** Predictive analytics forecasts future events based on past data. Weather forecasting, for example, predicts tomorrow's weather by analyzing historical patterns.
4. **Prescriptive Analytics:** However, this category not only predicts results but also offers recommendations for action to achieve the best results. In e-commerce, it may suggest the best price for a product to achieve the highest possible profit.
5. **Real-time Analytics:** The key function of real-time analytics is data processing in real time. It swiftly allows traders to make decisions based on real-time market events.
6. **Spatial Analytics:** Spatial analytics is about the location data. In urban management, it optimizes traffic flow from the data under the sensors and cameras to minimize the traffic jam.
7. **Text Analytics:** Text analytics delves into the unstructured data of text. In the hotel business, it can use the guest reviews to enhance services and guest satisfaction.



Big Data Analytics Technologies and Tools

Big Data Analytics relies on various technologies and tools that might sound complex, let's simplify them:

- **Hadoop:** Imagine Hadoop as an enormous digital warehouse. It's used by companies like Amazon to store tons of data efficiently. For instance, when Amazon suggests products you might like, it's because Hadoop helps manage your shopping history.
- **Spark:** Think of [Spark](#) as the super-fast data chef. Netflix uses it to quickly analyze what you watch and recommend your next binge-worthy show.
- **NoSQL Databases:** [NoSQL](#) databases, like [MongoDB](#), are like digital filing cabinets that Airbnb uses to store your booking details and user data. These databases are famous because of their quick and flexible, so the platform can provide you with the right information when you need it.
- **Tableau:** Tableau is like an artist that turns data into beautiful pictures. The World Bank uses it to create interactive charts and graphs that help people understand complex economic data.
- **Python and R:** [Python](#) and [R](#) are like magic tools for data scientists. They use these languages to solve tricky problems. For example, Kaggle uses them to predict things like house prices based on past data.
- **Machine Learning Frameworks (e.g., TensorFlow):** In [Machine learning](#) frameworks are the tools who make predictions. Airbnb uses [TensorFlow](#) to predict which properties are most likely to be booked in certain areas. It helps hosts make smart decisions about pricing and availability.



Benefits of Big Data Analytics

Big Data Analytics offers a host of real-world advantages, and let's understand with examples:

1. **Informed Decisions:** Imagine a store like Walmart. Big Data Analytics helps them make smart choices about what products to stock. This not only reduces waste but also keeps customers happy and profits high.
2. **Enhanced Customer Experiences:** Think about Amazon. Big Data Analytics is what makes those product suggestions so accurate. It's like having a personal shopper who knows your taste and helps you find what you want.
3. **Fraud Detection:** Credit card companies, like MasterCard, use Big Data Analytics to catch and stop fraudulent transactions. It's like having a guardian that watches over your money and keeps it safe.
4. **Optimized Logistics:** FedEx, for example, uses Big Data Analytics to deliver your packages faster and with less impact on the environment. It's like taking the fastest route to your destination while also being kind to the planet.

Challenges of Big data analytics

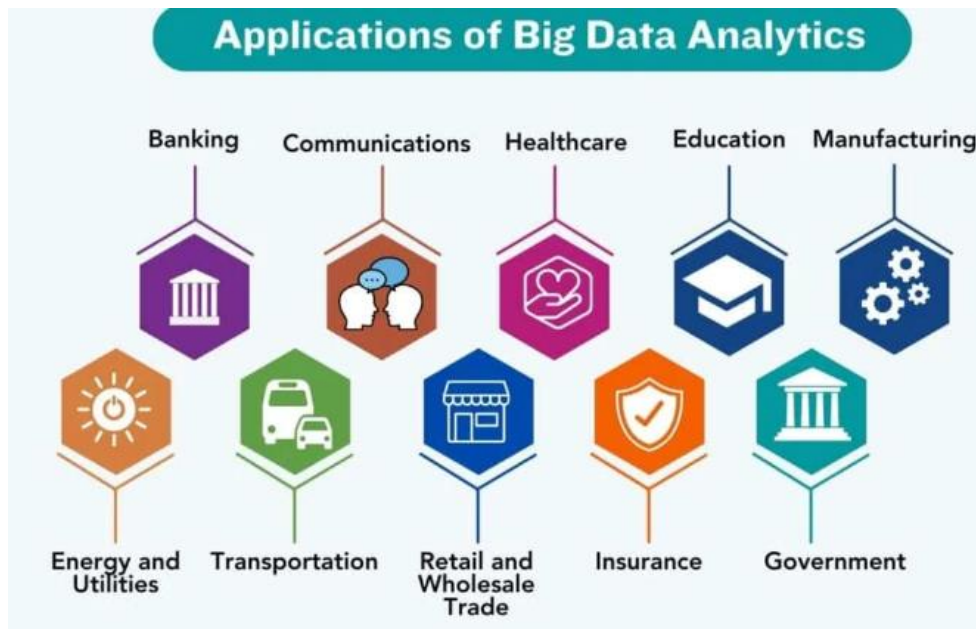
While Big Data Analytics offers incredible benefits, it also comes with its set of challenges:

- **Data Overload:** Consider Twitter, where approximately 6,000 tweets are posted every second. The challenge is sifting through this avalanche of data to find valuable insights.
- **Data Quality:** If the input data is inaccurate or incomplete, the insights generated by Big Data Analytics can be flawed. For example, incorrect sensor readings could lead to wrong conclusions in weather forecasting.
- **Privacy Concerns:** With the vast amount of personal data used, like in Facebook's ad targeting, there's a fine line between providing personalized experiences and infringing on privacy.
- **Security Risks:** With cyber threats increasing, safeguarding sensitive data becomes crucial. For instance, banks use Big Data Analytics to detect fraudulent activities, but they must also protect this information from breaches.
- **Costs:** Implementing and maintaining Big Data Analytics systems can be expensive. Airlines like Delta use analytics to optimize flight schedules, but they need to ensure that the benefits outweigh the costs.

Usage of Big Data Analytics

Big Data Analytics has a significant impact in various sectors:

- **Healthcare:** It aids in precise diagnoses and disease prediction, elevating patient care.
- **Retail:** Amazon's use of Big Data Analytics offers personalized product recommendations based on your shopping history, creating a more tailored and enjoyable shopping experience.
- **Finance:** Credit card companies such as Visa rely on Big Data Analytics to swiftly identify and prevent fraudulent transactions, ensuring the safety of your financial assets.
- **Transportation:** Companies like Uber use Big Data Analytics to optimize drivers' routes and predict demand, reducing wait times and improving overall transportation experiences.
- **Agriculture:** Farmers make informed decisions, boosting crop yields while conserving resources.
- **Manufacturing:** Companies like General Electric (GE) use Big Data Analytics to predict machinery maintenance needs, reducing downtime and enhancing operational efficiency.



2.2) Drivers of Big data analytics:

Drivers of Big Data Analytics: In-Depth Overview

Big Data Analytics is primarily driven by several factors that make it an indispensable tool for businesses, governments, and organizations across industries. These "drivers" stem from both technological advances and business needs. Understanding these drivers helps organizations harness the full potential of big data.

1. Technological Advancements in Data Generation and Storage

- **Data Explosion:** The rapid increase in data generated across various sources is one of the most significant drivers of big data analytics. From social media interactions to Internet of Things (IoT) devices, businesses are now capturing vast quantities of structured and unstructured data every day.
 - **Example:** The proliferation of mobile devices, sensors, and smart devices has led to an explosion of data. A company like **Tesla** collects data from its cars on driving patterns, maintenance needs, and road conditions in real-time, generating terabytes of data each day.
- **Advances in Storage Technologies:** Data storage technologies like **Hadoop** and **cloud storage** have made it possible to store, manage, and process huge amounts of data cost-effectively. This allows organizations to store vast volumes of unstructured data (such as images, videos, and social media content) and analyze them in real-time.
 - **Example:** Companies like **Amazon** and **Google** provide cloud-based solutions (AWS and Google Cloud) where businesses can store and process big data at a fraction of the traditional cost.

2. The Rise of Internet of Things (IoT)

- **Sensor-Driven Data:** IoT devices continuously collect data from the physical world. The growing adoption of smart devices such as wearables, connected home

appliances, and industrial sensors has created massive streams of data that need to be processed in real-time.

- **Example: General Electric (GE)** uses IoT sensors in its industrial machinery to monitor the performance of equipment. These sensors send real-time data, which is analyzed to predict machine failures before they occur, preventing costly downtime and maintenance.
 - **Example in Smart Cities:** IoT-powered devices are driving smart city initiatives. For instance, **Singapore** utilizes IoT sensors across the city to monitor traffic, air quality, water usage, and waste management. Real-time data collection enables government agencies to optimize resource allocation and improve urban planning.
-

3. Increasing Need for Data-Driven Decision Making

- **Real-Time Insights:** With the growing complexity of business environments, organizations are increasingly relying on data-driven decisions rather than intuition or gut feeling. Big data analytics enables them to make faster, more accurate decisions based on real-time insights from multiple sources.
 - **Example: UPS** uses data analytics to optimize delivery routes. By analyzing traffic data, weather forecasts, and package delivery histories in real-time, UPS can determine the most efficient route for delivery trucks, saving fuel and reducing delivery times.
 - **Competitive Advantage:** Businesses that adopt big data analytics are able to make more informed decisions, leading to a competitive edge in the marketplace. Retailers, for instance, leverage customer data to personalize shopping experiences.
 - **Example: Target**, the retail giant, uses big data to analyze customer buying patterns. By tracking purchasing behavior, it can predict when a customer is likely to need certain products, such as baby diapers, and send personalized offers. This ability to predict customer needs allows Target to increase sales and loyalty.
-

4. Availability of Advanced Analytics Tools and Techniques

- **Machine Learning & Artificial Intelligence:** Big data analytics is increasingly powered by machine learning (ML) and AI, which can process large data sets and automatically uncover patterns without human intervention. These tools help businesses derive actionable insights and predict future trends.
 - **Example: Spotify** uses machine learning algorithms to analyze listening patterns and provide personalized music recommendations. By analyzing millions of user interactions with the platform, Spotify's AI recommends music tailored to the individual, enhancing user engagement.
- **Predictive and Prescriptive Analytics:** As businesses generate more complex data, they rely on advanced analytics methods, such as predictive analytics (forecasting future trends) and prescriptive analytics (suggesting actions to optimize outcomes).
 - **Example: Delta Airlines** uses predictive analytics to estimate flight delays and proactively notify passengers, giving them the opportunity to make alternate travel arrangements. The use of big data helps the airline improve customer satisfaction and reduce operational costs.

5. The Growing Demand for Personalization and Customer Experience

- **Personalization:** Consumers now expect businesses to offer highly personalized products, services, and experiences. Big data analytics allows companies to analyze customer preferences, behavior, and purchase patterns to provide tailored offerings.
 - **Example: Netflix** uses big data analytics to offer personalized content recommendations. By analyzing what users watch, when they watch, and their ratings, Netflix is able to recommend movies and shows that match the individual's taste, keeping users engaged and reducing churn.
- **Omnichannel Experience:** Retailers use big data analytics to integrate customer interactions across multiple touchpoints (online, in-store, mobile apps) for a seamless experience.
 - **Example: Starbucks** leverages big data from its loyalty app to track customer purchase patterns. This enables Starbucks to provide personalized discounts and offers based on individual preferences, which not only improves customer experience but also increases sales.

6. Social Media and Online Content Generation

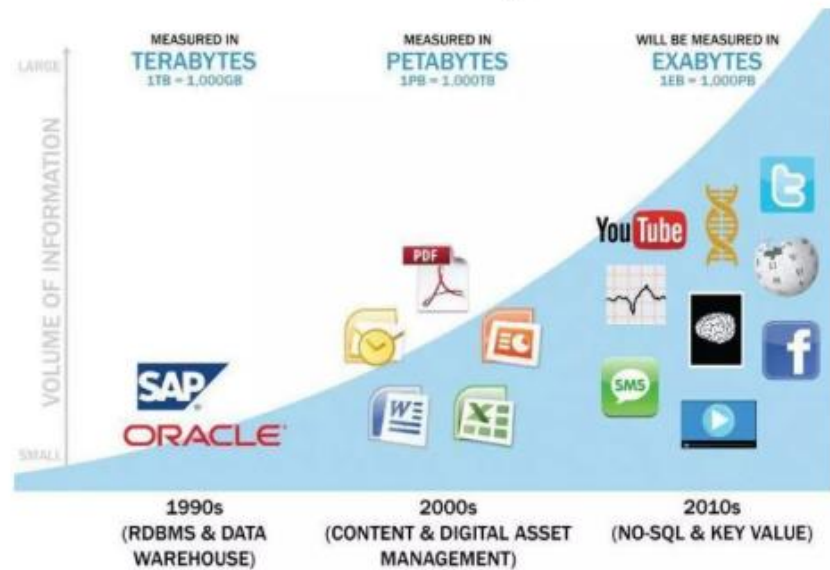
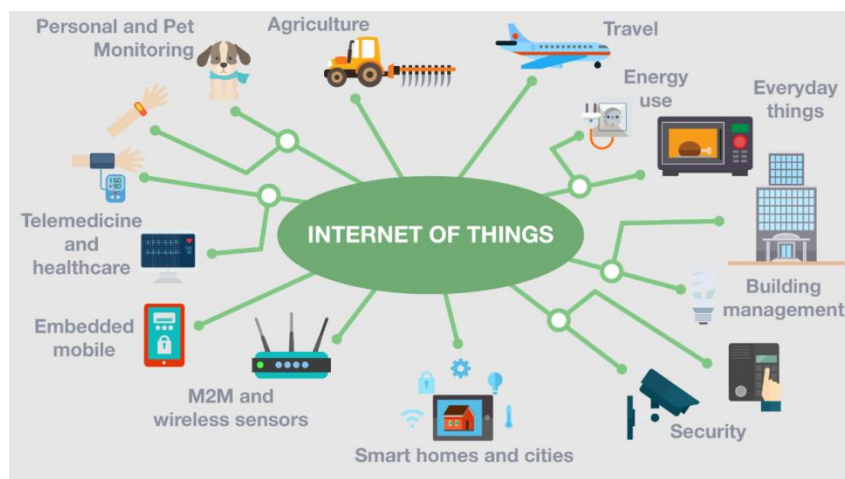
- **Volume and Variety of Data:** Social media platforms like Facebook, Twitter, Instagram, and YouTube generate massive amounts of user-generated content, including text, images, videos, and interactions. This rich data source is increasingly valuable for understanding customer sentiments, brand perception, and market trends.
 - **Example: Coca-Cola** uses social media sentiment analysis to understand consumer attitudes toward its products. By analyzing public sentiment on platforms like Twitter, Coca-Cola adjusts marketing strategies and promotional campaigns to align with consumer preferences.
- **Real-Time Analytics for Brand Management:** Big data tools are used to track online discussions and customer feedback in real-time. Companies can respond swiftly to public concerns or capitalize on opportunities for positive brand engagement.
 - **Example: Nike** tracks social media interactions to understand customer sentiment and identify trends. This allows Nike to launch targeted marketing campaigns or product releases that resonate with its audience.

7. Regulatory Compliance and Risk Management

- **Data Governance:** With stringent regulations around data privacy (e.g., GDPR, CCPA), companies need to ensure that their big data practices comply with legal standards. Big data analytics helps organizations ensure that sensitive data is handled securely and in compliance with regulations.
 - **Example: Financial Institutions** such as **JP Morgan Chase** use big data analytics to monitor and report transactions for compliance with financial regulations. They track customer transactions in real-time to prevent fraudulent activity, ensure transparency, and meet regulatory requirements.

- **Risk Management:** Big data analytics helps organizations identify, assess, and mitigate potential risks. By analyzing historical data and predicting future trends, companies can take proactive steps to avoid risks.
 - **Example: AIG** (American International Group), a leading insurance company, uses big data to assess risk factors, such as customer health data or environmental factors, to offer more accurate and competitive insurance policies.

Drivers of Big Data



2.3) Big Data Stack:

-A big data stack is a suite of complementary software technologies used to manage and analyze data sets too large or complex for traditional technologies.

-Big data stack technologies -- most often applied in analytics -- are specifically designed to address increases in the size, speed and structure of data.

- Big data products and services are often used to manage data within a data pipeline to deliver timely and efficient business insights.

Big data stack architecture layers

To address the challenges of big data, organizations must look beyond traditional data processing infrastructure. One area of recourse is special purpose big data software technologies. When used in concert, big data technologies can mitigate the effect of big data.

The following six layers are key to successful big data stack architecture:

1. Ingestion

The first step of a big data stack architecture is [data collection](#). Data acquisition can either push or pull from a wide range of internal and external data sources. Some examples of data sources include transactional systems, IoT devices, social media and static log files.

Big data ingestion software handles large static data sets, small real-time data sets and various data formats of each. The large data sets arrive slowly and the small sets arrive quickly. Postponing schema and quality validation to further along in the pipeline aids in higher throughput.

2. Storage

Once collected, raw data is typically stored as files [in a data lake](#) optimized for getting data into the analytics pipeline. Native format repositories are both landing zones for batch data and sandboxes for time sensitive exploratory inquiry.

Big data storage software stores both large and small files with various formats, which typically takes the form of a distributed file system such as object storage. Non-transient data can persist with long retention periods and requires software with automated tiering throughout the data's lifecycle.

3. Processing

Processing involves the preparation of batch data sets at rest and streaming data in motion [for analysis](#). Data curation can include cleaning, conforming, enriching, integrating, filtering, aggregating and otherwise preparing data for analysis.

Big data processing software operates on large batches of data, with higher latency and more complex computation, which requires long-running high-efficiency computing. Using distributed processing software acting on smaller pieces of partitioned data can achieve this.

Big data processing software also operates on high-speed streaming data, with lower latency and relatively simple computation. Streaming data processing requires guaranteed durability, order and delivery achieved through a continuously available streaming service.

4. Analytic stores

Analytic data stores process or refine data for analysis. Examples of data stores include SQL-based dimensional data warehouses, NoSQL technologies and distributed data stores with an abstraction layer for access to varying data types through interfaces.

Big data analytic stores support diverse storage approaches and technologies, referred to as [polyglot persistence](#). Special purpose single-model databases deliver performance and scalability by optimizing data storage and processing a particular data type. Essential tactics include data processing, parallel execution and data partitioning.

5. Analysis

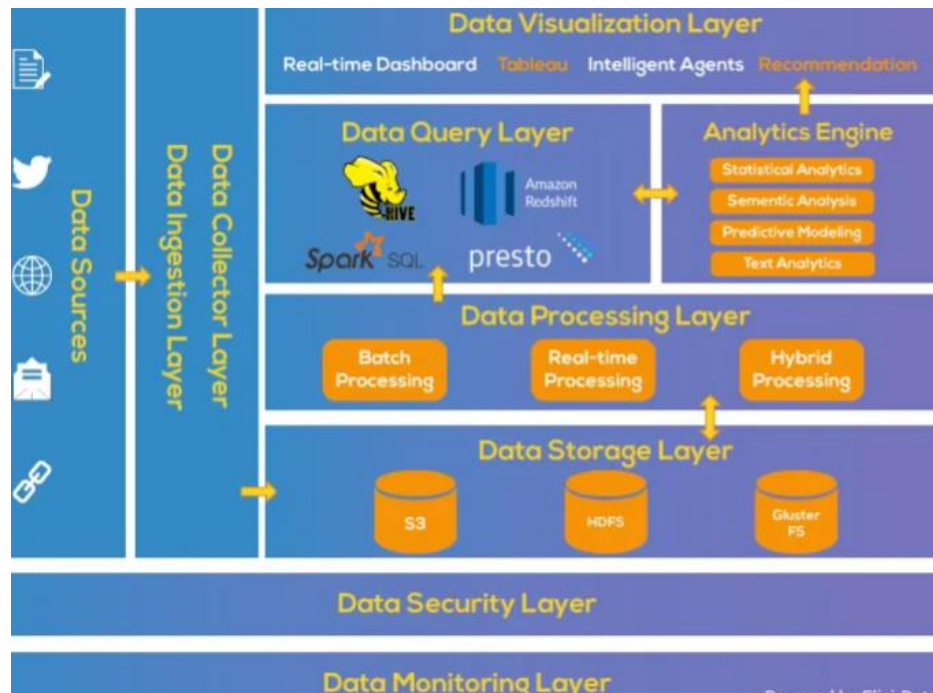
-Analysis examines analytic data stores and raw storage.

-Human users in an interactive setting use BI tools to gain insights through visualization. Advanced analytic tools crunch data to extract intelligence. Machine learning uses artificial intelligence to work directly on the data to teach itself.

-Big data analysis software handles the inquiries from simple ad hoc queries to sophisticated predictive analytics and machine learning operations.

6. Orchestration

Big data orchestration software [automates the data pipeline](#), which minimizes latency and achieves time to value. Workflow software provides an easy-to-use administrative interface and seamless integrations between architecture components.



2.4) Typical analytical architecture:

The analytical architecture of big data refers to the layered structure and components designed to manage, process, analyze, and extract meaningful insights from large and complex datasets. Below is a detailed explanation of its typical architecture:

1. Data Sources

This layer consists of the origins of the data, which can be structured, semi-structured, or unstructured. Examples include:

- **Structured:** Relational databases, ERP systems.
- **Semi-structured:** JSON, XML files, logs.
- **Unstructured:** Videos, images, audio, social media, IoT sensors.
- **Real-time streams:** Clickstream data, sensor feeds, and web logs.

2. Data Ingestion Layer

The data ingestion layer collects and imports raw data from various sources into the system.

- **Batch Ingestion:** Periodic collection of data (e.g., Apache Sqoop, Apache Nifi).
- **Real-time Ingestion:** Continuous streaming of data (e.g., Apache Kafka, Flume).

3. Data Storage Layer

Data is stored in this layer for processing and analysis. It can handle large-scale, heterogeneous data types:

- **Data Lakes:** For raw, unstructured, and semi-structured data (e.g., HDFS, Amazon S3).
- **Data Warehouses:** For structured data and analytics (e.g., Amazon Redshift, Snowflake).
- **NoSQL Databases:** For scalable, high-speed, schema-less storage (e.g., Cassandra, MongoDB).

4. Data Processing Layer

This layer processes raw data to make it analyzable. There are two types of processing:

- **Batch Processing:** Suitable for large datasets processed over time (e.g., Hadoop MapReduce, Apache Spark).
- **Stream Processing:** Handles real-time data streams (e.g., Apache Storm, Apache Flink).

5. Data Integration and Transformation Layer

- **ETL (Extract, Transform, and Load):** Transforms raw data into structured formats and loads it into storage for further processing.
- **Data Cleaning:** Removal of duplicate, incomplete, or erroneous data.
- **Data Transformation:** Data normalization, enrichment, and aggregation.

6. Analytical Layer

This layer provides tools for analyzing the processed data:

- **Machine Learning and AI:** Algorithms and models to find patterns and predictions (e.g., TensorFlow, PyTorch).
- **Statistical Analysis:** Tools like R, SAS, or Python libraries (e.g., pandas, NumPy, SciPy).
- **Data Querying:** SQL engines for querying large datasets (e.g., Hive, Presto).

7. Visualization Layer

Data insights are presented in user-friendly formats for decision-making:

- **BI Tools:** Dashboards and reports (e.g., Tableau, Power BI, QlikView).
 - **Custom Visualization:** Built with libraries like D3.js, Matplotlib, or Plotly.
-

8. Access and Security Layer

This layer ensures data is accessible and secure:

- **Access Control:** Authentication and authorization (e.g., Kerberos, LDAP).
 - **Data Encryption:** Securing data at rest and in transit.
 - **Audit Trails:** Tracking data access and changes.
-

9. Feedback and Optimization Layer

This layer refines the system based on user feedback and performance:

- **Monitoring Tools:** For system health (e.g., Nagios, Splunk).
 - **Optimization:** Fine-tuning processing pipelines, storage layers, and analytical models for efficiency.
-

Key Characteristics of Big Data Architecture

1. **Scalability:** Ability to handle increasing data volumes.
 2. **Fault Tolerance:** Ensures reliability even when components fail.
 3. **Low Latency:** Quick processing, especially for real-time use cases.
 4. **Flexibility:** Accommodates diverse data types and sources.
-

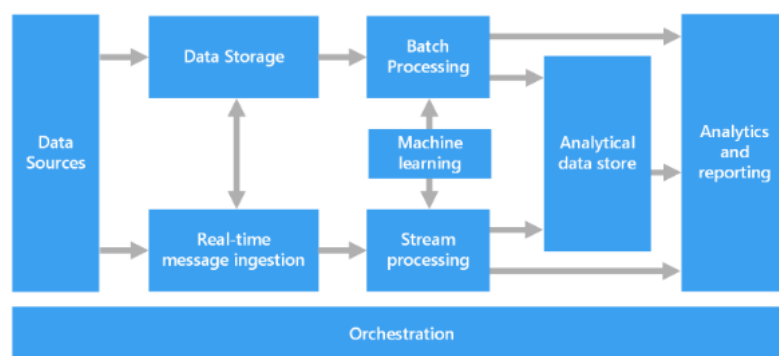


Fig.Name: Typical Architecture of Analytics

2.5) Virtualization:

-The foundation of data virtualization technology is the execution of distributed data management processes, mostly for queries, against numerous heterogeneous data sources, and the Federation of query results into virtual views.

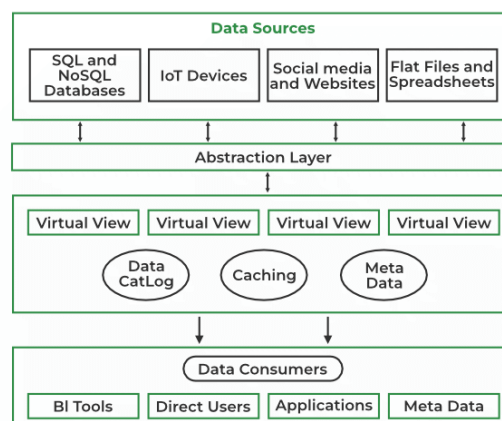
-Applications, query/reporting tools, message-oriented middleware, or other parts of the data management infrastructure then consume these virtual views. Instead of performing data movement and physically storing integrated views in a destination data structure, data virtualization can be utilized to construct virtualized and integrated views of data in memory.

-To make querying logic simpler, it provides an abstraction layer over the actual physical implementation of data.

-It is a method for combining data from various sources and different types into a comprehensive, logical representation without physically relocating the data. Simply put, users can theoretically access and examine data while it still exists in its original sources thanks to specialized middleware.

Features of Data Virtualization

- **Time to market acceleration from data to final product:** - Virtual data objects can be created considerably more quickly than existing ETL tools and databases since they include integrated data. Customers may now more easily get the information they require.
- **One-Stop Security:-** The contemporary data architecture makes it feasible to access data from a single location. Data can be secured down to the row and column level thanks to the virtual layer that grants access to all organizational data. Authorizing numerous user groups on the same virtual dataset is feasible by using data masking, anonymization, and pseudonymization.
- **Combine data explicitly from different sources:-** The virtual data layer makes it simple to incorporate distributed data from Data Warehouses, Big Data Platforms, Data lakes, Cloud Solutions, and Machine Learning into user-required data objects.
- **Flexibility:-** It is feasible to react quickly to new advances in various sectors thanks to data virtualization. This is up to ten times faster than conventional ETL and data warehousing methods. By providing integrated virtual data objects, data virtualization enables you to reply instantly to fresh data requests. This does away with the necessity to copy data to various data levels but just makes it virtually accessible.



Layers of Data Virtualization:

Following are the working layers in data virtualization architecture.

- **Connection layer:** With the use of connectors and communication protocols, this layer is in charge of accessing the data dispersed across numerous source systems that contain both organized and unstructured data. Platforms for data virtualization can connect to various data sources, such as SQL and NoSQL databases like MySQL, Oracle and MongoDB etc.
- **Abstraction layer:** The abstraction layer, also known as the virtual or semantic layer, serves as a link between all data sources and all business users, forming the backbone of the entire virtualization system. This tier just holds the logical views and information required to access the sources; it does not itself store any data. The complexity of the underlying data structures is hidden from end users, who only see the schematic data models thanks to the abstraction layer.
- **Consumption layer:** A single point of access to the data stored in the underlying sources is offered by a different tier of the data virtualization architecture. Depending on the type of consumer, several protocols and connectors are used to give abstracted data representations. They can interface with the virtual layer using SQL and a variety of APIs, such as REST and SOAP APIs, as well as access standards like JDBC and ODBC. A variety of corporate users, tools, and apps can all have access to data virtualization software, including well-known ones like Tableau, Cognos, and Power BI.

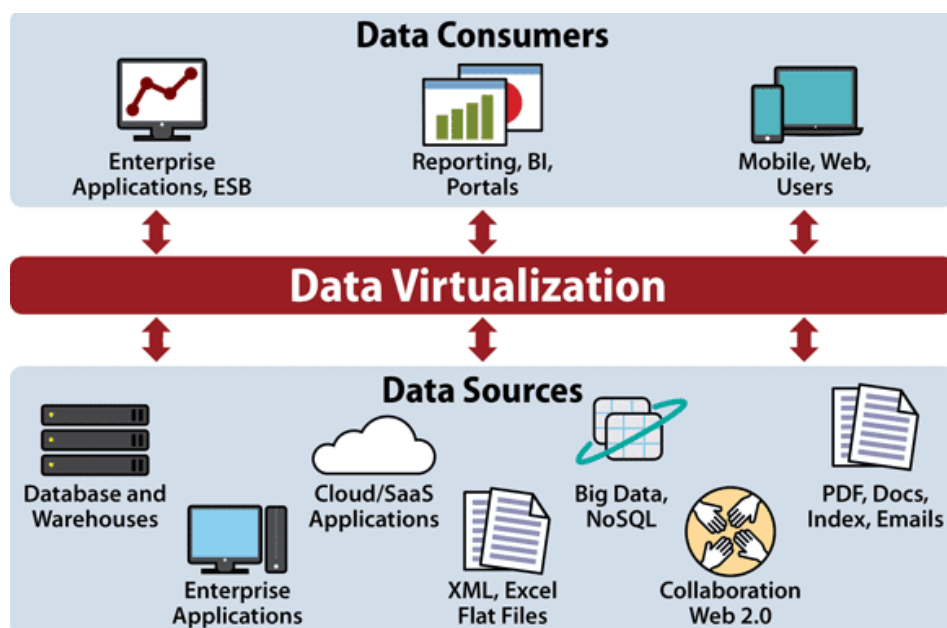
Applications of Data Virtualization

- **Migration:** Think of a scenario where you migrate a CRM system to the cloud from a traditional one. Or a gradual migration of old systems to the cloud. You can accomplish this with data virtualization without halting operations or reporting.
- **Uses In Operations:** - For call centres or customer support systems, data silos are a big source of misery that have lasted for a very long time. A bank would, for instance, choose one call centre for credit cards and another for home loans. Data virtualization that spans data silos enables everyone, from a call centre to a database manager, to see the full range of data repositories from a single point of access.
- **Agile BI:** - With data virtualization, you can use your data for data science, API or system linkages, governed (regulated), and self-service BI. Additionally, it's perfect for "agile" BI, which involves developing dashboards and reports in incredibly fast iterations that include testing, piloting, and production. Would you wish to add new sources to your current BI stream by connecting SaaS cloud services like Salesforce or Google Analytics? You may! You may combine all of your data with data virtualization, even in a hybrid environment. Additionally, you don't need to worry about security because it is highly centralised.
- **Data Integration:-** This is the most likely situation you will encounter because practically every company contains data from multiple different data sources. Connecting an antiquated client/server-based data source with modern digital platforms like social media is required for that. You use the data catalogue to search your data after connecting using methods like Java DAO, ODBC, SOAP, or other APIs. Constructing connections is more likely to be difficult, even with data virtualization.

- **Accessing Real-Time Data:-** Are your SLA agreements under pressure and a source system not performing adequately in terms of (near) real-time accessibility to massive amounts of data? Data virtualization allows you to blend real-time data from the source system with historical data that has been “offloaded” to a different source. You can prevent overtaxing your source systems by optimising your caching or conducting more intelligent system queries. Without initially copying every type of data with ETL operations, even near real-time analytics on huge data are feasible. Additionally, it is simple to create a virtual data mart by combining an outdated data warehouse with a fresh data source.

Advantages of Data Virtualization

- Data virtualization enables real-time access to and manipulation of source data through the virtual/logical layer without physically relocating the data to a new location. ETL is typically not required.
- Comparing the implementation of data virtualization to the construction of a separate consolidated store, the former takes less funding and resources.
- There is no need to relocate the material, and access levels may be controlled.
- Without worrying about a data type or where the data is located, users can build and execute whatever reports and analyses they require.
- Through a single virtual layer, all corporate data is accessible to all consumers and use cases.



2.6) Virtualization Approaches:

Virtualization in the context of **Big Data Analytics** refers to leveraging virtualization technologies to manage, store, and process large volumes of data efficiently and flexibly. By virtualizing resources such as servers, storage, and networks, big data platforms can enhance scalability, reduce costs, and improve performance. Here's a detailed breakdown of **virtualization approaches in big data analytics**:

1. Server Virtualization

Server virtualization involves partitioning a single physical server into multiple virtual servers, each capable of running its own operating system and applications. This approach is particularly beneficial for big data analytics platforms like Hadoop and Spark.

Key Benefits:

- **Efficient Resource Utilization:** Virtual servers enable better use of CPU and memory.
- **Cost Savings:** Reduces the need for additional physical hardware.
- **Scalability:** Virtual servers can be provisioned or decommissioned on demand.

Use in Big Data:

- Hosting **Hadoop clusters** or **Apache Spark nodes** in virtualized environments.
- Ensuring fault tolerance by quickly spinning up new virtual servers in case of failure.

Tools:

- VMware vSphere
 - Microsoft Hyper-V
 - KVM (Kernel-based Virtual Machine)
-

2. Storage Virtualization

Storage virtualization abstracts physical storage across multiple devices to create a unified storage pool. Big data applications can leverage this to handle large volumes of unstructured and structured data seamlessly.

Key Benefits:

- **Centralized Management:** Simplifies the management of diverse storage resources.
- **Scalability:** Easily scales to accommodate growing data volumes.
- **High Availability:** Provides redundancy and replication across storage devices.

Use in Big Data:

- Creating **data lakes** to store raw data from multiple sources.
- Implementing **distributed file systems** like HDFS (Hadoop Distributed File System) on virtualized storage.

Tools:

- VMware vSAN
 - IBM Spectrum Virtualize
 - NetApp ONTAP
-

3. Network Virtualization

Network virtualization abstracts the physical network infrastructure into virtual networks, enabling better control and flexibility in big data deployments.

Key Benefits:

- **Traffic Optimization:** Isolates big data traffic to minimize congestion.
- **Security:** Virtual networks can segment sensitive data processing tasks.
- **Flexibility:** Dynamic reconfiguration of network resources.

Use in Big Data:

- Managing **data flows** in big data pipelines.
- Enhancing performance in **real-time analytics** scenarios.

Tools:

- VMware NSX
 - Cisco ACI
 - OpenStack Neutron
-

4. Data Virtualization

Data virtualization provides a unified data access layer that abstracts the complexities of data sources, enabling analytics tools to access data without needing to know its physical location or format.

Key Benefits:

- **Real-Time Access:** Offers on-demand data integration across silos.
- **Reduced Data Movement:** Minimizes the need to physically move data.
- **Simplified Integration:** Aggregates structured and unstructured data seamlessly.

Use in Big Data:

- Enabling **faster querying** of distributed datasets.
- Simplifying data preparation for analytics and machine learning.

Tools:

- Denodo
 - TIBCO Data Virtualization
 - IBM Cloud Pak for Data
-

5. Application Virtualization

Application virtualization allows big data analytics applications to run in isolated environments without requiring direct installation on the host OS.

Key Benefits:

- **Portability:** Analytics tools can run on various systems without compatibility issues.
- **Simplified Updates:** Easier to update or replace analytics tools without affecting the host environment.
- **Reduced Conflicts:** Applications run independently of the underlying OS.

Use in Big Data:

- Deploying **analytics software** like Tableau, Power BI, or custom ML models.
- Containerizing **data analytics pipelines** for scalability and portability.

Tools:

- Docker
 - VMware ThinApp
 - Citrix App Layering
-

6. Desktop Virtualization

In big data, desktop virtualization can provide analysts and data scientists with secure, remote access to data analysis tools and platforms.

Key Benefits:

- **Centralized Access:** Analysts can access tools and data from anywhere.
- **Enhanced Security:** Sensitive data never leaves the central server.
- **Cost-Effective:** Reduces the need for high-performance local machines.

Use in Big Data:

- Providing access to **cloud-based analytics platforms**.
- Enabling remote collaboration for **data science teams**.

Tools:

- Citrix Virtual Apps
 - VMware Horizon
 - Amazon WorkSpaces
-

7. Containerization (OS-Level Virtualization)

Containerization is a form of OS-level virtualization that packages applications and their dependencies into lightweight containers. It has revolutionized big data processing by making applications portable and scalable.

Key Benefits:

- **Lightweight:** Uses fewer resources than traditional VMs.
- **Portability:** Containers can run consistently across environments.
- **Rapid Deployment:** Quickly scale analytics applications or tools.

Use in Big Data:

- Hosting **big data frameworks** like Apache Kafka, Hadoop, or Spark in containers.
- Running **AI/ML models** in containerized environments for real-time predictions.

Tools:

- Docker
 - Kubernetes
 - Apache Mesos
-

8. Cloud Virtualization

Cloud virtualization allows big data platforms to use virtualized resources hosted on cloud providers, enabling elastic scaling and pay-as-you-go cost models.

Key Benefits:

- **Elasticity:** Scale compute, storage, and network resources dynamically.
- **Global Availability:** Access data and analytics from anywhere.
- **Cost Efficiency:** Only pay for what you use.

Use in Big Data:

- Running big data platforms like **Google BigQuery**, **Amazon EMR**, or **Azure Synapse Analytics**.
- Hosting **data lakes** and supporting **real-time analytics**.

Providers:

- AWS (Elastic Compute Cloud - EC2)
- Microsoft Azure (Virtual Machines, Synapse Analytics)
- Google Cloud (Compute Engine, BigQuery)

Advantages of Virtualization in Big Data Analytics

1. **Resource Efficiency:** Maximizes hardware utilization.
2. **Cost-Effectiveness:** Reduces capital expenditure on physical infrastructure.
3. **Scalability:** Supports dynamic scaling of big data applications.
4. **Flexibility:** Enables the deployment of heterogeneous applications.
5. **Resilience:** Enhances fault tolerance and disaster recovery.

Challenges

- **Overhead:** Virtualization can introduce performance overhead.
- **Complexity:** Managing virtualized environments can be challenging.
- **Security:** Requires robust security measures to prevent breaches in multi-tenant environments.

2.7) Business Intelligence Vs. Data science

S. No.	Factor	Data Science	Business Intelligence
1.	Concept	It is a field that uses mathematics, statistics and various other tools to discover the hidden patterns in the data.	It is basically a set of technologies, applications and processes that are used by the enterprises for business data analysis.
2.	Focus	It focuses on the future.	It focuses on the past and present.
3.	Data	It deals with both structured as well as unstructured data.	It mainly deals only with structured data.

S. No.	Factor	Data Science	Business Intelligence
4.	Flexibility	Data science is much more flexible as data sources can be added as per requirement.	It is less flexible as in case of business intelligence data sources need to be pre-planned.
5.	Method	It makes use of the scientific method.	It makes use of the analytic method.
6.	Complexity	It has a higher complexity in comparison to business intelligence.	It is much simpler when compared to data science.
7.	Expertise	It's expertise is data scientist.	It's expertise is the business user.
8.	Questions	It deals with the questions of what will happen and what if.	It deals with the question of what happened.
9.	Storage	The data to be used is disseminated in real-time clusters.	Data warehouse is utilized to hold data.
10.	Integration of data	The ELT (Extract-Load-Transform) process is generally used for the integration of data for data science applications.	The ETL (Extract-Transform-Load) process is generally used for the integration of data for business intelligence applications.
11.	Tools	It's tools are SAS, BigML, MATLAB, Excel, etc.	It's tools are InsightSquared Sales Analytics, Klipfolio, ThoughtSpot, Cyfe, TIBCO Spotfire, etc.

S. No.	Factor	Data Science	Business Intelligence
12.	Usage	Companies can harness their potential by anticipating the future scenario using data science in order to reduce risk and increase income.	Business Intelligence helps in performing root cause analysis on a failure or to understand the current status.
13.	Business Value	Greater business value is achieved with data science in comparison to business intelligence as it anticipates future events.	Business Intelligence has lesser business value as the extraction process of business value carries out statically by plotting charts and KPIs (Key Performance Indicator).
14.	Handling data sets	The technologies such as Hadoop are available and others are evolving for handling understanding Its large data sets.	The sufficient tools and technologies are not available for handling large data sets.

2.8) Applications of Big data analytics:

1. Tracking Customer Spending Habit, Shopping Behavior: In big retails store (like Amazon, Walmart, Big Bazar etc.) management team has to keep data of customer's spending habit (in which product customer spent, in which brand they wish to spent, how frequently they spent), shopping behavior, customer's most liked product (so that they can keep those products in the store). Which product is being searched/sold most, based on that data, production/collection rate of that product get fixed.

Banking sector uses their customer's spending behavior-related data so that they can provide the offer to a particular customer to buy his particular liked product by using bank's credit or debit card with discount or cashback. By this way, they can send the right offer to the right person at the right time.

2. Recommendation: By tracking customer spending habit, shopping behavior, Big retails store provide a recommendation to the customer. E-commerce site like Amazon, Walmart, and Flipkart does product recommendation. They track what product a customer is searching, based on that data they recommend that type of product to that customer. As an example, suppose any customer searched bed cover on Amazon. So, Amazon got data that customer may be interested to buy bed cover. Next time when that customer will

go to any google page, advertisement of various bed covers will be seen. Thus, advertisement of the right product to the right customer can be sent.

YouTube also shows recommend video based on user's previous liked, watched video type. Based on the content of a video, the user is watching, relevant advertisement is shown during video running. As an example suppose someone watching a tutorial video of Big data, then advertisement of some other big data course will be shown during that video.

3. Smart Traffic System: Data about the condition of the traffic of different road, collected through camera kept beside the road, at entry and exit point of the city, GPS device placed in the vehicle (Ola, Uber cab, etc.). All such data are analyzed and jam-free or less jam way, less time taking ways are recommended. Such a way smart traffic system can be built in the city by Big data analysis. One more profit is fuel consumption can be reduced.

4. Secure Air Traffic System: At various places of flight (like propeller etc) sensors present. These sensors capture data like the speed of flight, moisture, temperature, other environmental condition. Based on such data analysis, an environmental parameter within flight are set up and varied.

By analyzing flight's machine-generated data, it can be estimated how long the machine can operate flawlessly when it to be replaced/repaired.

5. Auto Driving Car: Big data analysis helps drive a car without human interpretation. In the various spot of car camera, a sensor placed, that gather data like the size of the surrounding car, obstacle, distance from those, etc. These data are being analyzed, then various calculation like how many angles to rotate, what should be speed, when to stop, etc carried out. These calculations help to take action automatically.

6. Virtual Personal Assistant Tool: Big data analysis helps virtual personal assistant tool (like Siri in Apple Device, Cortana in Windows, Google Assistant in Android) to provide the answer of the various question asked by users. This tool tracks the location of the user, their local time, season, other data related to question asked, etc. Analyzing all such data, it provides an answer.

As an example, suppose one user asks "Do I need to take Umbrella?", the tool collects data like location of the user, season and weather condition at that location, then analyze these data to conclude if there is a chance of raining, then provide the answer.

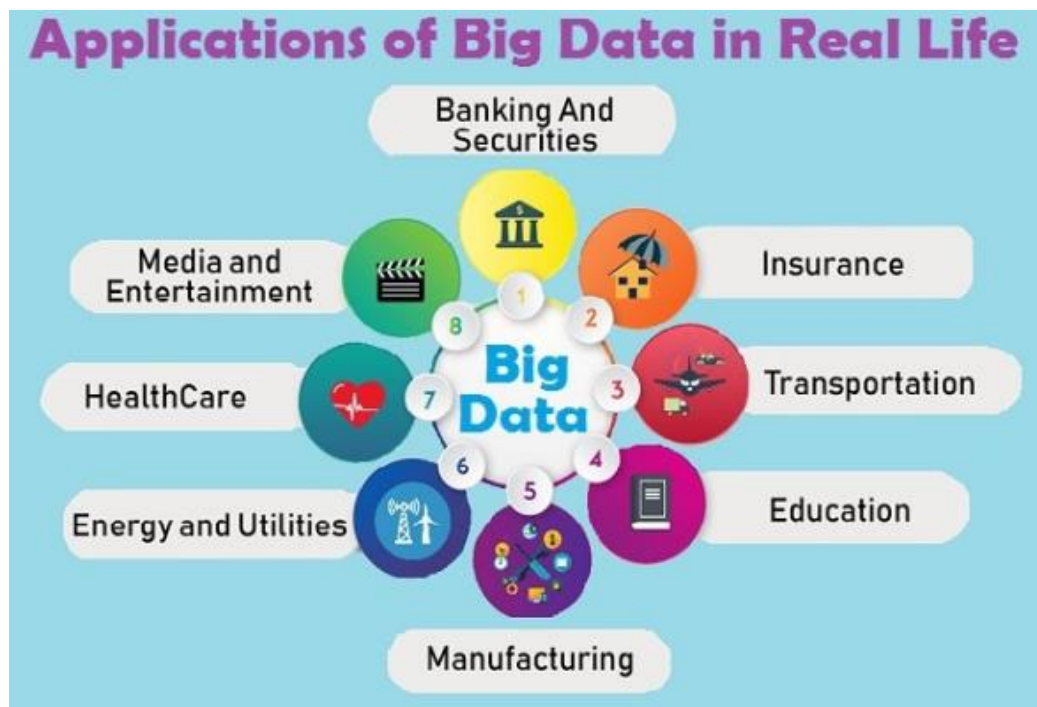
7. IoT:

- Manufacturing company install IOT sensor into machines to collect operational data. Analyzing such data, it can be predicted how long machine will work without any problem when it requires repairing so that company can take action before the situation when machine facing a lot of issues or gets totally down. Thus, the cost to replace the whole machine can be saved.
- In the Healthcare field, Big data is providing a significant contribution. Using big data tool, data regarding patient experience is collected and is used by doctors to give better treatment. IoT device can sense a symptom of probable coming disease in the human body and prevent it from giving advance treatment. IoT Sensor placed near-patient, new-born baby constantly keeps track of various health condition like heart bit rate, blood presser, etc. Whenever any parameter crosses the safe limit, an alarm sent to a doctor, so that they can take step remotely very soon.

8. Education Sector: Online educational course conducting organization utilize big data to search candidate, interested in that course. If someone searches for YouTube tutorial video on a subject, then online or offline course provider organization on that subject send ad online to that person about their course.

9. Energy Sector: Smart electric meter read consumed power every 15 minutes and sends this read data to the server, where data analyzed and it can be estimated what is the time in a day when the power load is less throughout the city. By this system manufacturing unit or housekeeper are suggested the time when they should drive their heavy machine in the night time when power load less to enjoy less electricity bill.

10. Media and Entertainment Sector: Media and entertainment service providing company like Netflix, Amazon Prime, Spotify do analysis on data collected from their users. Data like what type of video, music users are watching, listening most, how long users are spending on site, etc are collected and analyzed to set the next business strategy.



Unit III: Data Analytics Lifecycle

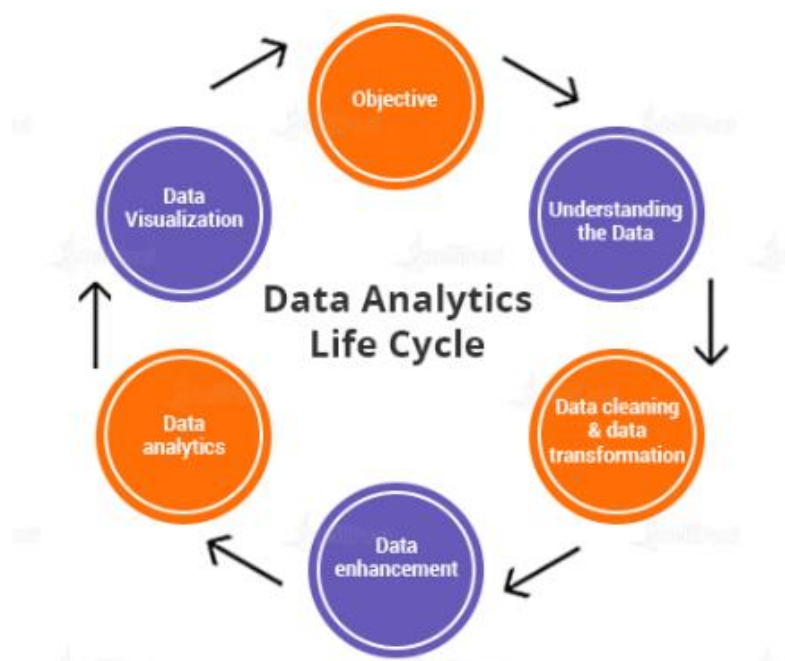
3.1] Need of Data analytic lifecycle:

Data is precious in today's digital environment. It goes through several life stages, including creation, testing, processing, consumption, and reuse. These stages are mapped out in the Data Analytics Life Cycle for professionals working on data analytics initiatives. Each stage has its own significance and characteristics.

Importance of Data Analytics Life Cycle

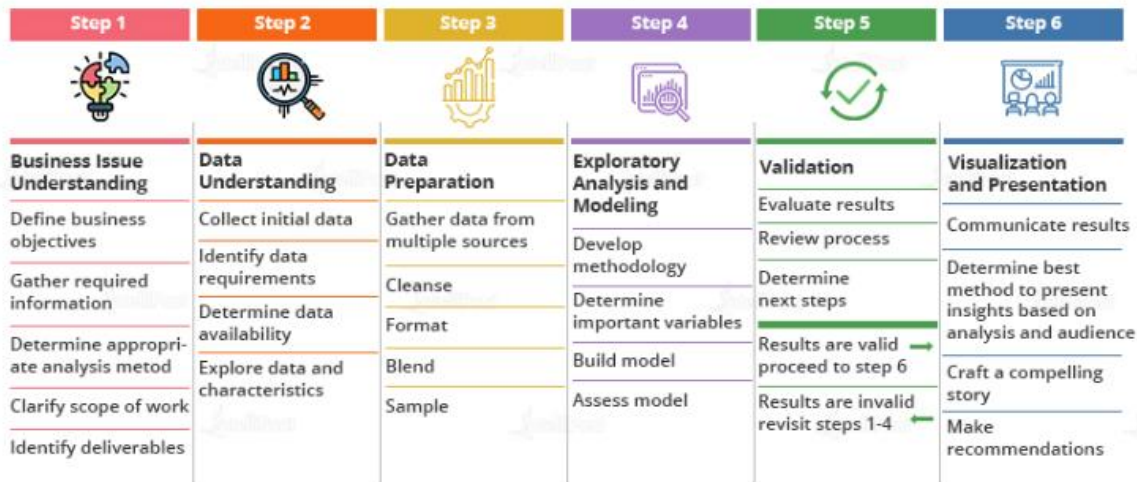
The data analytics Life Cycle encompasses the process of producing, collecting, processing, using, and analyzing data in order to meet corporate objectives.

[Data Analytics](#) offers a systematic way for managing data into useful information that can help achieve organizational or project goals; additionally, it provides guidance and strategies for extracting this information and moving in the appropriate direction in order to meet corporate objectives



LIFE CYCLE OF DATA ANALYSIS PROJECT

Based on CRISP-DM Methodology



OR

The **Data Analytics Lifecycle** is a structured methodology used to guide data-driven projects, ensuring that insights derived from data are accurate, actionable, and aligned with business objectives. It consists of a series of stages that organizations follow to efficiently process data and generate meaningful insights. Below is a detailed explanation of each stage:

1. Discovery

- **Objective:** Understand the project goals, define the problem, and identify data requirements.
- **Activities:**
 - Meet with stakeholders to clarify objectives.
 - Identify key questions that need to be answered.
 - Determine resources (tools, personnel, technology).
 - Assess data availability and quality.
- **Outcome:**
 - A clear understanding of the problem and goals.
 - A project plan or roadmap.
 - Feasibility analysis and risk assessment.

2. Data Preparation

- **Objective:** Gather, clean, and organize data for analysis.

- **Activities:**
 - Data collection from various sources (databases, APIs, sensors).
 - Cleaning data to handle missing values, outliers, and inconsistencies.
 - Data transformation and integration to standardize formats.
 - Sampling, if necessary, to reduce computational complexity.
 - **Outcome:**
 - A well-structured dataset ready for analysis.
-

3. Model Planning

- **Objective:** Define and select analytical techniques and models.
 - **Activities:**
 - Exploratory Data Analysis (EDA) to identify patterns and relationships.
 - Selection of algorithms or models based on the data type and objectives (e.g., regression, clustering).
 - Development of workflows or pipelines.
 - Setting up validation strategies to evaluate model performance.
 - **Outcome:**
 - A blueprint of the analytical approach.
 - Initial hypotheses and feature selection.
-

4. Model Building

- **Objective:** Develop and train the chosen analytical models.
 - **Activities:**
 - Implement models using programming languages or tools (e.g., Python, R, SAS).
 - Train models using historical data.
 - Optimize parameters for improved accuracy.
 - Validate models with test datasets.
 - **Outcome:**
 - A trained and validated model ready for deployment.
-

5. Model Evaluation

- **Objective:** Assess the model's performance and ensure it meets business objectives.
 - **Activities:**
 - Compare predicted results with actual outcomes using metrics like accuracy, precision, recall, etc.
 - Perform robustness checks and test for overfitting or underfitting.
 - Seek feedback from stakeholders to align results with business goals.
 - **Outcome:**
 - A validated model that aligns with the problem statement and objectives.
-

6. Deployment

- **Objective:** Implement the model in a production environment and deliver actionable insights.
 - **Activities:**
 - Deploy the model as an application or integrate it with existing systems.
 - Automate processes for real-time data ingestion and analysis.
 - Document processes and train users or stakeholders.
 - **Outcome:**
 - A live solution generating insights or predictions.
 - Documentation for reproducibility and knowledge sharing.
-

7. Monitoring and Maintenance

- **Objective:** Ensure the solution remains accurate and relevant over time.
 - **Activities:**
 - Monitor model performance with real-time data.
 - Update models with new data as needed (retraining).
 - Identify and address potential biases or errors.
 - Gather user feedback for continuous improvement.
 - **Outcome:**
 - A sustainable and reliable analytics solution.
-

Why Is the Data Analytics Lifecycle Important?

1. **Structured Workflow:** Helps in organizing efforts, ensuring all steps are covered systematically.
2. **Improved Accuracy:** Reduces errors by validating and iterating through models.
3. **Alignment with Business Goals:** Ensures that analytics projects are driven by and deliver on organizational needs.
4. **Transparency:** Promotes clear communication and documentation at every stage.
5. **Scalability:** Supports replication and adaptation for future projects.

3.2] Key roles for successful analytic projects:

Successful analytics projects require a diverse team with specialized roles to ensure that all aspects of the project are effectively addressed. Each role contributes unique expertise, and collaboration among these roles is critical. Here's a detailed explanation of the key roles:

1. Project Sponsor

- **Role:** A senior stakeholder who champions the analytics project within the organization.
- **Responsibilities:**
 - Define the project's objectives and align them with business goals.
 - Secure funding and resources.
 - Provide guidance and make high-level decisions.
 - Advocate for the project's outcomes to ensure adoption.
- **Skills:**
 - Strong understanding of business strategy.
 - Leadership and decision-making capabilities.
- **Impact:**
 - Ensures the project has organizational support and aligns with strategic goals.

2. Project Manager

- **Role:** Oversees the project's execution, ensuring timelines, budgets, and objectives are met.

- **Responsibilities:**
 - Develop and manage the project plan.
 - Coordinate team activities and communication.
 - Mitigate risks and address issues.
 - Ensure deliverables meet quality standards.
 - **Skills:**
 - Expertise in project management methodologies (e.g., Agile, Scrum).
 - Strong organizational and communication skills.
 - **Impact:**
 - Keeps the project on track and ensures smooth collaboration among team members.
-

3. Data Scientist

- **Role:** Analyzes data to extract insights, build predictive models, and solve complex problems.
 - **Responsibilities:**
 - Perform data exploration and pre-processing.
 - Develop and validate machine learning or statistical models.
 - Interpret and present findings in a meaningful way.
 - Collaborate with domain experts to ensure relevance.
 - **Skills:**
 - Proficiency in programming languages (e.g., Python, R).
 - Knowledge of machine learning, statistics, and data visualization.
 - **Impact:**
 - Drives the analytical core of the project and produces actionable insights.
-

4. Data Engineer

- **Role:** Builds and maintains the infrastructure needed for data storage, processing, and accessibility.
- **Responsibilities:**
 - Design and manage data pipelines.
 - Ensure data quality, integrity, and security.
 - Optimize database performance and scalability.
 - Integrate data from multiple sources into a unified system.
- **Skills:**

- Expertise in databases (SQL, NoSQL), ETL tools, and cloud platforms.
 - Knowledge of big data frameworks like Hadoop or Spark.
 - **Impact:**
 - Ensures data is accessible, clean, and ready for analysis.
-

5. Business Analyst

- **Role:** Acts as a bridge between technical teams and business stakeholders.
 - **Responsibilities:**
 - Understand and translate business requirements into analytical objectives.
 - Identify key performance indicators (KPIs) and success metrics.
 - Ensure findings align with business needs and goals.
 - **Skills:**
 - Strong analytical and problem-solving skills.
 - Effective communication and stakeholder management.
 - **Impact:**
 - Ensures the project delivers business value and actionable outcomes.
-

6. Domain Expert

- **Role:** Provides specialized knowledge about the industry or subject area relevant to the project.
 - **Responsibilities:**
 - Define the context and relevance of data.
 - Validate assumptions and interpret results from a domain perspective.
 - Offer insights to refine analytical approaches.
 - **Skills:**
 - Deep knowledge of the specific industry or field.
 - Ability to collaborate with technical teams.
 - **Impact:**
 - Ensures that analytics solutions are practical and applicable in the real-world context.
-

7. Data Analyst

- **Role:** Focuses on querying, reporting, and visualizing data to generate insights.
 - **Responsibilities:**
 - Perform exploratory data analysis (EDA) and basic statistical analyses.
 - Create dashboards and reports for stakeholders.
 - Monitor trends and anomalies in data.
 - **Skills:**
 - Proficiency in tools like Excel, Tableau, Power BI, or SQL.
 - Strong visualization and storytelling capabilities.
 - **Impact:**
 - Provides immediate, actionable insights and supports decision-making.
-

8. IT Specialist

- **Role:** Ensures the necessary technical infrastructure and tools are available.
 - **Responsibilities:**
 - Manage hardware, software, and network requirements.
 - Ensure system reliability, security, and compliance.
 - Support the deployment and scaling of analytics solutions.
 - **Skills:**
 - Expertise in IT systems and infrastructure management.
 - Knowledge of cybersecurity and compliance standards.
 - **Impact:**
 - Provides a stable and secure environment for analytics processes.
-

9. Visualization Expert

- **Role:** Focuses on presenting data in a visually appealing and understandable format.
- **Responsibilities:**
 - Design and create dashboards, charts, and infographics.
 - Tailor visualizations for diverse audiences.
 - Optimize interactivity and user experience in data presentations.
- **Skills:**
 - Expertise in visualization tools and design principles.

3.3] Various phases of Data analytic lifecycle

- **Phase 1: Discovery –**

- The data science team learns and investigates the problem.
- Develop context and understanding.
- Come to know about data sources needed and available for the project.
- The team formulates the initial hypothesis that can be later tested with data.

- **Phase 2: Data Preparation –**

- Steps to explore, pre-process, and condition data before modelling and analysis.
- It requires the presence of an analytic sandbox, the team executes, loads, and transforms, to get data into the sandbox.
- Data preparation tasks are likely to be performed multiple times and not in predefined order.
- Several tools commonly used for this phase are – Hadoop, Alpine Miner, Open Refine, etc.

- **Phase 3: Model Planning –**

- The team explores data to learn about relationships between variables and subsequently, selects key variables and the most suitable models.
- In this phase, the data science team develops data sets for training, testing, and production purposes.
- Team builds and executes models based on the work done in the model planning phase.
- Several tools commonly used for this phase are – Matlab and STASTICA.

- **Phase 4: Model Building –**

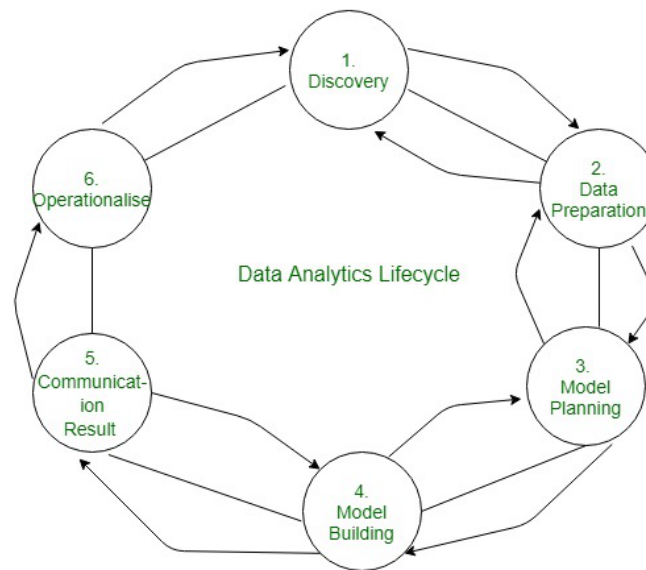
- Team develops datasets for testing, training, and production purposes.
- Team also considers whether its existing tools will suffice for running the models or if they need more robust environment for executing models.
- Free or open-source tools – R and PL/R, Octave, WEKA.
- Commercial tools – Matlab and STASTICA.

- **Phase 5: Communication Results –**

- After executing model team need to compare outcomes of modelling to criteria established for success and failure.
- Team considers how best to articulate findings and outcomes to various team members and stakeholders, taking into account warning, assumptions.
- Team should identify key findings, quantify business value, and develop narrative to summarize and convey findings to stakeholders.

- **Phase 6: Operationalize –**

- The team communicates benefits of project more broadly and sets up pilot project to deploy work in controlled way before broadening the work to full enterprise of users.
- This approach enables team to learn about performance and related constraints of the model in production environment on small scale which make adjustments before full deployment.
- The team delivers final reports, briefings, codes.
- Free or open source tools – Octave, WEKA, SQL, MADlib.



***what is Data Analytics Life Cycle?

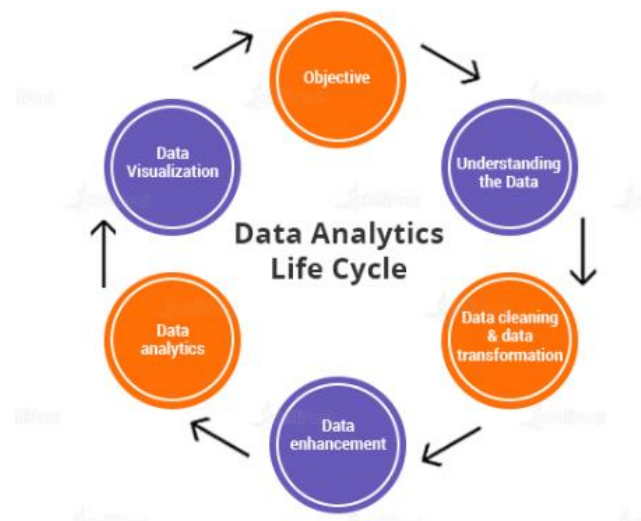
Data is precious in today's digital environment. It goes through several life stages, including creation, testing, processing, consumption, and reuse. These stages are mapped out in the Data Analytics Life Cycle for professionals working on data analytics initiatives. Each stage has its own significance and characteristics.

- [Importance of Data Analytics Life Cycle](#)
- [Data Analytics Life Cycle Phases](#)
- [Phase 1: Data Discovery and Formation](#)
- [Phase 2: Data Preparation and Processing](#)
- [Phase 3: Design a Model](#)
- [Phase 4: Model Building](#)
- [Phase 5: Result Communication and Publication](#)

- [Phase 6: Measuring Effectiveness](#)

Importance of Data Analytics Life Cycle

The data analytics Life Cycle encompasses the process of producing, collecting, processing, using, and analyzing data in order to meet corporate objectives. [Data Analytics](#) offers a systematic way for managing data into useful information that can help achieve organizational or project goals; additionally, it provides guidance and strategies for extracting this information and moving in the appropriate direction in order to meet corporate objectives

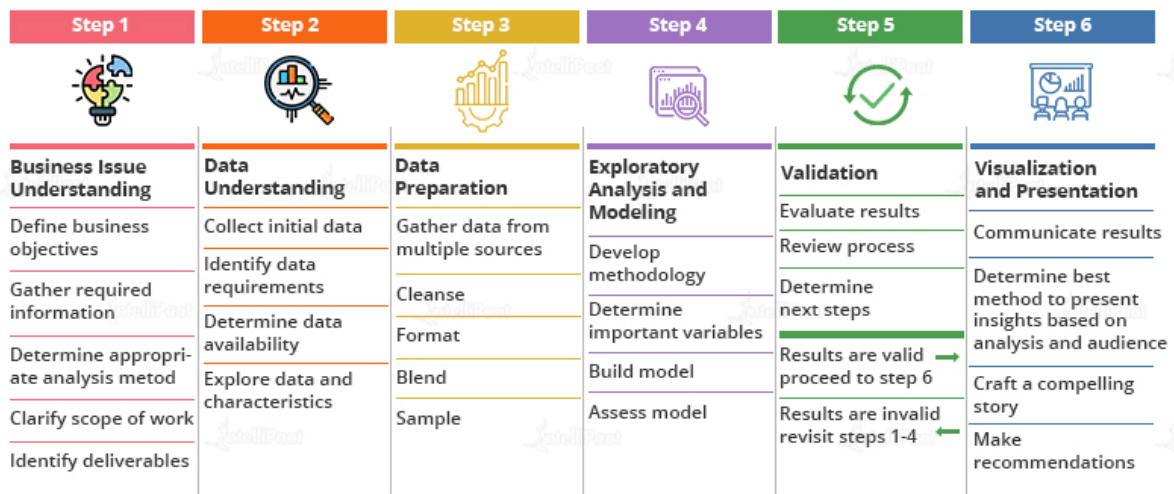


Data professionals use the circular nature of the Life Cycle to go ahead or backward with data analytics. Based on the new information, they can decide whether to continue with their current research or abandon it and redo the entire analysis. Throughout the process, they are guided by the Data Analytics Life Cycle.

Data Analytics Life Cycle Phases:

LIFE CYCLE OF DATA ANALYSIS PROJECT

Based on CRISP-DM Methodology



The scientific method for creating a structured framework of the data analytics life cycle involves six stages of architecture for data analytics. The framework is direct and cyclical, meaning all big data analytics-related processes must be completed sequentially.

Notably, these phases are circular; therefore they may be undertaken either forwards or backwards. Below are six data analytics phases that serve as fundamental processes in data science projects.

Phase 1: Data Discovery and Formation

Every good journey begins with a purpose in mind. In this phase, you will identify your desired data objectives and how best to attain them through data analytics Life Cycle implementation. Evaluations and assessments should also be undertaken during this initial phase to develop a basic hypothesis capable of solving business issues or problems.

In the initial step, data will be evaluated for its potential uses and demands – such as where it comes from, what message you wish for it to send and how this incoming information benefits your business.

As a [data analyst](#), you will need to explore case studies using similar data analytics and, most crucially, examine current company trends. Then you must evaluate all in-house infrastructure and resources, as well as time and technological needs, in order to match the previously acquired data.

Following the completion of the evaluations, the team closes this stage with hypotheses that will be tested using data later on. This is the first and most critical step in the life cycle of big data analytics.

Key takeaways:

- The data science team investigates and learns about the challenge.
- Create context and understanding.
- Learn about the data sources that will be required and available for the project.
- The team develops preliminary hypotheses that can later be tested with data.

Phase 2: Data Preparation and Processing

Data preparation and processing involves gathering, sorting, processing and purifying collected information to make sure it can be utilized by subsequent steps of analysis. An important element of this step is making sure all necessary information is readily accessible before moving ahead with processing it further.

Following are methods of data acquisition

- **Data Collection:** Draw information from external sources.
- **Data Entry:** Within an organization, data entry refers to creating new points of information using either digital technologies or manual input procedures.
- **Signal Reception:** Accumulating data from digital devices like the Internet of Things devices and control systems.

An analytical sandbox is essential during the data preparation stage of data analytics Life Cycle. This scalable platform is used by data analysts and scientists alike for processing their data sets; once executed, loaded, or altered it resides securely inside this sandbox for later examination and modification.

This phase of the analytical cycle does not need to take place in any particular order; rather it can take place as necessary and be repeated at later times as appropriate.

Phase 3: Design a Model

After you've defined your business goals and gathered a large amount of data (formatted, unformatted, or semi-formatted), it's time to create a model that uses the data to achieve the goal. Model planning is the name given to this stage of the data analytics process.

There are numerous methods for loading data into the system and starting to analyze it:

- ETL (Extract, Transform, and Load) converts the information before loading it into a system using a set of business rules.
- ELT (Extract, Load, and Transform) loads raw data into the sandbox before transforming it.
- ETLT (Extract, Transform, Load, Transform) is a combination of two layers of transformation.

This step also involves teamwork to identify the approaches, techniques, and workflow to be used in the succeeding phase to develop the model. The process of developing a model begins with finding the relationship between data points to choose the essential variables and, subsequently, create a suitable model.

Phase 4: Model Building

This stage of the data analytics life cycle involves creating datasets for testing, training, and production. The data analytics professionals develop and operate the model they designed in the previous stage with proper effort.

They use tools and methods, such as decision trees, regression techniques logistic regression), and neural networks to create and run the model. The experts also run the model through a trial run to see if it matches the datasets.

It assists them in determining whether the tools they now have will be enough to execute the model or if a more robust system is required for it to function successfully.

Key Takeaways:

- The team creates datasets for use in testing, training, and production.
- The team also examines if its present tools will serve for running the models or if a more robust environment is required for model execution.

- Rand PL/R, Octave, and WEKA are examples of free or open-source tools.

Phase 5: Result Communication and Publication

Recall the objective you set for your company in phase 1. Now is the time to see if the tests you ran in the previous phase matched those criteria.

The communication process begins with cooperation with key stakeholders to decide whether the project's outcomes are successful or not.

The project team is responsible for identifying the major conclusions of the analysis, calculating the business value associated with the outcome, and creating a narrative to summarize and communicate the results to stakeholders.

Phase 6: Measuring Effectiveness

As your data analytics life cycle comes to an end, the final stage is to offer stakeholders a complete report that includes important results, coding, briefings, and technical papers or documents.

Furthermore, to assess the effectiveness of the study, the data is transported from the sandbox to a live environment and observed to see if the results match the desired business aim.

If the findings meet the objectives, the reports and outcomes are finalized. However, if the conclusion differs from the purpose stated in phase 1, then you can go back in the data analytics life cycle to any of the previous phases to adjust your input and get a different result.

Unit IV: Machine Learning: Supervised Learning

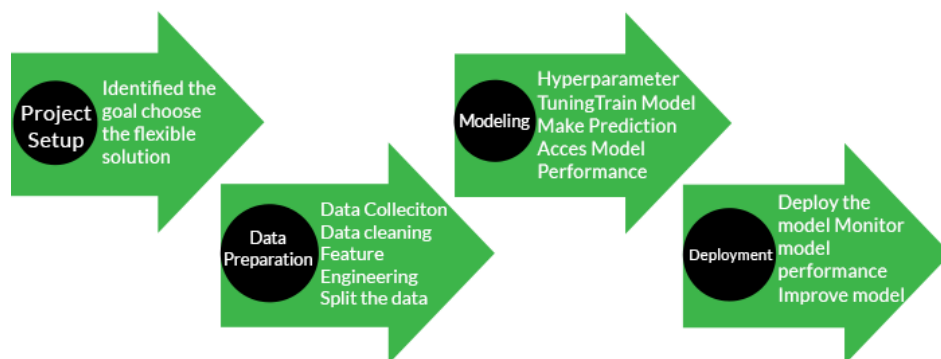
4.1] what is Machine Learning?

Machine Learning, often abbreviated as **ML** is a branch of [Artificial Intelligence \(AI\)](#) that works on algorithm developments and statistical models that allow computers to learn from data and make predictions or decisions without being explicitly programmed.

Hence, in simpler terms, **machine learning allows computers to learn from data and make decisions or predictions without being explicitly programmed to do so**. Essentially, machine learning algorithms learn patterns and relationships from data, allowing them to generalize from instances and make predictions or conclusions on new and uncovered data.

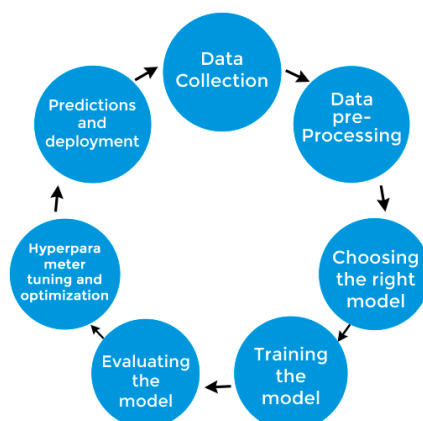
How does Machine Learning Work?

Broadly Machine Learning process includes Project Setup, Data Preparation, Modelling and Deployment. The following figure demonstrates the common working process of Machine Learning. It follows some set of steps to do the task; a sequential process of its workflow is as follows -



Stages of Machine Learning

A detailed sequential process of Machine Learning includes some set of steps of phases which are as –



1. Data collection: Data collection is an initial step in the process of machine learning. Data is a fundamental part of machine learning, the quality and quantity of your data can have direct consequences for model performance. Different sources such as databases, text files, pictures, sound files, or web scraping may be used for data collection. Data needs to be prepared for machine learning once it has been collected. This process is to organize the data in an appropriate format, such as a CSV file or database, and make sure that they are useful for solving your problem.

2. Data pre-processing: Pre-processing of data is a key step in the process of machine learning. It involves deleting duplicate data, fixing errors, managing missing data either by eliminating or filling it in, and adjusting and formatting the data. Pre-processing improves the quality of your data and ensures that your machine-learning model can read it right. The accuracy of your model may be significantly improved by this step.

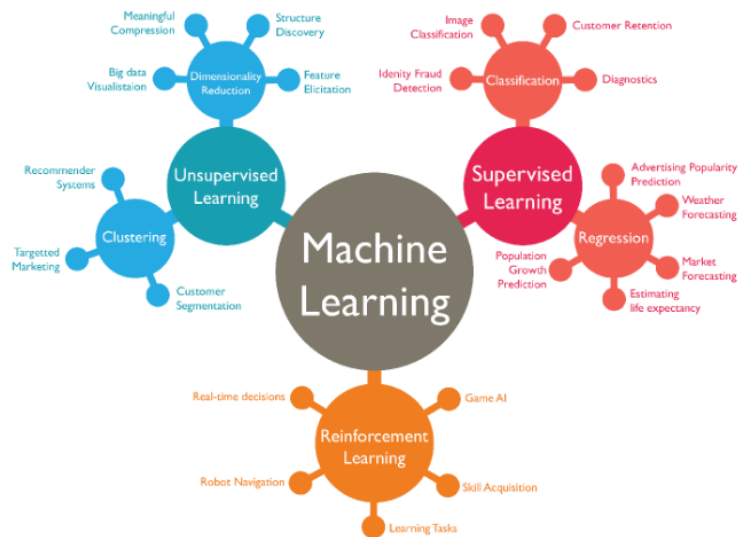
3. Choosing the right model: The next step is to select a machine learning model; once data is prepared then we apply it to ML Models like Linear regression, decision trees, and Neural Networks that may be selected to implement. The selection of the model generally depends on what kind of data you're dealing with and your problem. The size and type of data, complexity, and computational resources should be taken into account when choosing a model to apply.

4. Training the model: The next step is to train it with the data that has been prepared after you have chosen a model. Training is about connecting the data to the model and enabling it to adjust its parameters to predict output more accurately. Over fitting and underfitting must be avoided during the training.

5. Evaluating the model: It is important to assess the model's performance before deployment as soon as a model has been trained. This means that the model has to be tested on new data that they haven't been able to see during training. Accuracy in classifying problems, precision and recall for binary classification problems, as well as mean error squared with regression problems, are common metrics to evaluate the performance of a model.

6. Hyperparameter tuning and optimization: You may need to adjust its hyperparameters to make it more efficient after you've evaluated the model. Grid searches, where you try different combinations of parameters, and cross-validation, where you divide your data into subsets and train your model on each subset, to ensure that it performs well on different data sets, are techniques for hyperparameter tuning.

7. Predictions and deployment: As soon as the model has been programmed and optimized, it will be ready to estimate new data. This is done by adding new data to the model and using its output for decision-making or other analysis. The deployment of this model involves its integration into a production environment where it is capable of processing real-world data and providing timely information.



4.2] Applications of Machine Learning:

Some of the successful machine learning applications are chatbots, language translation, face recognition, recommendation systems, autonomous vehicles, object detection, medical image analysis, etc. Here are some popular applications of machine learning –

- [Image and Speech Recognition](#)
- [Natural Language Processing](#)
- [Finance Sector](#)
- [E-commerce and Retail](#)
- [Automotive Sector](#)
- [Computer Vision](#)
- [Manufacturing and Industries](#)
- [Healthcare Sector](#)

1. Image and speech recognition are two areas where machine learning has significantly improved. Machine learning algorithms are used in applications such as facial recognition, object detection, and speech recognition to accurately identify and classify images and speech.

2. Natural Language Processing

[Natural Language Processing \(NLP\)](#) is a field of computer science that deals with the interaction between computers and humans using natural language. NLP uses machine learning algorithms to identify parts of speech, sentiment and other aspects of text. It analyzes, understands, and generates human language. It is currently all over the internet which includes translation software, search engines, chatbots, grammar correction software and voice assistants, etc.

Here is a list of some applications of machine learning in natural language processing –

- Sentiment Analysis

- Speech synthesis
- Speech recognition
- Text classification
- Chatbots
- Language translation
- Caption generation
- Document summarization
- Question answering
- Autocomplete in search engines

3. Finance Sector

The role of machine learning in finance is to maintain secure transactions. Also, in trading, the data is converted to information for the decision-making process. Some applications of machine learning in the finance sector are –

1. Fraud Detection

Machine learning is widely used in the finance industry for fraud detection. Fraud detection is a process of using a machine learning model to monitor transactions and understand patterns in the dataset to identify fraudulent and suspicious activities.

Machine learning algorithms can analyze vast amounts of transactional data to detect patterns and anomalies that may indicate fraudulent activity, helping to prevent financial losses and protect customers.

2. Algorithmic Trading

Machine learning algorithms are used to identify complex patterns in the large dataset to discover trading signals which might not be possible for humans.

Some other applications of machine learning in the finance sector are as follows –

- Stock market analysis and forecasting
- Credit risk assessment and management
- Security analysis and portfolio optimization
- Asset evaluation and management

4. E-commerce and Retail

Machine learning is used to enhance the business in e-commerce and retail sector through recommendation systems and target advertising which improve user experience. Machine learning makes the process of marketing easy by performing repetitive tasks. Some tasks where Machine learning is applied are:

1. Recommendation Systems

Recommendation systems are used to provide personalized recommendations to users based on their past behavior and preferences and previous interaction with the website. Machine learning algorithms are used to analyze user data and generate recommendations for products, services, and content.

2. Demand Forecasting

Companies use machine learning to understand the future demand for their product or services based on various factors like market trends, customer behavior and historical data regarding sales.

3. Customer Segmentation

Machine learning can be used to segment customers into particular groups with similar characteristics. The purpose of customer segmentation is to understand customer behavior and target them with personalized experience.

5. Automotive Sector

Who would have thought of a car that would move independently without driving? Machine learning enabled manufacturers to improve the performance of existing products and vehicles. One massive innovation is the development of **autonomous vehicles** also called drive less vehicles which can sense its environment and drive for itself passing the obstacles without human assistance. It uses machine learning algorithms for continuous analysis of the surroundings and predicting possible outcomes.

6. Computer Vision

Computer vision is an application of machine learning that uses algorithms and neural networks to teach computers to derive meaningful information from digital images and videos. Computer vision is applied in face recognition, to diagnose diseases based on MRI scans, and autonomous vehicles.

- Object detection and recognition
- Image classification and recognition
- Facial recognition
- Autonomous vehicles
- Object segmentation
- Image reconstruction

Manufacturing and Industries

Machine learning is also used in manufacturing and industries to keep a check on the working conditions of machinery. **Predictive Maintenance** is used to identify defects in operational

machines and equipment to avoid unexpected outages. This detection of anomalies would also help with regular maintenance.

Predictive maintenance is a process of using machine learning algorithms to predict when maintenance will be required on a machine, such as a piece of equipment in a factory. By analyzing data from sensors and other sources, machine learning algorithms can detect patterns that indicate when a machine is likely to fail, enabling maintenance to be performed before the machine breaks down.

7. Healthcare Sector

Machine learning has also found many applications in the healthcare industry. For example, machine learning algorithms can be used to analyze medical images and detect diseases such as cancer or to predict patient outcomes based on their medical history and other factors.

Some applications of machine learning in healthcare are discussed below –

1. Medical Imaging and Diagnostics

Machine learning in medical imaging is used to analyze the patterns in the image that indicate the presence of a particular disease.

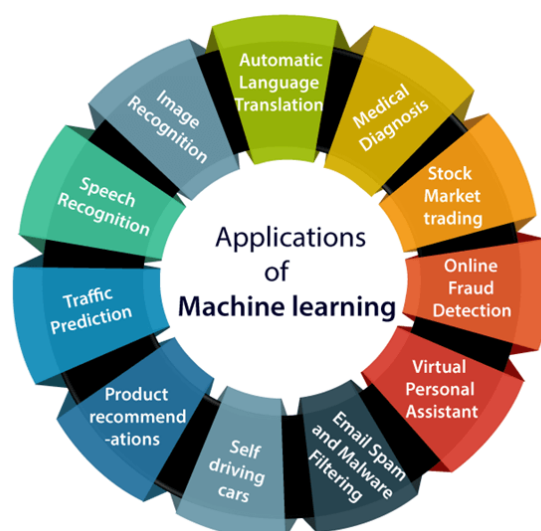
2. Drug Discovery

Machine learning techniques are used to analyze vast datasets, to predict the biological activity of compounds, and to identify potential drugs for a disease by analyzing its chemical structures.

3. Disease Diagnosis

Machine learning may also be used to identify some types of diseases. Breast cancer, heart failure, Alzheimer's disease, and pneumonia are some examples of such diseases that can be identified using machine learning algorithms.

These are just a few examples of the many applications of machine learning. As machine learning continues to evolve and improve, we can expect to see it used in more areas of our lives, improving efficiency, accuracy, and convenience in a variety of industries.



4.3] Supervised Learning:

Supervised learning, also known as **supervised machine learning**, is a type of machine learning that trains the model using **labelled datasets** to predict outcomes. A Labelled dataset is one that consists of input data (features) along with corresponding output data (targets).

The main objective of supervised learning algorithms is to learn an association between input data samples and corresponding outputs after performing multiple training data instances.

How does Supervised Learning Work?

In supervised machine learning, models are trained using a dataset that consists of input-output pairs.

The supervised learning algorithm analyzes the dataset and learns the relation between the input data (features) and correct output (labels/ targets). In the process of training, the model estimates the algorithm's parameters by minimizing a loss function. The loss function measures the difference between the model's predictions and actual target values.

The model iteratively updates its parameters until the loss/ error has been sufficiently minimized.

Once the training is completed, the model parameters have optimal values. The model has learned the optimal mapping/ relation between the inputs and targets. Now, the model can predict values for the new and unseen input data.

Types of Supervised Learning Algorithm:

Supervised machine learning is categorized into two types of problems – classification and regression.

1. Classification

The key objective of classification-based tasks is to predict categorical output labels or responses for the given input data such as true-false, male-female, yes-no etc. As we know, the categorical output responses mean unordered and discrete values; hence, each output response will belong to a specific class or category.

Some popular classification algorithms are decision trees, random forests, support vector machines (SVM), logistic regression, etc.

2. Regression

The key objective of regression-based tasks is to predict output labels or responses, which are continuous numeric values, for the given input data. Basically, regression models use the input data features (independent variables) and their corresponding continuous numeric output values (dependent or outcome variables) to learn specific associations between inputs and corresponding outputs.

Some popular regression algorithms are linear regression, polynomial regression, Lasso regression, etc.

Algorithms for Supervised Learning:

Supervised learning is one of the important models of learning involved in training machines. This chapter talks in detail about the same.

There are several algorithms available for supervised learning. Some of the widely used algorithms of supervised learning are as shown below –

- [Linear Regression](#)
- [k-Nearest Neighbours](#)
- [Decision Trees](#)
- [Naive Bayes](#)
- [Logistic Regression](#)
- [Support Vector Machines](#)
- [Random Forest](#)
- [Gradient Boosting](#)

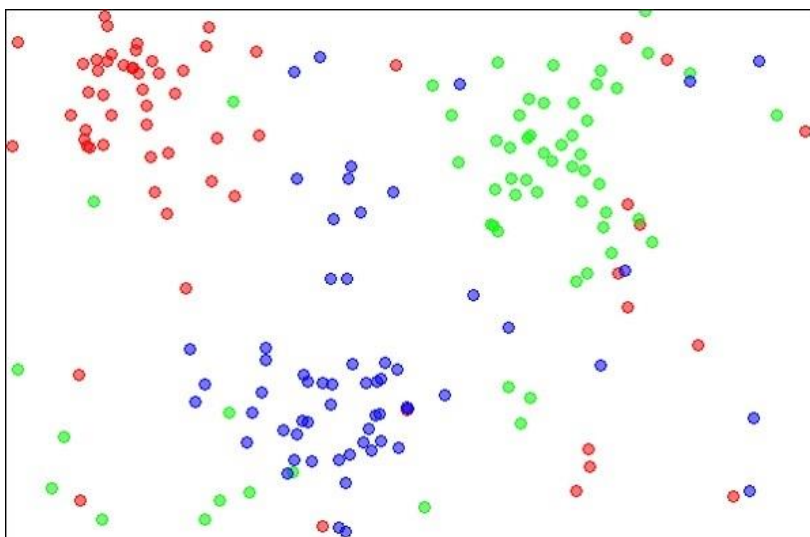
1. Linear Regression

[Linear regression](#) is a type of algorithm that tries to find the linear relation between input features and output values for the prediction of future events. This algorithm is widely used to perform stock analysis, weather forecasting and others.

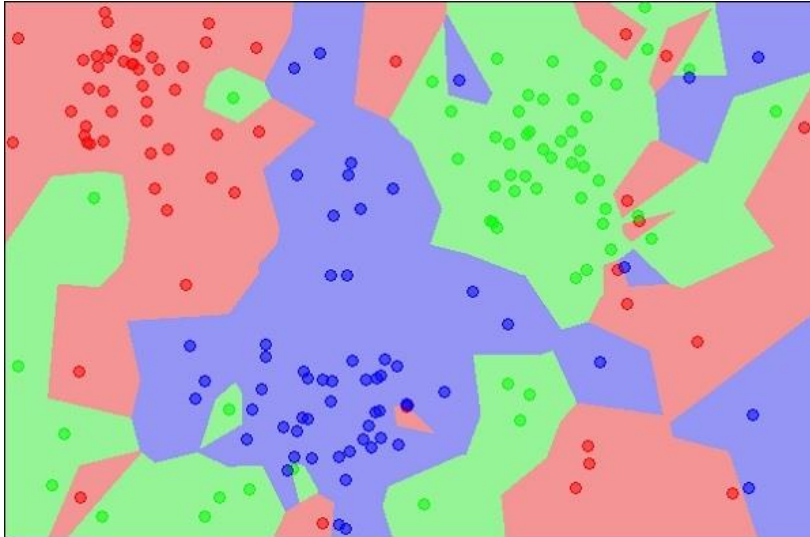
2. K-Nearest Neighbours:

The [k-Nearest Neighbours \(kNN\)](#) is a statistical technique that can be used for solving classification and regression problems. This algorithm classifies or predicts values for new data by mathematically calculating the nearest distance with other points in training data.

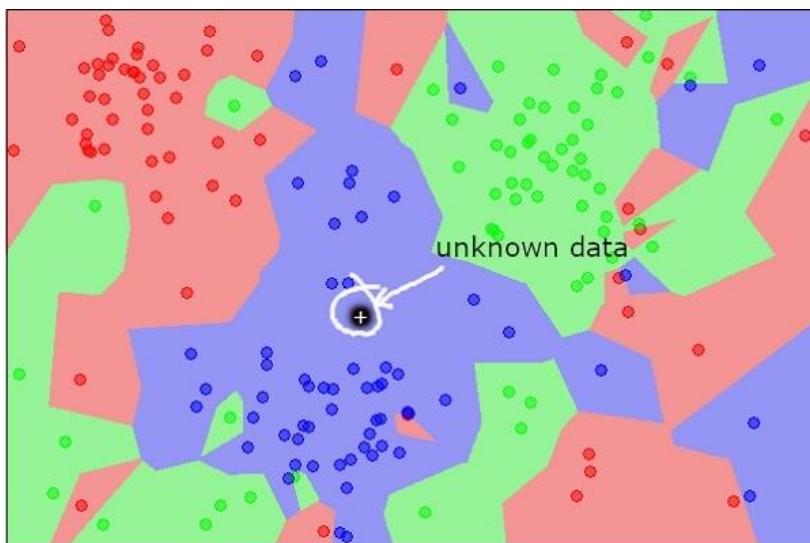
Let us discuss the case of classifying an unknown object using kNN. Consider the distribution of objects as shown in the image given below –



The diagram shows three types of objects, marked in red, blue and green colors. When you run the kNN classifier on the above dataset, the boundaries for each type of object will be marked as shown below –



Now, consider a new unknown object you want to classify as red, green or blue. This is depicted in the figure below.



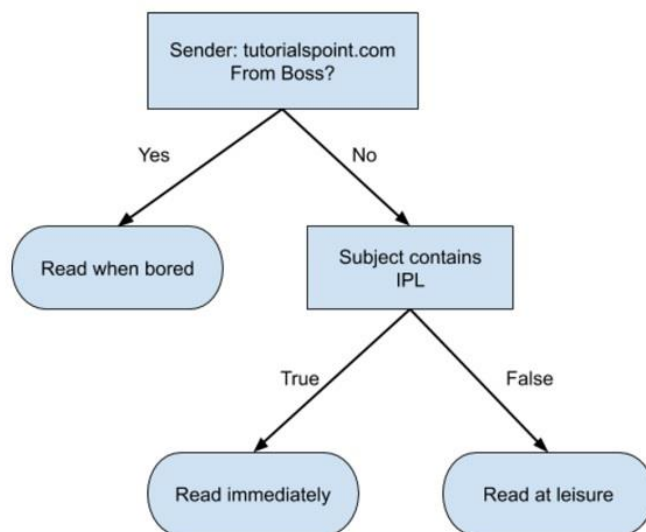
As you see it visually, the unknown data point belongs to a class of blue objects. Mathematically, this can be concluded by measuring the distance of this unknown point with every other point in the data set. When you do so, you will know that most of its neighbours are blue in colour. The average distance between red and green objects would definitely be more than the average distance between blue objects. Thus, this unknown object can be classified as belonging to blue class.

The kNN algorithm can also be used for regression problems. The kNN algorithm is available as ready-to-use in most of the ML libraries.

3. Decision Trees

A [Decision tree](#) is a tree-like structure used to make decisions and analyze the possible consequences. The algorithm splits the data into subsets based on features, where each parent node represents internal decisions and the leaf node represents final prediction.

A simple decision tree in a flowchart format is shown below –



You would write a code to classify your input data based on this flowchart. The flowchart is self-explanatory and trivial. In this scenario, you are trying to classify an incoming email to decide when to read it.

In reality, the decision trees can be large and complex. There are several algorithms available to create and traverse these trees. As a Machine Learning enthusiast, you need to understand and master these techniques of creating and traversing decision trees.

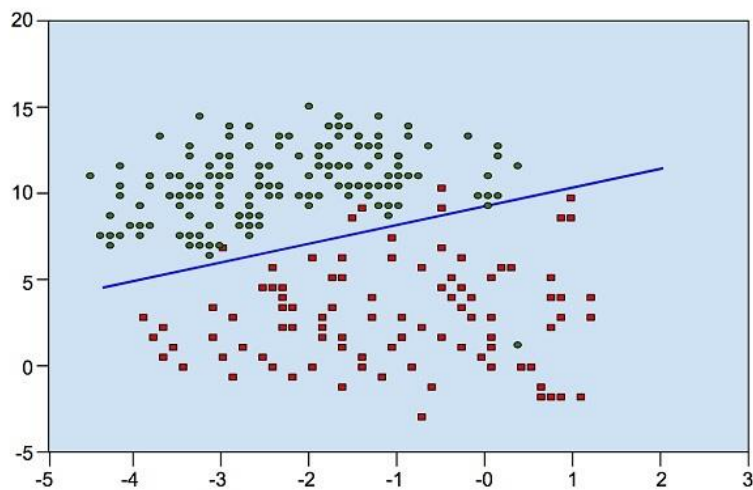
4. Naive Bayes

[Naive Bayes](#) is used for creating classifiers. Suppose you want to sort out (classify) fruits of different kinds from a fruit basket. You may use features such as colour, size, and shape of fruit; for example, any fruit that is red in colour, round in shape, and about 10 cm in diameter may be considered an Apple. So to train the model, you would use these features and test the probability that a given feature matches the desired constraints. The probabilities of different features are then combined to arrive at the probability that a given fruit is an Apple. Naive Bayes generally requires a small number of training data for classification.

5. Logistic Regression

[Logistic regression](#) is a type of statistical algorithm that estimates the probability of occurrence of an event.

Look at the following diagram. It shows the distribution of data points in the XY plane.

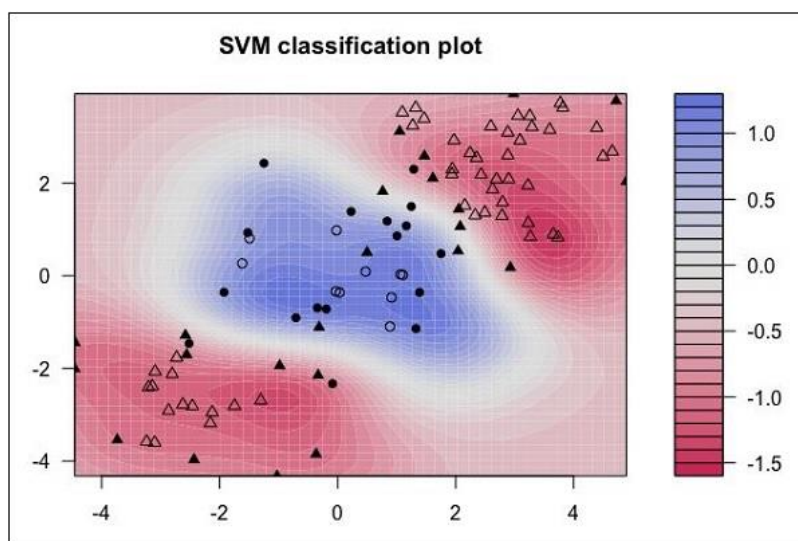


From the diagram, we can visually inspect the separation of red and green dots. You may draw a boundary line to separate out these dots. Now, to classify a new data point, you will just need to determine on which side of the line the point lies.

6. Support Vector Machines

Support Vector Machines (SVM) algorithm can be typically used for both classification and regression. For classification tasks, the algorithm creates a hyperplane to separate data into classes. While for regression, the algorithm tries to fit a regression line with minimal error.

Look at the following distribution of data. Here the three classes of data cannot be linearly separated. The boundary curves are non-linear. In such a case, finding the curve's equation becomes a complex job.

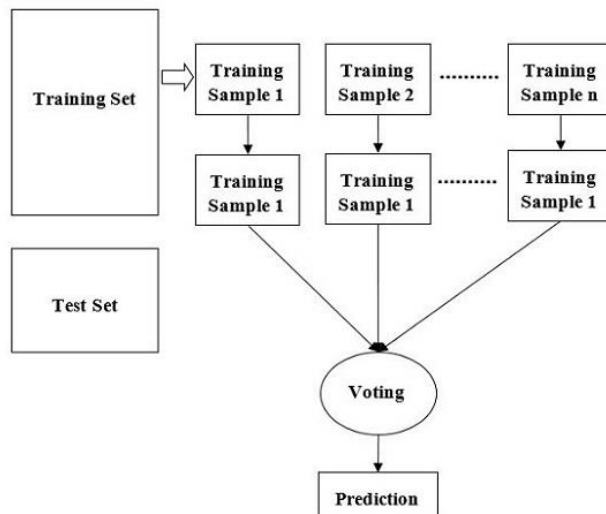


The Support Vector Machines (SVM) come in handy in determining the separation boundaries in such situations.

7. Random Forest

Random forest is also a supervised learning algorithm that is flexible for classification and regression. This algorithm is a combination of multiple decision trees which are merged to improve the accuracy of prediction .

The following diagram illustrates how the Random Forest Algorithm works –



8. Gradient Boosting

Gradient boosting combines weak learners(decision trees), to create a strong model. It builds new models that correct errors of the previous ones. The goal of this algorithm is to minimize the loss function. It can be efficiently used for classification and regression tasks.

Advantages of Supervised Learning:

Supervised learning algorithms are one of the most popular among the machine learning models. Some benefits are:-

- The goal in supervised learning is well-defined, which improves the prediction accuracy.
- Models trained using supervised learning are effective at predicting and classification since they use labelled datasets.
- It can be highly versatile, i.e., applied to various problems, like spam detection, stock prices, etc.

Disadvantages of Supervised Learning:

Though supervised learning is the most used, it comes with certain challenges too. Some of them are:

- Supervised learning requires a large amount of labelled data for the model to train effectively. It is practically very difficult to collect such huge data; it is expensive and time-consuming.
- Supervised learning cannot predict accurately if the test data is different from the training data.
- Accurately labelling the data is complex and requires expertise and effort.

Applications of Supervised learning:

Supervised learning models are widely used in many applications in various sectors, including the following-

- **Image recognition** – A model is trained on a labelled dataset of images, where each image is associated with a label. The model is fed with data, which allows it to learn patterns and features. Once trained, the model can now be tested using new, unseen data. This is widely used in applications like facial recognition and object detection.
- **Predictive analytics** – Supervised learning algorithms are used to train labelled historical data, allowing the model to learn patterns and relations between input features and output to identify trends and make accurate predictions. Businesses use this method to make data-driven decisions and enhance strategic planning.

4.4] Structure of RegressionModel:

[Regression](#) is a statistical approach used to analyze the relationship between a dependent variable (target variable) and one or more independent variables (predictor variables). The objective is to determine the most suitable function that characterizes the connection between these variables.

It seeks to find the best-fitting model, which can be utilized to make predictions or draw conclusions.

Regression in Machine Learning

It is a supervised machine learning technique, used to predict the value of the dependent variable for new, unseen data. It models the relationship between the input features and the target variable, allowing for the estimation or prediction of numerical values.

Regression analysis problem works with if output variable is a real or continuous value, such as “salary” or “weight”. Many different models can be used; the simplest is the linear regression. It tries to fit data with the best hyper-plane which goes through the points.

Terminologies Related to the Regression Analysis in Machine Learning

Terminologies Related to Regression Analysis:

- **Response Variable:** The primary factor to predict or understand in regression, also known as the dependent variable or target variable.
- **Predictor Variable:** Factors influencing the response variable, used to predict its values; also called independent variables.
- **Outliers:** Observations with significantly low or high values compared to others, potentially impacting results and best avoided.
- **Multicollinearity:** High correlation among independent variables, which can complicate the ranking of influential variables.

- **Underfitting and Overfitting:** Overfitting occurs when an algorithm performs well on training but poorly on testing, while underfitting indicates poor performance on both datasets.

4.5] Linear Regression:

Linear regression in machine learning is defined as a statistical model that analyzes the linear relationship between a dependent variable and a given set of independent variables. The linear relationship between variables means that when the value of one or more independent variables will change (increase or decrease), the value of the dependent variable will also change accordingly (increase or decrease).

In machine learning, linear regression is used for predicting continuous numeric values based on learned linear relation for new and unseen data. It is used in predictive modelling, financial forecasting, risk assessment, etc.

What is Linear Regression?

Linear regression is a statistical technique that estimates the linear relationship between a dependent and one or more independent variables. In machine learning, linear regression is implemented as a **supervised learning** approach. In machine learning, labelled datasets contain input data (features) and output labels (target values). For linear regression in machine learning, we represent features as independent variables and target values as the dependent variable.

For the simplicity, take the following data (Single feature and single target)

Square Feet (X)	House Price (Y)
1300	240
1500	320
1700	330
1830	295
1550	256
2350	409

In the above data, the target House Price is the dependent variable represented by X , and the feature, Square Feet, is the independent variable represented by Y . The input features (X) are used to predict the target label (Y). So, the independent variables are also known as predictor variables, and the dependent variable is known as the response variable.

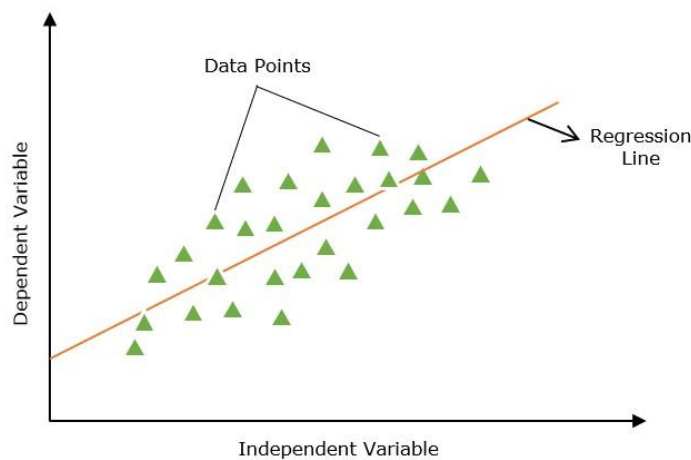
So let's define linear regression in machine learning as follows:

In machine learning, linear regression uses a linear equation to model the relationship between a dependent variable (Y) and one or more independent variables (X).

The main goal of the linear regression model is to find the best-fitting straight line (often called a regression line) through a set of data points.

Line of Regression

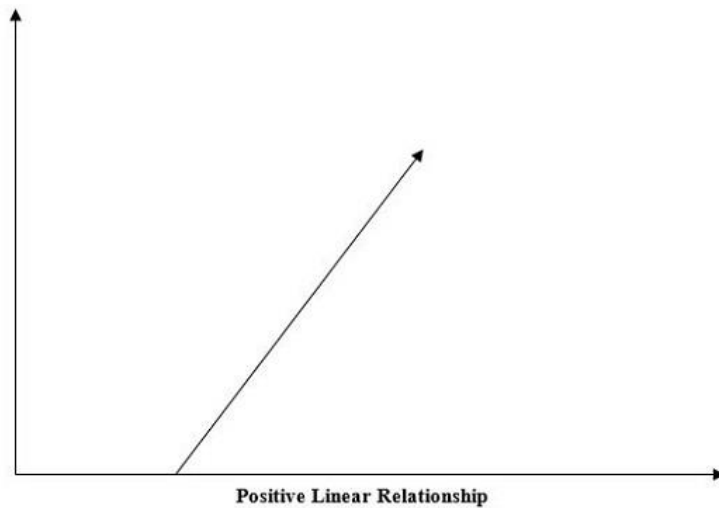
A straight line that shows a relation between the dependent variable and independent variables is known as the line of regression or regression line.



Furthermore, the linear relationship can be positive or negative in nature as explained below –

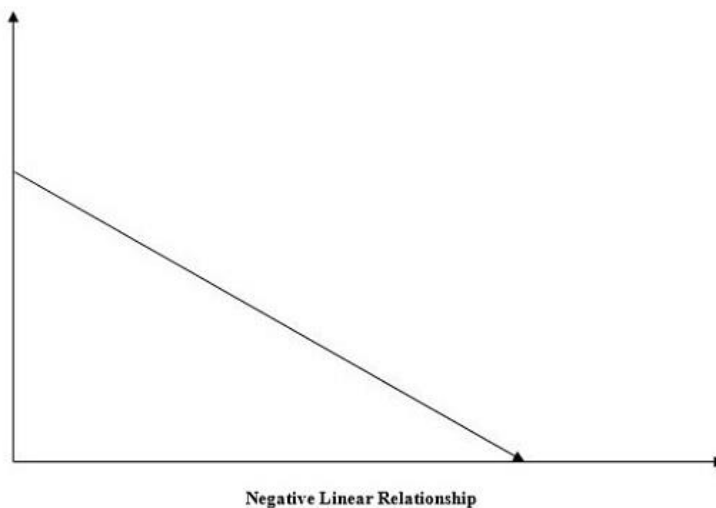
1. Positive Linear Relationship

A linear relationship will be called positive if both independent and dependent variable increases. It can be understood with the help of the following graph –



2. Negative Linear Relationship

A linear relationship will be called positive if the independent increases and the dependent variable decrease. It can be understood with the help of the following graph –



Linear regression is of two types, "simple linear regression" and "multiple linear regression", which we are going to discuss in the next two chapters of this tutorial.

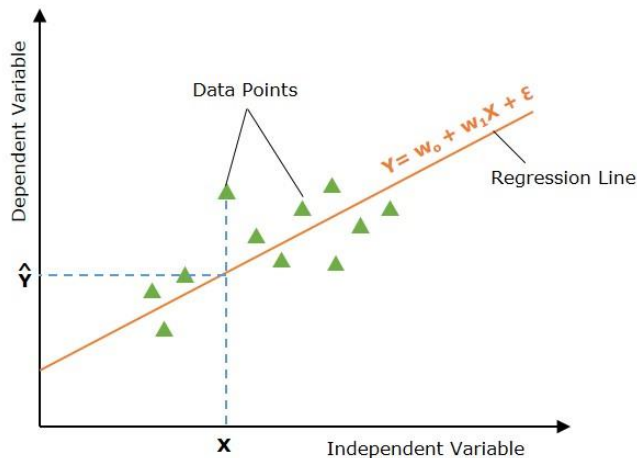
Types of Linear Regression

Linear regression is of the following two types –

- Simple Linear Regression
- Multiple Linear Regression

1. Simple Linear Regression

Simple linear regression is a type of regression analysis in which a single independent variable (also known as a predictor variable) is used to predict the dependent variable. In other words, it models the linear relationship between the dependent variable and a single independent variable.



In the above image, the straight line represents the simple linear regression line where $\hat{Y}$ is the predicted value, and X is the input value.

Mathematically, the relationship can be modelled as a linear equation –

$$Y = w_0 + w_1 X + \epsilon$$

Where

- Y is the dependent variable (target).
- X is the independent variable (feature).
- w_0 is the y-intercept of the line.
- w_1 is the slope of the line, representing the effect of X on Y .
- ϵ is the error term, capturing the variability in Y not explained by X .

2. Multiple Linear Regression

Multiple linear regression is basically the extension of simple linear regression that predicts a response using two or more features.

When dealing with more than one independent variable, we extend simple linear regression to multiple linear regression. The model is expressed as:

Multiple linear regression extends the concept of simple linear regression to multiple independent variables. The model is expressed as:

$$Y = w_0 + w_1X_1 + w_2X_2 + \dots + w_pX_p + \epsilon$$

Where

- $X_1, X_2, \dots, X_p$ are the independent variables (features).
- $w_0, w_1, \dots, w_p$ are the coefficients for these variables.
- ϵ is the error term.

How Does Linear Regression Work?

The main goal of linear regression is to find the best-fit line through a set of data points that minimizes the difference between the actual values and predicted values. So it is done? This is done by estimating the parameters w_0, w_1 etc.

The working of linear regression in machine learning can be broken down into many steps as follows –

- **Hypothesis** – We assume that there is a linear relation between input and output.
- **Cost Function** – Define a loss or cost function. The cost function quantifies the model's prediction error. The cost function takes the model's predicted values and actual values and returns a single scalar value that represents the cost of the model's prediction.
- **Optimization** – Optimize (minimize) the model's cost function by updating the model's parameters.

It continues updating the model's parameters until the cost or error of the model's prediction is optimized (minimized).

Hypothesis Function For Linear Regression

In linear regression problems, we assume that there is a linear relationship between input features (X) and predicted value ($\hat{Y}$).

The [hypothesis function](#) returns the predicted value for a given input value. Generally we represent a hypothesis by $h_w(X)$ and it is equal to $\hat{Y}$.

Hypothesis function for simple linear regression –

$$\hat{Y} = w_0 + w_1X$$

Hypothesis function for multiple linear regression –

$$\hat{Y} = w_0 + w_1X_1 + w_2X_2 + \dots + w_pX_p$$

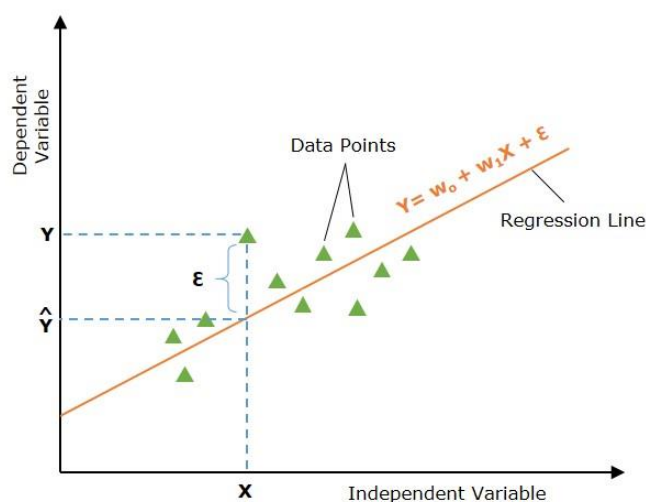
For different values of parameters (weights), we can find many regression lines. The main goal is to find the best-fit lines. Let's discuss it as below –

Finding the Best Fit Line

We discussed above that different set of parameters will provide different regression lines. However, each regression line does not represent the optimal relation between the input and output values. The main goal is to find the best-fit line.

A regression line is said to be the best fit if the error between actual and predicted values is minimal.

Below image shows a regression line with error (ϵ) at input data point X. The error is calculated for all data points and our goal is to minimize the average error/ loss. We can use different types of loss functions such as mean square error (MSE), mean average error (MAE), L_1 loss, L_2 Loss, etc.



So, how can we minimize the error between the actual and predicted values? Let's discuss the important concept, which is cost function or loss function.

Loss Function for Linear Regression

The error between actual and predicted values can be quantified using a loss function of the cost function. The [cost function](#) takes the model's predicted values and actual values and returns a single scalar value that represents the cost of the model's prediction. Our main goal is to minimize the cost function.

The most commonly used cost function is the mean squared error function.

$$J(w_0, w_1) = \frac{1}{2n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

Where,

- n is the number of data points.
- Y_i is the observed value for the i -th data point.
- $\hat{Y}_i = w_0 + w_1 X_i$ is the predicted value for the i -th data point.

Gradient Descent for Optimization

Now we have defined our loss function. The next step is to minimize it and find the optimized values of the parameters or weights. This process of finding optimal values of parameters such that the loss or error is minimal is known as model optimization.

Gradient Descent is one of the most used optimization techniques for linear regression.

To find the optimal values of parameters, gradient descent is often used, especially in cases with large datasets. Gradient descent iteratively adjusts the parameters in the direction of the steepest descent of the cost function.

The parameter updates are given by

$$w_0 = w_0 - \alpha \frac{\partial J}{\partial w_0} \quad w_0 = w_0 - \alpha \frac{\partial J}{\partial w_0}$$

$$w_1 = w_1 - \alpha \frac{\partial J}{\partial w_1} \quad w_1 = w_1 - \alpha \frac{\partial J}{\partial w_1}$$

Where α is the learning rate, and the partial derivatives are:

$$\frac{\partial J}{\partial w_0} = -\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i) \quad \frac{\partial J}{\partial w_0} = -\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)$$

$$\frac{\partial J}{\partial w_1} = -\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i) X_i \quad \frac{\partial J}{\partial w_1} = -\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i) X_i$$

These gradients are used to update the parameters until convergence is reached (i.e., when the changes in w_0 and w_1 become negligible).

Assumptions of Linear Regression

The following are some assumptions about the dataset that are made by the Linear Regression model –

Multi-collinearity – Linear regression model assumes that there is very little or no multi-collinearity in the data. Basically, multi-collinearity occurs when the independent variables or features have a dependency on them.

Auto-correlation – Another assumption the Linear regression model assumes is that there is very little or no auto-correlation in the data. Basically, auto-correlation occurs when there is dependency between residual errors.

Relationship between variables – Linear regression model assumes that the relationship between response and feature variables must be linear.

Violations of these assumptions can lead to biased or inefficient estimates. It is essential to validate these assumptions to ensure model accuracy.

Evaluation Metrics for Linear Regression

To assess the performance of a linear regression model, several evaluation metrics are used –

R-squared (R^2) – It measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \quad R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

Mean Squared Error (MSE) – It measures an average of the sum of the squared difference between the predicted values and the actual values.

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) – It measures the square root of the MSE.

$$RMSE = \sqrt{MSE} \quad RMSE = \sqrt{MSE}$$

Mean Absolute Error (MAE) – It measures the average of the sum of the absolute values of the difference between the predicted values and the actual values.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Applications of Linear Regression:

1. Predictive Modelling

Linear regression is widely used for predictive modelling. For instance, in real estate, predicting house prices based on features such as size, location, and number of bedrooms can help buyers, sellers, and real estate agents make informed decisions.

2. Feature Selection

In multiple linear regressions, analyzing the coefficients can help in feature selection. Features with small or zero coefficients might be considered less important and can be dropped to simplify the model.

3. Financial Forecasting

In finance, linear regression models predict stock prices, economic indicators, and market trends. Accurate forecasts can guide investment strategies and financial planning.

4. Risk Management

Linear regression helps in risk assessment by modelling the relationship between risk factors and financial metrics. For example, in insurance, it can model the relationship between policyholder characteristics and claim amounts.

Advantages of Linear Regression

- **Interpretability** – Linear regression is easy to understand, which is useful when explaining how a model makes decisions.
- **Speed** – Linear regression is faster to train than many other machine learning algorithms.
- **Predictive analytics** – Linear regression is a fundamental building block for predictive analytics.
- **Linear relationships** – Linear regression is a powerful statistical method for finding linear relationships between variables.
- **Simplicity** – Linear regression is simple to implement and interpret.
- **Efficiency** – Linear regression is efficient to compute.

Common Challenges with Linear Regression

1. Overfitting

Overfitting occurs when the regression model performs well on training data but lacks generalization on test data. Overfitting leads to poor prediction on new, unseen data.

2. Multicollinearity

When the dependent variables (predictor or feature variables) correlate, the situation is known as multicollinearity. In this, the estimates of the parameters (coefficients) can be unstable.

3. Outliers and Their Impact

Outliers can cause the regression line to be a poor fit for the majority of data

4.6] Logistics Regression:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts $P(Y=1)$ as a function of X . It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Types of Logistic Regression

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those numbers of categories, Logistic regression can be divided into following types –

Binary or Binomial

In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

Multinomial

In such a kind of classification, dependent variable can have 3 or more possible unordered types or the types having no quantitative significance. For example, these variables may represent "Type A" or "Type B" or "Type C".

Ordinal

In such a kind of classification, dependent variable can have 3 or more possible ordered types or the types having a quantitative significance. For example, these variables may represent "poor" or "good", "very good", "Excellent" and each category can have the scores like 0,1,2,3.

Logistic Regression Assumptions:

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same –

- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other .
- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.

Binary Logistic Regression Model

The simplest form of logistic regression is binary or binomial logistic regression in which the target or dependent variable can have only 2 possible types either 1 or 0. It allows us to model a relationship between multiple predictor variables and a binary/binomial target

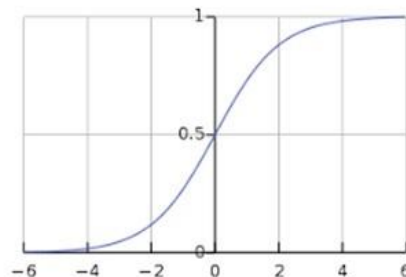
variable. In case of logistic regression, the linear function is basically used as an input to another function such as g in the following relation –

$$h\theta(x)=g(\theta^T x) \text{ where } 0 \leq h\theta \leq 1 \quad h\theta(x)=g(\theta^T x) \text{ where } 0 \leq h\theta \leq 1$$

Here, g is the logistic or sigmoid function which can be given as follows –

$$g(z)=\frac{1}{1+e^{-z}} \text{ where } z=\theta^T x \quad g(z)=\frac{1}{1+e^{-z}} \text{ where } z=\theta^T x$$

The sigmoid curve can be represented with the help of following graph. We can see the values of y-axis lie between 0 and 1 and crosses the axis at 0.5.



The classes can be divided into positive or negative. The output comes under the probability of positive class if it lies between 0 and 1. For our implementation, we are interpreting the output of hypothesis function as positive if it is ≥ 0.5 , otherwise negative.

We also need to define a loss function to measure how well the algorithm performs using the weights on functions, represented by theta as follows –

$$h=g(X\theta) \quad h=g(X\theta)$$

$$J(\theta)=\frac{1}{m} \sum_{i=1}^m (-y^i \log(h^i) - (1-y^i) \log(1-h^i)) \quad J(\theta)=\frac{1}{m} \sum_{i=1}^m (-y^i \log(h^i) - (1-y^i) \log(1-h^i))$$

Now, after defining the loss function our prime goal is to minimize the loss function. It can be done with the help of fitting the weights which means by increasing or decreasing the weights. With the help of derivatives of the loss function w.r.t each weight, we would be able to know what parameters should have high weight and what should have smaller weight.

The following gradient descent equation tells us how loss would change if we modified the parameters –

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m X_j^i (g(X\theta) - y^i) \quad \frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^m X_j^i (g(X\theta) - y^i)$$

Implementation of Binary Logistic Regression Model in Python

Now we will implement the above concept of binomial logistic regression in Python. For this purpose, we are using a multivariate flower dataset named 'iris' which have 3 classes of 50 instances each, but we will be using the first two feature columns. Every class represents a type of iris flower.

First, we need to import the necessary libraries as follows –

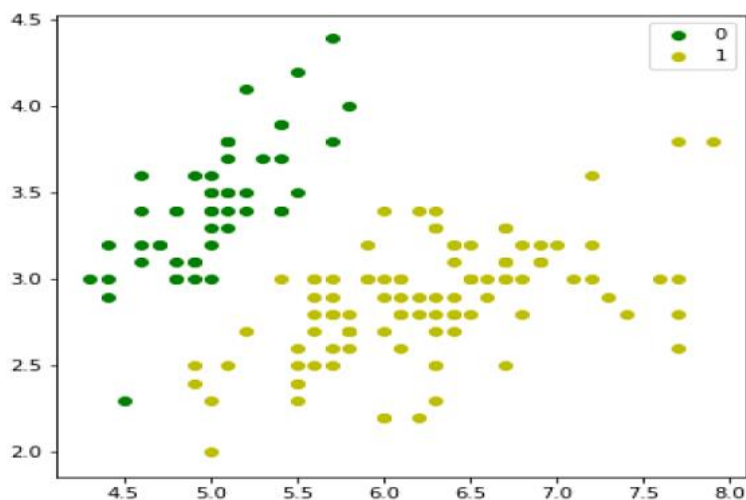
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
```

Next, load the iris dataset as follows –

```
iris = datasets.load_iris()
X = iris.data[:, :2]
y = (iris.target != 0) * 1
```

We can plot our training data s follows –

```
plt.figure(figsize=(6, 6))
plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], color='g', label='0')
plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], color='y', label='1')
plt.legend();
```



Next, we will define sigmoid function, loss function and gradient descend as follows –

```
class LogisticRegression:
    def __init__(self, lr=0.01, num_iter=100000, fit_intercept=True, verbose=False):
        self.lr = lr
        self.num_iter = num_iter
        self.fit_intercept = fit_intercept
        self.verbose = verbose
    def __add_intercept(self, X):
        intercept = np.ones((X.shape[0], 1))
```

```

        return np.concatenate((intercept, X), axis=1)
    def __sigmoid(self, z):
        return 1 / (1 + np.exp(-z))
    def __loss(self, h, y):
        return (-y * np.log(h) - (1 - y) * np.log(1 - h)).mean()
    def fit(self, X, y):
        if self.fit_intercept:
            X = self.__add_intercept(X)

```

Now, initialize the weights as follows –

```

self.theta = np.zeros(X.shape[1])
for i in range(self.num_iter):
    z = np.dot(X, self.theta)
    h = self.__sigmoid(z)
    gradient = np.dot(X.T, (h - y)) / y.size
    self.theta -= self.lr * gradient
    z = np.dot(X, self.theta)
    h = self.__sigmoid(z)
    loss = self.__loss(h, y)
    if(self.verbose == True and i % 10000 == 0):
        print(f'loss: {loss} \t')

```

With the help of the following script, we can predict the output probabilities –

```

def predict_prob(self, X):
    if self.fit_intercept:
        X = self.__add_intercept(X)
    return self.__sigmoid(np.dot(X, self.theta))
def predict(self, X):
    return self.predict_prob(X).round()

```

Next, we can evaluate the model and plot it as follows –

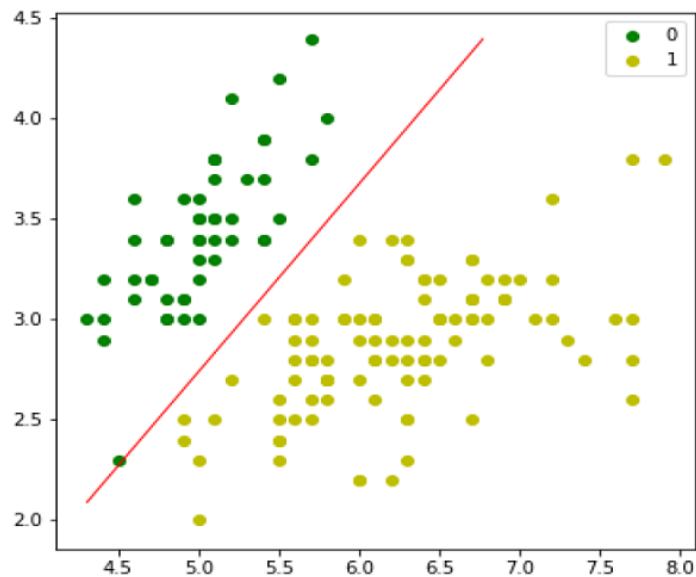
```

model = LogisticRegression(lr=0.1, num_iter=300000)
preds = model.predict(X)
(preds == y).mean()

plt.figure(figsize=(10, 6))
plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], color='g', label='0')

```

```
plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], color='y', label='1')
plt.legend()
x1_min, x1_max = X[:,0].min(), X[:,0].max(),
x2_min, x2_max = X[:,1].min(), X[:,1].max(),
xx1, xx2 = np.meshgrid(np.linspace(x1_min, x1_max), np.linspace(x2_min, x2_max))
grid = np.c_[xx1.ravel(), xx2.ravel()]
probs = model.predict_prob(grid).reshape(xx1.shape)
plt.contour(xx1, xx2, probs, [0.5], linewidths=1, colors='red');
```



Multinomial Logistic Regression Model

Another useful form of logistic regression is multinomial logistic regression in which the target or dependent variable can have 3 or more possible unordered types i.e. the types having no quantitative significance.

Implementation of Multinomial Logistic Regression Model in Python

Now we will implement the above concept of multinomial logistic regression in Python. For this purpose, we are using a dataset from sklearn named digit.

First, we need to import the necessary libraries as follows –

```
Import sklearn
from sklearn import datasets
from sklearn import linear_model
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

Next, we need to load digit dataset –

```
digits = datasets.load_digits()
```

Now, define the feature matrix(X) and response vector(y) as follows –

```
X = digits.data  
y = digits.target
```

With the help of next line of code, we can split X and y into training and testing sets –

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
```

Now create an object of logistic regression as follows –

```
digreg = linear_model.LogisticRegression()
```

Now, we need to train the model by using the training sets as follows –

```
digreg.fit(X_train, y_train)
```

Next, make the predictions on testing set as follows –

```
y_pred = digreg.predict(X_test)
```

Next print the accuracy of the model as follows –

```
print("Accuracy of Logistic Regression model is:",  
      metrics.accuracy_score(y_test, y_pred)*100)
```

Output

Accuracy of Logistic Regression model is: 95.6884561891516

From the above output we can see the accuracy of our model is around 96 percent.

4.7] Time series analysis:

A time series is a sequence of data points collected, recorded, or measured at successive, evenly-spaced time intervals.

Each data point represents observations or measurements taken over time, such as stock prices, temperature readings, or sales figures. Time series data is commonly represented graphically with time on the horizontal axis and the variable of interest on the vertical axis, allowing analysts to identify trends, patterns, and changes over time.

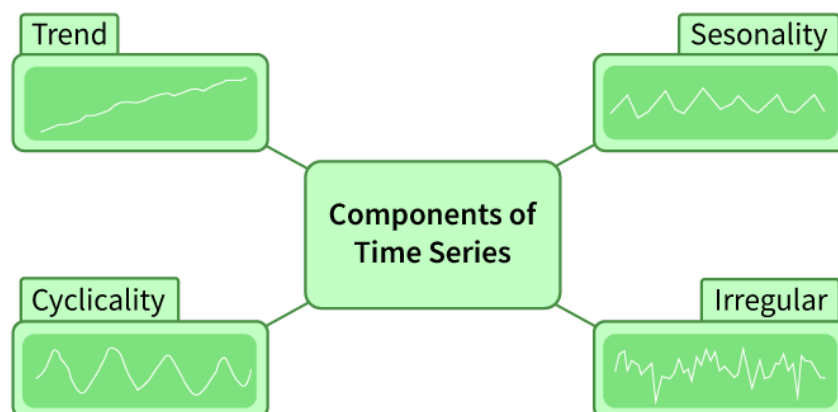
Time series data is often represented graphically as a line plot, with time depicted on the horizontal x-axis and the variable's values displayed on the vertical y-axis. This graphical representation facilitates the visualization of trends, patterns, and fluctuations in the variable over time, aiding in the analysis and interpretation of the data.

Importance of Time Series Analysis

1. **Predict Future Trends:** Time series analysis enables the prediction of future trends, allowing businesses to anticipate market demand, stock prices, and other key variables, facilitating proactive decision-making.
2. **Detect Patterns and Anomalies:** By examining sequential data points, time series analysis helps detect recurring patterns and anomalies, providing insights into underlying behaviors and potential outliers.
3. **Risk Mitigation:** By spotting potential risks, businesses can develop strategies to mitigate them, enhancing overall risk management.
4. **Strategic Planning:** Time series insights inform long-term strategic planning, guiding decision-making across finance, healthcare, and other sectors.
5. **Competitive Edge:** Time series analysis enables businesses to optimize resource allocation effectively, whether it's inventory, workforce, or financial assets. By staying ahead of market trends, responding to changes, and making data-driven decisions, businesses gain a competitive edge.

Components of Time Series Data

There are four main [components of a time series](#):



Components of Time Series Data

1. **Trend:** Trend represents the long-term movement or directionality of the data over time. It captures the overall tendency of the series to increase, decrease, or remain stable. Trends can be linear, indicating a consistent increase or decrease, or nonlinear, showing more complex patterns.
2. **Seasonality:** Seasonality refers to periodic fluctuations or patterns that occur at regular intervals within the time series. These cycles often repeat annually, quarterly, monthly, or weekly and are typically influenced by factors such as seasons, holidays, or business cycles.

3. **Cyclic variations:** Cyclical variations are longer-term fluctuations in the time series that do not have a fixed period like seasonality. These fluctuations represent economic or business cycles, which can extend over multiple years and are often associated with expansions and contractions in economic activity.
4. **Irregularity (or Noise):** Irregularity, also known as noise or randomness, refers to the unpredictable or random fluctuations in the data that cannot be attributed to the trend, seasonality, or cyclical variations. These fluctuations may result from random events, measurement errors, or other unforeseen factors. Irregularity makes it challenging to identify and model the underlying patterns in the time series data.

Time Series Visualization

[Time series visualization](#) is the graphical representation of data collected over successive time intervals. It encompasses various techniques such as line plots, seasonal subseries plots, autocorrelation plots, histograms, and interactive visualizations. These methods help analysts identify trends, patterns, and anomalies in time-dependent data for better understanding and decision-making.

Different Time series visualization graphs

1. **Line Plots:** Line plots display data points over time, allowing easy observation of trends, cycles, and fluctuations.
2. **Seasonal Plots:** These plots break down time series data into seasonal components, helping to visualize patterns within specific time periods.
3. **Histograms and Density Plots:** Shows the distribution of data values over time, providing insights into data characteristics such as skewness and kurtosis.
4. **Autocorrelation and Partial Autocorrelation Plots:** These plots visualize correlation between a time series and its lagged values, helping to identify seasonality and lagged relationships.
5. **Spectral Analysis:** Spectral analysis techniques, such as periodograms and spectrograms, visualize frequency components within time series data, useful for identifying periodicity and cyclical patterns.
6. **Decomposition Plots:** Decomposition plots break down a time series into its trend, seasonal, and residual components, aiding in understanding the underlying patterns.

4.8] Support Vector Machine:

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithm which is used for both classification and regression. But generally, they are used in classification problems.

In 1960s, SVMs were first introduced but later they got refined in 1990 also. SVMs have their unique way of implementation as compared to other machine learning algorithms. Now a days, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

Working of SVM:

The goal of SVM is to find a hyperplane that separates the data points into different classes. A hyperplane is a line in 2D space, a plane in 3D space, or a higher-dimensional surface in n-dimensional space. The hyperplane is chosen in such a way that it maximizes the margin, which is the distance between the hyperplane and the closest data points of each class. The closest data points are called the support vectors.

The distance between the hyperplane and a data point "x" can be calculated using the formula –

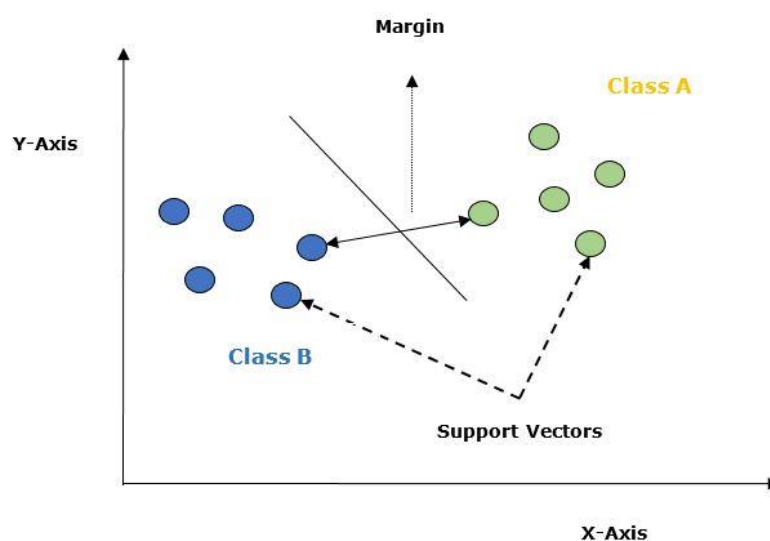
$$\text{distance} = (w \cdot x + b) / \|w\|$$

where "w" is the weight vector, "b" is the bias term, and " $\|w\|$ " is the Euclidean norm of the weight vector. The weight vector "w" is perpendicular to the hyperplane and determines its orientation, while the bias term "b" determines its position.

The optimal hyperplane is found by solving an optimization problem, which is to maximize the margin subject to the constraint that all data points are correctly classified. In other words, we want to find the hyperplane that maximizes the margin between the two classes while ensuring that no data point is misclassified. This is a convex optimization problem that can be solved using quadratic programming.

If the data points are not linearly separable, we can use a technique called kernel trick to map the data points into a higher-dimensional space where they become separable. The kernel function computes the inner product between the mapped data points without computing the mapping itself. This allows us to work with the data points in the higher dimensional space without incurring the computational cost of mapping them.

Let's understand it in detail with the help of following diagram –



Given below are the important concepts in SVM –

- **Support Vectors** – Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyperplane** – As we can see in the above diagram it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** – It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

Implementing SVM Using Python

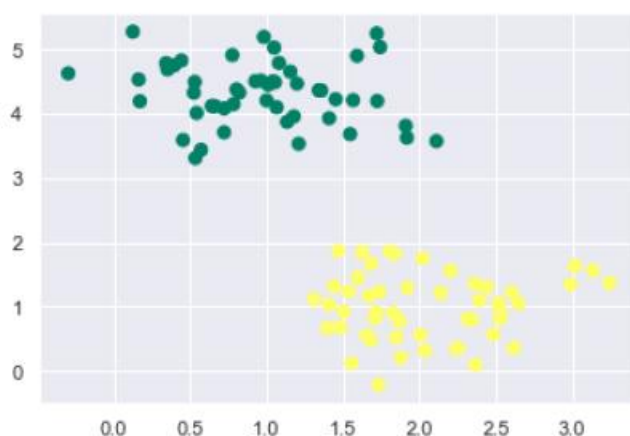
For implementing SVM in Python we will start with the standard libraries import as follows –

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns; sns.set()
```

Next, we are creating a sample dataset, having linearly separable data, from `sklearn.datasets.sample_generator` for classification using SVM –

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=100, centers=2, random_state=0, cluster_std=0.50)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer');
```

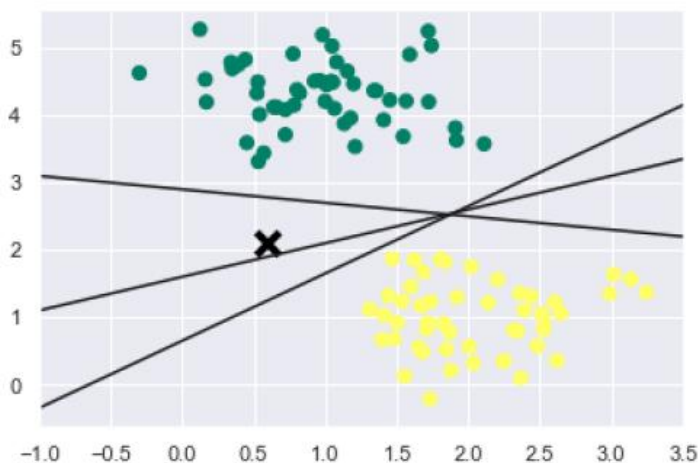
The following would be the output after generating sample dataset having 100 samples and 2 clusters –



We know that SVM supports discriminative classification. it divides the classes from each other by simply finding a line in case of two dimensions or manifold in case of multiple dimensions. It is implemented on the above dataset as follows –

```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer')
plt.plot([0.6], [2.1], 'x', color='black', markeredgewidth=4, markersize=12)
for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:
    plt.plot(xfit, m * xfit + b, '-k')
plt.xlim(-1, 3.5);
```

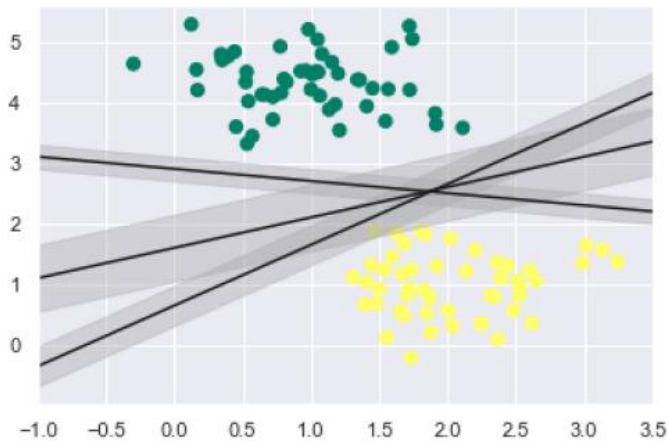
The output is as follows –



We can see from the above output that there are three different separators that perfectly discriminate the above samples.

As discussed, the main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) hence rather than drawing a zero line between classes we can draw around each line a margin of some width up to the nearest point. It can be done as follows –

```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer')
for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
    yfit = m * xfit + b
    plt.plot(xfit, yfit, '-k')
    plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
                    color='#AAAAAA', alpha=0.4)
plt.xlim(-1, 3.5);
```



From the above image in output, we can easily observe the "margins" within the discriminative classifiers. SVM will choose the line that maximizes the margin.

Next, we will use Scikit-Learn's support vector classifier to train an SVM model on this data. Here, we are using linear kernel to fit SVM as follows –

```
from sklearn.svm import SVC # "Support vector classifier"
model = SVC(kernel='linear', C=1E10)
model.fit(X, y)
```

The output is as follows –

```
SVC(C=10000000000.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
kernel='linear', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001, verbose=False)
```

Now, for a better understanding, the following will plot the decision functions for 2D SVC –

```
def decision_function(model, ax=None, plot_support=True):
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
```

For evaluating model, we need to create grid as follows –

```
x = np.linspace(xlim[0], xlim[1], 30)
y = np.linspace(ylim[0], ylim[1], 30)
Y, X = np.meshgrid(y, x)
xy = np.vstack([X.ravel(), Y.ravel()]).T
```

```
P = model.decision_function(xy).reshape(X.shape)
```

Next, we need to plot decision boundaries and margins as follows –

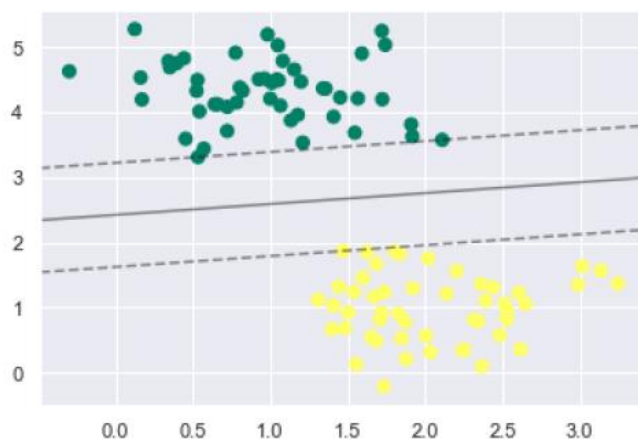
```
ax.contour(X, Y, P, colors='k',  
           levels=[-1, 0, 1], alpha=0.5,  
           linestyles=['--', '-', '--'])
```

Now, similarly plot the support vectors as follows –

```
if plot_support:  
    ax.scatter(model.support_vectors_[0],  
              model.support_vectors_[1],  
              s=300, linewidth=1, facecolors='none');  
ax.set_xlim(xlim)  
ax.set_ylim(ylim)
```

Now, use this function to fit our models as follows –

```
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer')  
decision_function(model);
```



We can observe from the above output that an SVM classifier fit to the data with margins i.e. dashed lines and support vectors, the pivotal elements of this fit, touching the dashed line. These support vector points are stored in the `support_vectors_` attribute of the classifier as follows –

```
model.support_vectors_
```

The output is as follows –

```
array([[0.5323772 , 3.31338909],
```

[2.11114739, 3.57660449],
[1.46870582, 1.86947425]])

SVM Kernels

In practice, SVM algorithm is implemented with kernel that transforms an input data space into the required form. SVM uses a technique called the kernel trick in which kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, kernel converts non-separable problems into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible and accurate. The following are some of the types of kernels used by SVM –

Linear Kernel

It can be used as a dot product between any two observations. The formula of linear kernel is as below –

$$k(x, x_i) = \text{sum}(x * x_i)$$

From the above formula, we can see that the product between two vectors say x & x_i is the sum of the multiplication of each pair of input values.

Polynomial Kernel

It is more generalized form of linear kernel and distinguish curved or nonlinear input space. Following is the formula for polynomial kernel –

$$K(x, x_i) = 1 + \text{sum}(x * x_i)^d$$

Here d is the degree of polynomial, which we need to specify manually in the learning algorithm.

Radial Basis Function (RBF) Kernel

RBF kernel, mostly used in SVM classification, maps input space in indefinite dimensional space. Following formula explains it mathematically –

$$K(x, x_i) = \exp(-\gamma * \text{sum}((x - x_i)^2))$$

Here, γ ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of γ is 0.1.

As we implemented SVM for linearly separable data, we can implement it in Python for the data that is not linearly separable. It can be done by using kernels.

Example

The following is an example for creating an SVM classifier by using kernels. We will be using iris dataset from scikit-learn –

We will start by importing following packages –

```
import pandas as pd
import numpy as np
from sklearn import svm, datasets
import matplotlib.pyplot as plt
```

Now, we need to load the input data –

```
iris = datasets.load_iris()
```

From this dataset, we are taking first two features as follows –

```
X = iris.data[:, :2]
y = iris.target
```

Next, we will plot the SVM boundaries with original data as follows –

```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
X_plot = np.c_[xx.ravel(), yy.ravel()]
```

Now, we need to provide the value of regularization parameter as follows –

```
C = 1.0
```

Next, SVM classifier object can be created as follows –

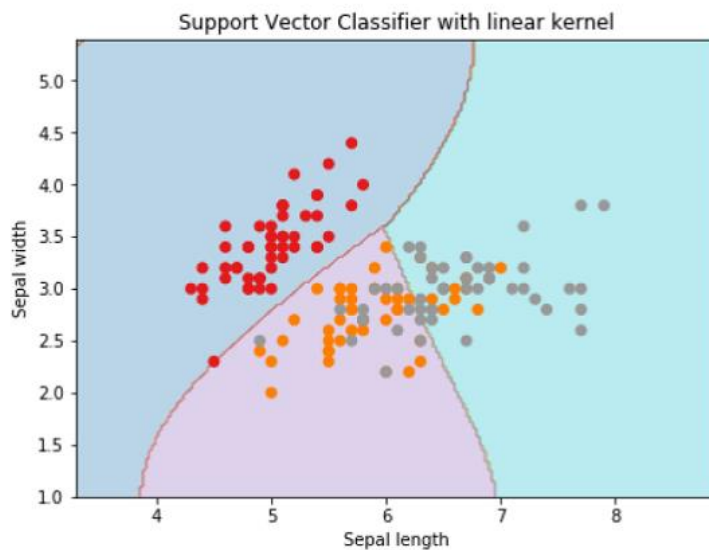
```
Svc_classifier = svm.SVC(kernel='linear', C=C).fit(X, y)
```

```
Z = svc_classifier.predict(X_plot)
Z = Z.reshape(xx.shape)
plt.figure(figsize=(15, 5))
plt.subplot(121)
plt.contourf(xx, yy, Z, cmap=plt.cm.tab10, alpha=0.3)
```

```
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('Support Vector Classifier with linear kernel')
```

Output

Text(0.5, 1.0, 'Support Vector Classifier with linear kernel')

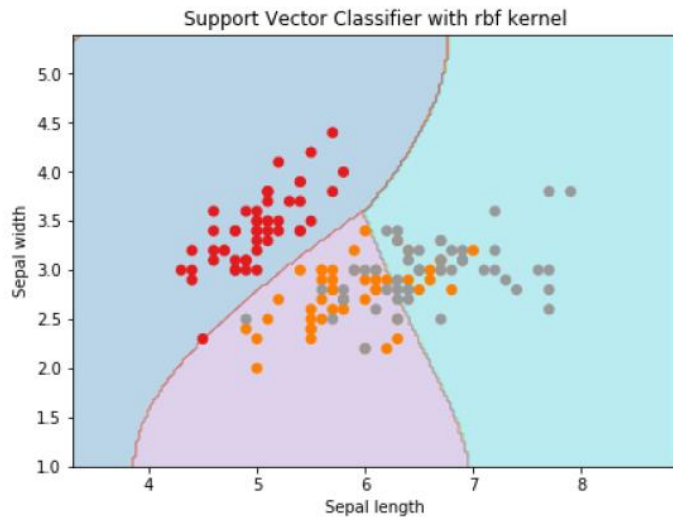


For creating SVM classifier with **rbf** kernel, we can change the kernel to **rbf** as follows –

```
Svc_classifier = svm.SVC(kernel='rbf', gamma='auto', C=C).fit(X, y)
Z = svc_classifier.predict(X_plot)
Z = Z.reshape(xx.shape)
plt.figure(figsize=(15, 5))
plt.subplot(121)
plt.contourf(xx, yy, Z, cmap=plt.cm.tab10, alpha=0.3)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('Support Vector Classifier with rbf kernel')
```

Output

Text(0.5, 1.0, 'Support Vector Classifier with rbf kernel')



We put the value of gamma to 'auto' but you can provide its value between 0 to 1 also.

Tuning SVM Parameters

In practice, SVMs often require tuning of their parameters to achieve optimal performance. The most important parameters to tune are the kernel, the regularization parameter C , and the kernel-specific parameters.

The kernel parameter determines the type of kernel to use. The most common kernel types are linear, polynomial, radial basis function (RBF), and sigmoid. The linear kernel is used for linearly separable data, while the other kernels are used for non-linearly separable data.

The regularization parameter C controls the trade-off between maximizing the margin and minimizing the classification error. A higher value of C means that the classifier will try to minimize the classification error at the expense of a smaller margin, while a lower value of C means that the classifier will try to maximize the margin even if it means more misclassifications.

The kernel-specific parameters depend on the type of kernel being used. For example, the polynomial kernel has parameters for the degree of the polynomial and the coefficient of the polynomial, while the RBF kernel has a parameter for the width of the Gaussian function.

Example

```
from sklearn.model_selection import GridSearchCV
# define the parameter grid
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'degree': [2, 3, 4],
    'coef0': [0.0, 0.1, 0.5],
```

```
'gamma': ['scale', 'auto']
}

# create an SVM classifier
svm = SVC()

# perform grid search to find the best set of parameters
grid_search = GridSearchCV(svm, param_grid, cv=5)
grid_search.fit(X_train, y_train)
# print the best set of parameters and their accuracy
print("Best parameters:", grid_search.best_params_)
print("Best accuracy:", grid_search.best_score_)
```

We start by importing the **GridSearchCV** module from scikit-learn, which is a tool for performing grid search on a set of parameters. We define a parameter grid that contains the possible values for each parameter we want to tune.

We create an SVM classifier using **SVC()** and then pass it to **GridSearchCV** along with the parameter grid and the number of cross-validation folds (**cv=5**). We then call **grid_search.fit(X_train, y_train)** to perform the grid search.

Once the grid search is complete, we print the best set of parameters and their accuracy using **grid_search.best_params_** and **grid_search.best_score_**, respectively.

Output

On executing this program, you will get the following output –

```
Best parameters: {'C': 0.1, 'coef0': 0.5, 'degree': 3, 'gamma': 'scale', 'kernel': 'poly'}
Best accuracy: 0.975
```

What is Naive Bayes Theorem?

Naive Bayes Theorem, a probabilistic algorithm used in machine learning predicts the likelihood of an outcome based on prior knowledge of conditions related to that outcome. This theorem assumes independence among predictors and calculates the probability of a certain event occurring. It is widely used for classification tasks, such as spam filtering and sentiment analysis, due to its simplicity, efficiency, and effectiveness, especially in cases with large datasets and multiple variables.

Imagine you have a bag of jellybeans. There are different colors like red, green, and blue. Now, let's say you know that most of the blue jellybeans are sour, and only a few are sweet.

Naive Bayes Theorem is like a smart guess about whether a jellybean from the bag will be sweet or sour, based on its color. So, if you pick a blue jellybean, Naive Bayes helps you guess it's probably going to be sour because most blue jellybeans in your bag are sour.

It's like if you see clouds in the sky, you might guess it's going to rain, because often when there are clouds, it rains. Naive Bayes does something similar but with math and probability. It looks at what usually happens and uses that information to make a good guess.

So, in our jellybean example, Naive Bayes uses the information about how many jellybeans of each colour are sour or sweet to help guess the taste of a new jellybean you pick, before you even taste it!

Think about your email inbox; it is flooded with messages, some of which are spam. Naïve Bayes helps your email provider automatically identify and filter out those annoying spam messages. It is also used in sentiment analysis, which involves determining the emotional tone of a piece of text, like a social media post or product review. For example, companies can use naive Bayes to measure customer opinions about their products or services based on what people say online.

Probability, Bayes Theorem, and Conditional Probability

Before we discuss the naive Bayes algorithm, it's essential to grasp the underlying concepts of probability, Bayes' theorem, and conditional probability.

- **Probability:** Probability is a measure of the likelihood of an event occurring. It is expressed as a value between 0 and 1, where 0 indicates that an event is impossible and 1 means the event is certain to happen.
- **Bayes' Theorem:** Named after the Reverend Thomas Bayes, this theorem is a fundamental principle in probability theory. It describes how to update the probability of a hypothesis as more evidence or information becomes available.

Bayes' theorem is expressed as follows:

$$P(A|B) = P(B|A) \times P(A) / P(B)$$

- $P(A|B)$ represents the probability of event A occurring given that event B has occurred.
 - $P(B|A)$ is the probability of event B occurring given that event A has occurred.
 - $P(A)$ is the prior probability of event A.
 - $P(B)$ is the prior probability of event B.
- **Conditional Probability:** Conditional probability is the probability of an event occurring, given that another event has already occurred.

It is calculated as follows:

$$P(A|B) = P(A \cap B) / P(B)$$

- $P(A|B)$ is the probability of event A occurring given that event B has occurred.
- $P(B)$ is the prior probability of event B.
- $P(A \cap B)$ is the probability of events A and B occurring simultaneously.

This formula essentially determines the probability of event A happening under the condition that event B has already taken place. It quantifies the relationship between two events when one is known to have occurred, influencing the likelihood of the other event occurring.

Naive Bayes Theorem in Machine Learning:

Naive Bayes is a fundamental and easy-to-understand [machine learning](#) algorithm used for classification and probabilistic reasoning. It's widely applied in spam email filtering, sentiment analysis, and medical diagnosis. Naive Bayes leverages Bayes' theorem, developed by Thomas Bayes in the 18th century. This approach is "naive" because it assumes that the features (attributes) used in the model are independent, even though they might not be in reality. This simplification makes the algorithm computationally efficient and easy to implement.

The naive Bayes algorithm starts with a basic premise that we can estimate the probability of an event (such as an email being spam or not) based on prior knowledge and available evidence. This evidence consists of various features, like the presence of certain words in an email's text.

Let's break down the key components of naive Bayes:

- **Prior Probability:** Before observing any features, we start with a prior probability, which is our initial belief in the likelihood of an event occurring. For example, the prior probability of an email being spam might be based on the overall spam rate in your inbox.
- **Likelihood:** This is where we use the features. We calculate the likelihood of observing these features in emails of different classes (spam or not). In the context of spam detection, we look at the probability of specific words or patterns appearing in spam or non-spam emails.
- **Evidence:** The features in our dataset provide evidence. For instance, if an email contains words like "free," "win," and "money," it's more likely to be spam.
- **Posterior Probability:** Using Bayes' theorem, we combine the prior probability and the likelihood to compute the posterior probability. This is our updated belief about the probability of an email being spam, given the observed features.

How Does the Naive Bayes Theorem Work?

Naive Bayes is popular because it is easy to implement and often performs well, even with the "naive" assumption of feature independence. The naive Bayes theorem figures out whether something is this or that based on evidence.

Imagine you have a box of fruits and you want to know if a fruit is an apple or an orange. Naive Bayes helps you with this decision.

Let's discuss how it works:



- **Collecting Evidence:** First, you collect some clues (features) about the fruit, like its colour, shape, and size. These clues are like hints to solve the fruit mystery.
- **Prior Beliefs:** You start with what you already know. You might know that in your fruit box, 70% are apples and 30% are oranges. These are your initial beliefs or “prior probabilities.”
- **Checking Clues:** You look at the clues you collected about the fruit. For example, if the fruit is red, small, and round, these clues suggest it’s more likely an apple.
- **Probability Update:** Naive Bayes combines your initial beliefs with the clues. It calculates how likely it is that the fruit is an apple or an orange based on the clues. If the clues are more apple-like, it increases the chance of it being an apple.
- **Decision Time:** Finally, you look at the updated probabilities. If the chance of it being an apple is higher, you conclude that the fruit is most likely an apple.

Let us see how naive Bayes theorem works in a technical context, spam filtering:

Imagine two people, Alice and Bob, who are known for sending you emails. You know that Alice sends about 80% of the emails, and Bob sends the remaining 20%. You also know that Alice uses the word “vacation” 70% of the time, while Bob uses it only 10% of the time. If you receive an email with the word “vacation,” you can use naive Bayes to estimate the probability of it being from Alice or Bob.

In this case,

- The prior probability for Alice is 80% (0.8), and for Bob, it’s 20% (0.2).

- The likelihood of observing “vacation” given Alice is 70% (0.7), and given Bob, it’s 10% (0.1).

The naive Bayes theorem helps in determining the probability of occurrences for individuals like Alice and Bob. This methodology involves evaluating the prior probability and likelihood for each person and subsequently standardizing the outcomes to ensure consistency.

The algorithm applies these principles to high-dimensional datasets with multiple features, but the fundamental idea remains the same: updating probabilities based on observed evidence.

Types of the Naive Bayes Model:

The naive Bayes algorithm has several types, made to handle diverse types of data effectively. These versions are designed to accommodate specific kinds of information, ensuring optimal performance in various scenarios and tasks.

- **Multinomial Naive Bayes:** Multinomial Naive Bayes is like a specialist in dealing with text data, making it perfect for language-related tasks like document classification, spam email filtering, or sentiment analysis. It thrives when the data is discrete, which means it’s counted in terms of occurrences. For example, when analyzing text, you often represent it as word frequencies or counts. Multinomial Naive Bayes takes these word counts and computes the probabilities of different words or features appearing in documents. It assumes that the data is generated from a multinomial distribution, hence the name.
- **Gaussian Naive Bayes:** Gaussian Naive Bayes is the go-to expert for handling continuous data. This variant is a good fit for scenarios where you’re dealing with measurements or attributes that follow a normal distribution (bell-shaped curve). Think about situations where you have data like height, weight, temperature, or any value that can take on a wide range of numeric values. Gaussian Naive Bayes assumes that the data follows a Gaussian (normal) distribution, which simplifies the probability calculations. It’s commonly used in fields such as medical diagnosis, image classification, and fraud detection, where continuous data is prevalent.

- **Bernoulli Naive Bayes:** Bernoulli Naive Bayes specializes in binary data, where the features are either present (1) or absent (0). It's particularly handy in document classification tasks like spam detection, where you're often interested in whether certain words are present in a document or not. Instead of counting occurrences, it focuses on whether a word is "on" or "off" in a document, simplifying the data representation. In this case, it assumes that the features follow a Bernoulli distribution, which models binary outcomes.

Advantages of Naive Bayes Theorem

- **Simplicity:** Naive Bayes is easy to grasp, making it an ideal choice for beginners in machine learning. The core concept revolves around updating probabilities based on observed evidence.
- **Efficiency:** Naive Bayes processes information swiftly and efficiently, making it a top pick when dealing with large datasets.
- **Applicability:** Naive Bayes has a broad range of use cases. It helps with spotting spam, diagnosing illnesses in healthcare, measuring public sentiment through social media analysis, etc.
- **Robust to Irrelevant Features:** This means that if you're working with a dataset containing lots of information, not all of it essential, naive Bayes can still provide reliable predictions. It pays attention to what's relevant.

Applications of Naive Bayes Algorithm:

- **Spam Email Detection:** Naive Bayes scrutinizes incoming emails, focusing on the words, phrases, and patterns within the email content. By comparing this content with a database of known spam and non-spam emails, naive Bayes makes informed decisions on whether an email is likely spam or not.
- **Sentiment Analysis:** Sentiment analysis employs naive Bayes to gauge the emotions conveyed in text data, such as product reviews or social media posts. The algorithm looks for clues like positive words ("amazing," "love"), negative words ("terrible," "hate"), and neutral expressions. This way, naive Bayes helps determine whether the sentiment behind

the text is positive, negative, or neutral. It's valuable for businesses wanting to understand customer reactions or public sentiment about their products or services.

- **Medical Diagnosis:** Within healthcare, the naive Bayes algorithm helps in disease prediction through symptom and test result analysis. By comparing patient data with extensive medical records, it assists physicians in treatment decisions. This approach significantly contributes to early ailment detection and ensures appropriate medical interventions.
- **Text Classification:** Naive Bayes aids in text classification tasks, such as sorting news articles into topics, categorizing legal documents, and classifying social media posts based on themes. It analyzes the words and patterns in the text to assign relevant labels or categories. This capability is invaluable for organizing and retrieving information efficiently.
- **Recommendation Systems:** Naive Bayes is your trusted advisor in recommendation systems. It uses your preferences and past behaviour to suggest products, movies, or content you might enjoy. For example, in e-commerce, it can recommend items based on your previous purchases or browsing history. By learning your habits and interests, naive Bayes assists in personalizing your online experience, guiding you to discover new items you'll likely find appealing.

Unit V: Classification & Unsupervised Learning

5.1] Classification:

Machine Learning for classification

Classification is a process of categorizing data or objects into predefined classes or categories based on their features or attributes.

[Machine Learning](#) classification is a type of [supervised learning](#) technique where an algorithm is trained on a labeled dataset to predict the class or category of new, unseen data.

The main objective of classification machine learning is to build a model that can accurately assign a label or category to a new observation based on its features.

For example, a classification model might be trained on a dataset of images labeled as either dogs or cats and then used to predict the class of new, unseen images of dogs or cats based on their features such as color, texture, and shape.

Classification Types

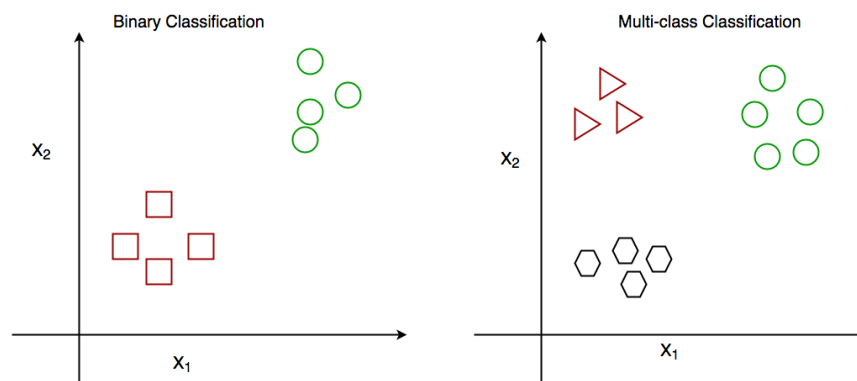
There are two main classification types in machine learning:

Binary Classification

In binary classification, the goal is to classify the input into one of two classes or categories. Example – On the basis of the given health conditions of a person, we have to determine whether the person has a certain disease or not.

Multiclass Classification

In multi-class classification, the goal is to classify the input into one of several classes or categories. For Example – On the basis of data about different species of flowers, we have to determine which specie our observation belongs to.



Binary vs Multi class classification

Other categories of classification involves:

Multi-Label Classification

In, [Multi-label Classification](#) the goal is to predict which of several labels a new data point belongs to. This is different from multiclass classification, where each data point can only belong to one class. For example, a multi-label classification algorithm could be used to classify images of animals as belonging to one or more of the categories cat, dog, bird, or fish.

Imbalanced Classification

In, [Imbalanced Classification](#) the goal is to predict whether a new data point belongs to a minority class, even though there are many more examples of the majority class. For example, a medical diagnosis algorithm could be used to predict whether a patient has a rare disease, even though there are many more patients with common diseases.

Classification Algorithms

There are various types of **classifiers algorithms**. Some of them are :

Linear Classifiers

Linear models create a linear decision boundary between classes. They are simple and computationally efficient. Some of the linear **classification** models are as follows:

- [Logistic Regression](#)
- [Support Vector Machines having kernel = 'linear'](#)
- [Single-layer Perceptron](#)
- Stochastic Gradient Descent (SGD) Classifier

Non-linear Classifiers

Non-linear models create a non-linear decision boundary between classes. They can capture more complex relationships between the input features and the target variable. Some of the non-linear **classification** models are as follows:

- [K-Nearest Neighbours](#)
- [Kernel SVM](#)
- [Naive Bayes](#)
- [Decision Tree Classification](#)
- [Ensemble learning classifiers:](#)
- [Random Forests,](#)
- [AdaBoost,](#)
- [Bagging Classifier,](#)
- [Voting Classifier,](#)
- [ExtraTrees Classifier](#)
- [Multi-layer Artificial Neural Networks](#)

Learners in Classifications Algorithm

In machine learning, classification learners can also be classified as either “lazy” or “eager” learners.

- **Lazy Learners:** Lazy Learners are also known as instance-based learners, lazy learners do not learn a model during the training phase. Instead, they simply store the training data and use it to classify new instances at prediction time. It is very fast at prediction time because it does not require computations during the predictions. It is less effective in high-dimensional spaces or when the number of training instances is large. Examples of lazy learners include k-nearest neighbours and case-based reasoning.
- **Eager Learners:** Eager Learners are also known as model-based learners; eager learners learn a model from the training data during the training phase and use this model to classify new instances at prediction time. It is more effective in high-dimensional spaces having large training datasets. Examples of eager learners include decision trees, random forests, and support vector machines.

5.2] Classification Models in Machine Learning

Evaluating a classification model is an important step in machine learning, as it helps to assess the performance and generalization ability of the model on new, unseen data. There are several metrics and techniques that can be used to evaluate a classification model, depending on the specific problem and requirements. Here are some commonly used evaluation metrics:

- **Classification Accuracy:** The proportion of correctly classified instances over the total number of instances in the test set. It is a simple and intuitive metric but can be misleading in imbalanced datasets where the majority class dominates the accuracy score.
- **Confusion matrix:** A table that shows the number of true positives, true negatives, false positives, and false negatives for each class, which can be used to calculate various evaluation metrics.
- **Precision and Recall:** Precision measures the proportion of true positives over the total number of predicted positives, while recall measures the proportion of true positives over the total number of actual positives. These metrics are useful in scenarios where one class is more important than the other, or when there is a trade-off between false positives and false negatives.
- **F1-Score:** The harmonic mean of precision and recall, calculated as $2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$. It is a useful metric for imbalanced datasets where both precision and recall are important.
- **ROC curve and AUC:** The Receiver Operating Characteristic (ROC) curve is a plot of the true positive rate (recall) against the false positive rate (1-specificity) for different threshold values of the classifier's decision function. The Area Under the Curve (AUC) measures the overall performance of the classifier, with values ranging from 0.5 (random guessing) to 1 (perfect classification).
- **Cross-validation:** A technique that divides the data into multiple folds and trains the model on each fold while testing on the others, to obtain a more robust estimate of the model's performance.

It is important to choose the appropriate evaluation metric(s) based on the specific problem and requirements, and to avoid overfitting by evaluating the model on independent test data.

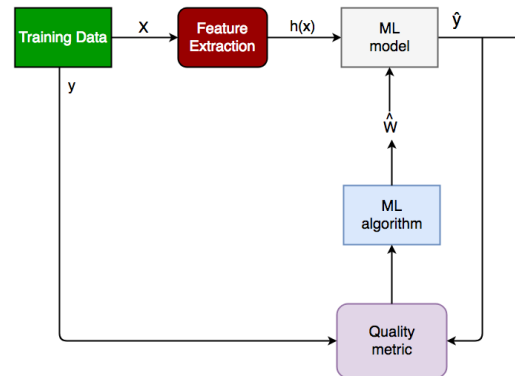
Characteristics of Classification

Here are the characteristics of the classification:

- **Categorical Target Variable:** Classification deals with predicting categorical target variables that represent discrete classes or labels. Examples include classifying emails as spam or not spam, predicting whether a patient has a high risk of heart disease, or identifying image objects.
- **Accuracy and Error Rates:** Classification models are evaluated based on their ability to correctly classify data points. Common metrics include accuracy, precision, recall, and F1-score.
- **Model Complexity:** Classification models range from simple linear classifiers to more complex nonlinear models. The choice of model complexity depends on the complexity of the relationship between the input features and the target variable.
- **Overfitting and Underfitting:** Classification models are susceptible to overfitting and underfitting. Overfitting occurs when the model learns the training data too well and fails to generalize to new data.

How does Classification Machine Learning Work?

The basic idea behind classification is to train a model on a labeled dataset, where the input data is associated with their corresponding output labels, to learn the patterns and relationships between the input data and output labels. Once the model is trained, it can be used to predict the output labels for new unseen data.



Classification Machine Learning

The classification process typically involves the following steps:

Understanding the problem

Before getting started with classification, it is important to understand the problem you are trying to solve. What are the class labels you are trying to predict? What is the relationship between the input data and the class labels?

Suppose we have to predict whether a patient has a certain disease or not, on the basis of 7 independent variables, called features. This means, there can be only two possible outcomes:

- The patient has the disease, which means “**True**”.
- The patient has no disease. which means “**False**”.

This is a binary classification problem.

Data preparation

Once you have a good understanding of the problem, the next step is to prepare your data. This includes collecting and pre-processing the data and splitting it into training, validation, and test sets. In this step, the data is cleaned, pre-processed, and transformed into a format that can be used by the classification algorithm.

- **X**: It is the independent feature, in the form of an $N \times M$ matrix. N is the no. of observations and M is the number of features.
- **y**: An N vector corresponding to predicted classes for each of the N observations.

Feature Extraction

The relevant features or attributes are extracted from the data that can be used to differentiate between the different classes.

Suppose our input X has 7 independent features, having only 5 features influencing the label or target values remaining 2 are negligibly or not correlated, then we will use only these 5 features only for the model training.

Model Selection

There are many different models that can be used for classification, including **logistic regression, decision trees, support vector machines (SVM), or neural networks**. It is important to select a model that is appropriate for your problem, taking into account the size and complexity of your data, and the computational resources you have available.

Model Training

Once you have selected a model, the next step is to train it on your training data. This involves adjusting the parameters of the model to minimize the error between the predicted class labels and the actual class labels for the training data.

Model Evaluation

Evaluating the model: After training the model, it is important to evaluate its performance on a validation set. This will give you a good idea of how well the model is likely to perform on new, unseen data.

Log Loss or Cross-Entropy Loss, Confusion Matrix, Precision, Recall, and AUC-ROC curve are the quality metrics used for measuring the performance of the model.

Fine-tuning the model

If the model's performance is not satisfactory, you can fine-tune it by adjusting the parameters, or trying a different model.

Deploying the model

Finally, once we are satisfied with the performance of the model, we can deploy it to make predictions on new data. it can be used for real world problem.

Examples of Machine Learning Classification in Real Life

Classification algorithms are widely used in many real-world applications across various domains, including:

- Email spam filtering
- Credit risk assessment
- Medical diagnosis
- Image classification
- Sentiment analysis.
- Fraud detection
- Quality control
- Recommendation systems

Implementation of Classification Model in Machine Learning

Let's get a hands-on experience with how Classification works.

We are going to study various Classifiers and see a rather simple analytical comparison of their performance on a well-known, standard data set, the Iris data set.

Requirements for running the given script:

- [Python](#)
- [Scipy and Numpy](#)
- [Pandas](#)
- [Scikit-learn](#)

```
# Importing the required libraries

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

from sklearn import datasets

from sklearn import svm

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive_bayes import GaussianNB


# import the iris dataset

iris = datasets.load_iris()

X = iris.data

y = iris.target


# splitting X and y into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.3, random_state=1)
```

```
# GAUSSIAN NAIVE BAYES
```

```
gnb = GaussianNB()
```

```
# train the model
```

```
gnb.fit(X_train, y_train)
```

```
# make predictions
```

```
gnb_pred = gnb.predict(X_test)
```

```
# print the accuracy
```

```
print("Accuracy of Gaussian Naive Bayes: ",
```

```
      accuracy_score(y_test, gnb_pred))
```

```
# print other performance metrics
```

```
print("Precision of Gaussian Naive Bayes: ",
```

```
      precision_score(y_test, gnb_pred, average='weighted'))
```

```
print("Recall of Gaussian Naive Bayes: ",
```

```
      recall_score(y_test, gnb_pred, average='weighted'))
```

```
print("F1-Score of Gaussian Naive Bayes: ",
```

```
      f1_score(y_test, gnb_pred, average='weighted'))
```

```
# DECISION TREE CLASSIFIER
```

```
dt = DecisionTreeClassifier(random_state=0)
```

```
# train the model
```

```
dt.fit(X_train, y_train)

# make predictions

dt_pred = dt.predict(X_test)

# print the accuracy

print("Accuracy of Decision Tree Classifier: ",

      accuracy_score(y_test, dt_pred))

# print other performance metrics

print("Precision of Decision Tree Classifier: ",

      precision_score(y_test, dt_pred, average='weighted'))

print("Recall of Decision Tree Classifier: ",

      recall_score(y_test, dt_pred, average='weighted'))

print("F1-Score of Decision Tree Classifier: ",

      f1_score(y_test, dt_pred, average='weighted'))


# SUPPORT VECTOR MACHINE

svm_clf = svm.SVC(kernel='linear') # Linear Kernel

# train the model

svm_clf.fit(X_train, y_train)

# make predictions

svm_clf_pred = svm_clf.predict(X_test)
```

```

# print the accuracy

print("Accuracy of Support Vector Machine: ",

      accuracy_score(y_test, svm_clf_pred))

# print other performance metrics

print("Precision of Support Vector Machine: ",

      precision_score(y_test, svm_clf_pred, average='weighted'))

print("Recall of Support Vector Machine: ",

      recall_score(y_test, svm_clf_pred, average='weighted'))

print("F1-Score of Support Vector Machine: ",

      f1_score(y_test, svm_clf_pred, average='weighted'))

```

Output:

```

Accuracy of Gaussian Naive Bayes: 0.9333333333333333
Precision of Gaussian Naive Bayes: 0.9352007469654529
Recall of Gaussian Naive Bayes: 0.9333333333333333
F1-Score of Gaussian Naive Bayes: 0.933615520282187
Accuracy of Decision Tree Classifier: 0.9555555555555556
Precision of Decision Tree Classifier: 0.9555555555555556
Recall of Decision Tree Classifier: 0.9555555555555556
F1-Score of Decision Tree Classifier: 0.9555555555555556
Accuracy of Support Vector Machine: 1.0
Precision of Support Vector Machine: 1.0
Recall of Support Vector Machine: 1.0
F1-Score of Support Vector Machine: 1.0

```

Conclusion

In conclusion, classification is a fundamental task in machine learning, involving the categorization of data into predefined classes or categories based on their features.

5.3] Classification Trees:

Classification tree methods (i.e., decision tree methods) are recommended when the data mining task contains classifications or predictions of outcomes, and the goal is to generate rules that can be easily explained and translated into SQL or a natural query language.

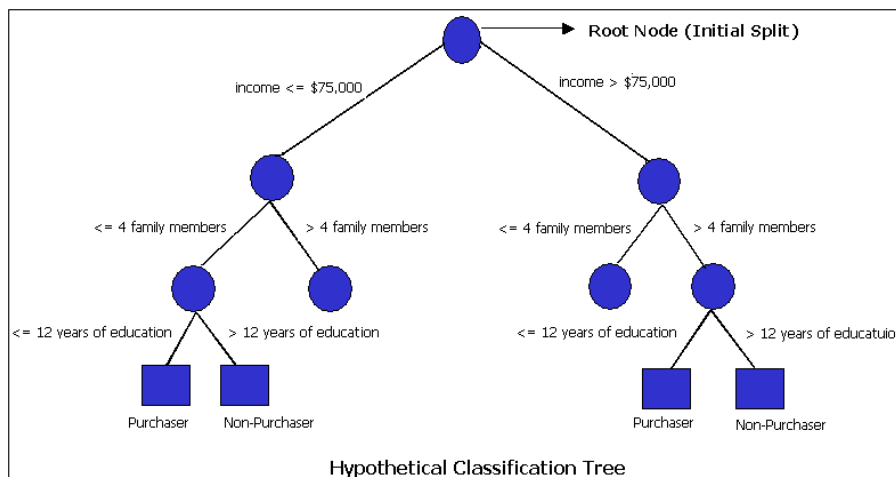
A Classification tree labels, records, and assigns variables to discrete classes. A Classification tree can also provide a measure of confidence that the classification is correct.

A Classification tree is built through a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches.

Initially, a Training Set is created where the classification label (i.e., purchaser or non-purchaser) is known (pre-classified) for each record. Next, the algorithm systematically assigns each record to one of two subsets on the same basis (i.e., income > \$75,000 or income ≤ \$75,000).

The object is to attain an homogeneous set of labels (i.e., purchaser or non-purchaser) in each partition. This partitioning (splitting) is then applied to each of the new partitions.

The process continues until no more useful splits can be found. The heart of the algorithm is the rule that determines the initial split rule (displayed in the following figure).



The process starts with a Training Set consisting of pre-classified records (target field or dependent variable with a known class or label such as purchaser or non-purchaser).

The goal is to build a tree that distinguishes among the classes. For simplicity, assume that there are only two target classes, and that each split is a binary partition.

The partition (splitting) criterion generalizes to multiple classes, and any multi-way partitioning can be achieved through repeated binary splits. To choose the best splitter at a node, the algorithm considers each input field in turn. In essence, each field is sorted.

Every possible split is tried and considered, and the best split is the one that produces the largest decrease in diversity of the classification label within each partition (i.e., the increase in homogeneity).

This is repeated for all fields, and the winner is chosen as the best splitter for that node. The process is continued at subsequent nodes until a full tree is generated.

5.4] Bayesian Method:

The **Bayesian Method** refers to a statistical approach based on **Bayes' Theorem**, which provides a framework for updating probabilities as new evidence or information becomes available. It is widely used in various fields such as machine learning, medicine, finance, and engineering to make decisions under uncertainty.

Key Concepts in the Bayesian Method

1. Bayes' Theorem

Bayes' Theorem is the mathematical foundation of the Bayesian method:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

- $P(A|B)$: Posterior probability (probability of AA given BB)
 - $P(B|A)$: Likelihood (probability of BB given AA)
 - $P(A)$: Prior probability (initial belief about AA)
 - $P(B)$: Evidence or marginal likelihood (overall probability of BB)
- ### 2. Prior Probability ($P(A)$)
- Represents the initial belief about an event before observing new data.
- ### 3. Likelihood ($P(B|A)$)
- Represents how likely the observed data BB is, given the event AA.
- ### 4. Posterior Probability ($P(A|B)$)
- Updated belief about the event AA after observing data BB.
- ### 5. Evidence ($P(B)$)
- A normalizing constant that ensures probabilities sum to 1.
-

Steps in the Bayesian Approach

1. **Define the Prior:**
Start with a prior probability distribution based on existing knowledge or assumptions.
 2. **Collect Data:**
Gather new evidence or data.
 3. **Compute the Likelihood:**
Assess how well the new data fits with different hypotheses.
 4. **Update Beliefs:**
Use Bayes' Theorem to update the prior with the likelihood to obtain the posterior distribution.
 5. **Make Decisions:**
Use the posterior distribution to make inferences or predictions.
-

Applications of the Bayesian Method

1. **Machine Learning:**
 - Bayesian Networks
 - Naive Bayes Classifier
 - Bayesian Optimization
2. **Medical Diagnosis:**
 - Estimating the probability of a disease given symptoms and test results.
3. **Finance:**
 - Portfolio optimization and risk assessment.

4. **Engineering:**
 - Reliability analysis and predictive maintenance.
 5. **Natural Sciences:**
 - Updating models based on experimental data.
-

Advantages of the Bayesian Method

- **Incorporates Prior Knowledge:** Combines prior beliefs with observed data.
- **Handles Uncertainty:** Provides a probabilistic framework for decision-making.
- **Flexibility:** Can update beliefs incrementally as new data arrives.

Challenges of the Bayesian Method

- **Computationally Intensive:** Calculating posterior distributions can be complex.
- **Subjectivity in Priors:** Choice of prior can influence results, especially with limited data.
- **Requires Sufficient Data:** For robust posterior estimates, substantial data is often needed.

5.5] Association Rule:

-Association rule mining finds interesting associations and relationships among large sets of data items.

-This rule shows how frequently an item set occurs in a transaction.

-A typical example is a Market Based Analysis. Market Based Analysis is one of the key techniques used by large relations to show associations between items. It allows retailers to identify relationships between the items that people buy together frequently.

-Given a set of transactions, we can find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer

TID	Items
5	Bread, Milk, Diaper, Coke

Before we start defining the rule, let us first see the basic definitions. **Support Count()** – Frequency of occurrence of a itemset.

Here $\text{Support Count}(\{\text{Milk, Bread, Diaper}\})=2$

Frequent Itemset – An itemset whose support is greater than or equal to minsup threshold. **Association Rule** – An implication expression of the form $X \rightarrow Y$, where X and Y are any 2 itemsets.

Example: $\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$

Rule Evaluation Metrics –

- **Support(s)** – The number of transactions that include items in the {X} and {Y} parts of the rule as a percentage of the total number of transaction. It is a measure of how frequently the collection of items occur together as a percentage of all transactions.
- **Support = $\frac{\text{X+Y}}{\text{total}}$** – It is interpreted as fraction of transactions that contain both X and Y.
- **Confidence(c)** – It is the ratio of the no of transactions that includes all items in {B} as well as the no of transactions that includes all items in {A} to the no of transactions that includes all items in {A}.
- **Conf(X=>Y) = $\frac{\text{Support(X \cup Y)}}{\text{Support(X)}}$** – It measures how often each item in Y appears in transactions that contains items in X also.
- **Lift(l)** – The lift of the rule $X \Rightarrow Y$ is the confidence of the rule divided by the expected confidence, assuming that the itemsets X and Y are independent of each other. The expected confidence is the confidence divided by the frequency of {Y}.
- **Lift(X=>Y) = $\frac{\text{Conf(X=>Y)}}{\text{Support(Y)}}$** – Lift value near 1 indicates X and Y almost often appear together as expected, greater than 1 means they appear together more than expected and less than 1 means they appear less than expected. Greater lift values indicate stronger association.

Example – From the above table, $\{\text{Milk, Diaper}\} \Rightarrow \{\text{Beer}\}$

$s = \frac{\text{Support}(\{\text{Milk, Diaper, Beer}\})}{|T|}$

$= \frac{2}{5}$

$= 0.4$

$c = \frac{\text{Support}(\text{Milk, Diaper, Beer})}{\text{Support}(\text{Milk, Diaper})}$

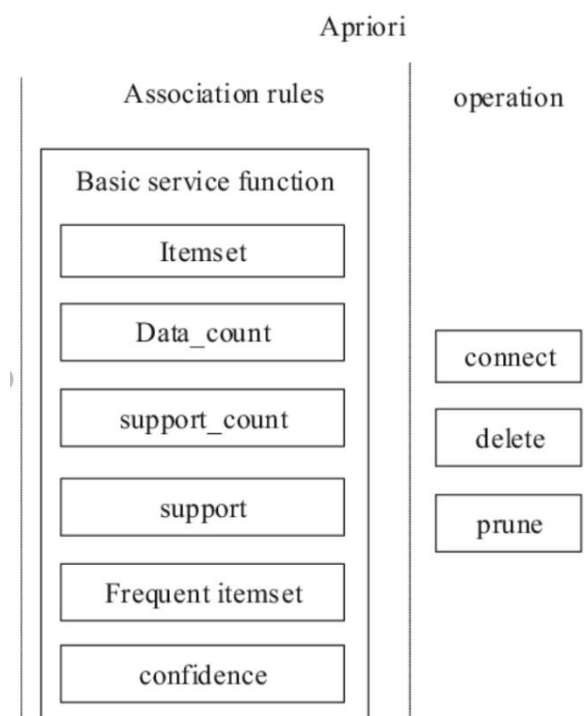
$= \frac{2}{3}$

$= 0.67$

$$\begin{aligned}
 I &= \text{Supp}(\{\text{Milk, Diaper, Beer}\}) - \text{Supp}(\{\text{Milk, Diaper}\}) * \text{Supp}(\{\text{Beer}\}) \\
 &= 0.4 / (0.6 * 0.6) \\
 &= 1.11
 \end{aligned}$$

-The Association rule is very useful in analyzing datasets. The data is collected using bar-code scanners in supermarkets. Such databases consist of a large number of transaction records which list all items bought by a customer on a single purchase. So the manager could know if certain groups of items are consistently purchased together and use this data for adjusting store layouts, cross-selling, promotions based on statistics

5.6] Structure of Association Rule:



The structure of an association rule is an if-then statement with two parts: an antecedent and a consequent:

- **Antecedent:** The item that is found in the data
 - **Consequent:** The item that is found in combination with the antecedent
- For example, "If a customer buys eggs, he is 80% likely to also purchase milk" is an association rule.

Association rules are used to find relationships between seemingly unrelated data. They are a key tool in market basket analysis and can also be used to gain insights into customer behavior.

To create association rules, data is searched for frequent if-then patterns. The strength of the association is measured using the confidence of the rule, which is the probability that the consequent is present given that the antecedent is present.

Other metrics used to measure associations between data items include support and lift:

- **Support:** The evidence of how frequent an item appears in the data
- **Lift:** A measure that compares the expected confidence and the actual confidence

5.7] Apriori Algorithm:

-Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

-To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called *Apriori property* which helps by reducing the search space.

Apriori Property –

All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure. Apriori assumes that

*‘All subsets of a frequent itemset must be frequent(Apriori property).
If an itemset is infrequent, all its supersets will be infrequent.’*

Before we start understanding the algorithm, go through some definitions which are explained in my previous post.

Consider the following dataset and we will find frequent itemsets and generate association rules for them.

TID	items
T1	I1, I2 , I5
T2	I2,I4
T3	I2,I3
T4	I1,I2,I4
T5	I1,I3
T6	I2,I3
T7	I1,I3
T8	I1,I2,I3,I5
T9	I1,I2,I3

minimum support count is 2
minimum confidence is 60%

Step-1: K=1

(I) Create a table containing support count of each item present in dataset – Called **C1(candidate set)**

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

(II) compare candidate set item's support count with minimum support count(here min_support=2 if support_count of candidate set items is less than min_support then remove those items). This gives us itemset L1.

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

Step-2: K=2

- Generate candidate set C2 using L1 (this is called join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common.
- Check all subsets of an itemset are frequent or not and if not frequent remove that itemset.(Example subset of {I1, I2} are {I1}, {I2} they are frequent.Check for each itemset)
- Now find support count of these itemsets by searching in dataset.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

(II) compare candidate (C2) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L2.

Itemset	sup_count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I2,I5	2

Step-3:

- Generate candidate set C3 using L2 (join step). Condition of joining L_{k-1} and L_{k-1} is that it should have (K-2) elements in common. So here, for L2, first element should match.
So itemset generated by joining L2 is {I1, I2, I3}{I1, I2, I5}{I1, I3, I5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}
- Check if all subsets of these itemsets are frequent or not and if not, then remove that itemset.(Here subset of {I1, I2, I3} are {I1, I2},{I2, I3},{I1, I3} which are frequent. For {I2, I3, I4}, subset {I3, I4} is not frequent so remove it. Similarly check for every itemset)
- find support count of these remaining itemset by searching in dataset.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

(II) Compare candidate (C3) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L3.

Itemset	sup_count
I1,I2,I3	2
I1,I2,I5	2

Step-4:

- Generate candidate set C4 using L3 (join step). Condition of joining L_{k-1} and L_{k-1} (K=4) is that, they should have (K-2) elements in common. So here, for L3, first 2 elements (items) should match.
- Check all subsets of these itemsets are frequent or not (Here itemset formed by joining L3 is {I1, I2, I3, I5} so its subset contains {I1, I3, I5}, which is not frequent). So no itemset in C4
- We stop here because no frequent itemsets are found further

5.8] General Association :

In **machine learning**, the term **general association** refers to the relationship or dependency between variables in a dataset.

Understanding and identifying associations is fundamental for building predictive models and uncovering patterns in data.

Associations can be direct or indirect and can take various forms, such as linear, non-linear, or even more complex relationships.

Key Concepts of Association in Machine Learning

1. Types of Associations

- **Linear Association:** A straight-line relationship between variables (e.g., $y=mx+cy = mx + c$).
- **Non-Linear Association:** A curved relationship, such as polynomial or exponential.
- **Monotonic Association:** A relationship where variables move in the same or opposite directions, but not necessarily linearly.
- **Complex Association:** High-dimensional or multi-variable dependencies that cannot be easily visualized or summarized by simple functions.

2. Measures of Association

Several statistical methods help quantify the strength and nature of associations:

- **Correlation Coefficients:**
 - Pearson's Correlation (linear relationships).
 - Spearman's Rank Correlation (monotonic relationships).
- **Mutual Information:** Measures any form of dependency, including non-linear.
- **Chi-Square Test:** Determines the association between categorical variables.
- **Covariance:** Measures how two variables change together.

3. Association vs. Causation

- Association does not imply causation.
 - Machine learning models detect patterns and dependencies but require causal analysis techniques (e.g., causal inference, randomized controlled trials) to establish cause-and-effect relationships.
-

Importance of Associations in Machine Learning

1. Feature Selection:

Identifying associations helps choose relevant features for predictive modelling. Strongly associated features with the target variable are often retained.

2. Feature Engineering:

Insights into associations can inspire the creation of new features, such as combining or transforming variables to better capture relationships.

3. Model Interpretability:

Understanding associations between features and target variables aids in interpreting model predictions.

4. Reducing Multicollinearity:

In regression models, high associations between features (multicollinearity) can distort coefficients and predictions.

Techniques to Capture Associations in Machine Learning Models

1. **Linear Models:**
Models like linear regression directly capture linear associations between predictors and the target variable.
 2. **Non-Linear Models:**
Decision trees, random forests, support vector machines, and neural networks capture more complex and non-linear associations.
 3. **Dimensionality Reduction:**
Techniques like PCA (Principal Component Analysis) identify hidden associations in high-dimensional datasets.
 4. **Dependency Modelling:**
 - Bayesian networks model probabilistic associations between variables.
 - Graph-based models represent associations explicitly as edges in a graph.
-

Real-World Applications of Associations in Machine Learning

1. **Recommendation Systems:**
Detect associations between users and items (e.g., collaborative filtering).
2. **Fraud Detection:**
Identify unusual associations in transaction patterns that indicate fraud.
3. **Healthcare:**
Analyze associations between symptoms, diseases, and treatments for diagnostic purposes.
4. **Market Basket Analysis:**
Identify associations between products in retail for cross-selling (e.g., "people who bought X also bought Y").

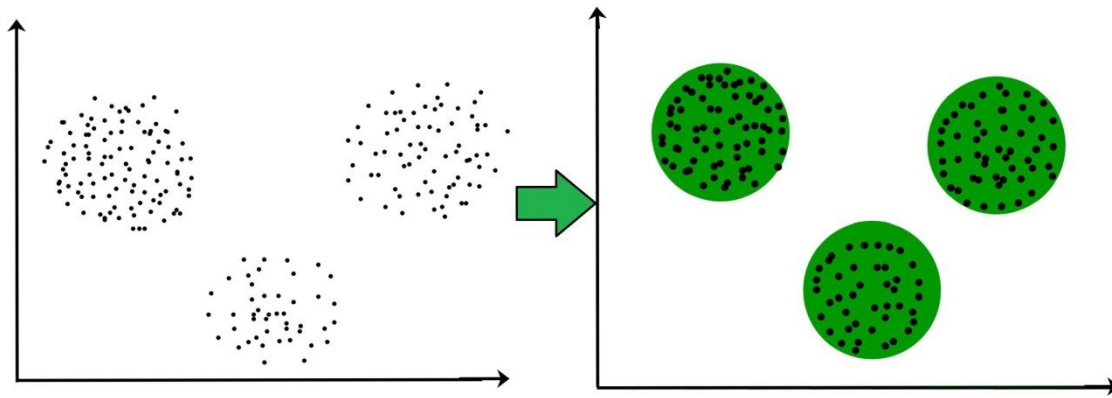
5.9] Clustering:

-The task of grouping data points based on their similarity with each other is called Clustering or Cluster Analysis.

-This method is defined under the branch of [Unsupervised Learning](#), which aims at gaining insights from unlabelled data points, that is, unlike [supervised learning](#) we don't have a target variable.

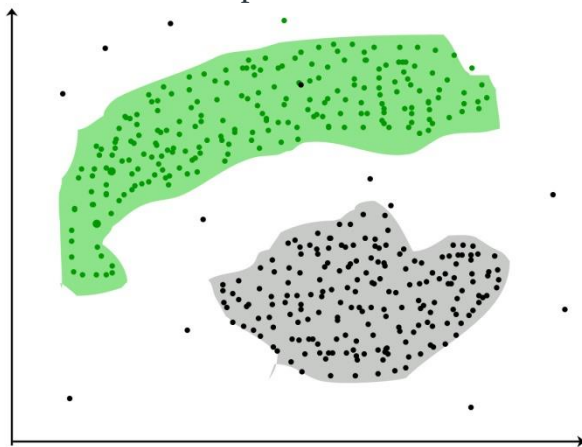
Clustering aims at forming groups of homogeneous data points from a heterogeneous dataset. It evaluates the similarity based on a metric like Euclidean distance, Cosine similarity, Manhattan distance, etc. and then group the points with highest similarity score together.

For Example, In the graph given below, we can clearly see that there are 3 circular clusters forming on the basis of distance.



-Now it is not necessary that the clusters formed must be circular in shape. The shape of clusters can be arbitrary. There are many algorithms that work well with detecting arbitrary shaped clusters.

-For example, In the below given graph we can see that the clusters formed are not circular in shape.



5.10] Clustering Methods:

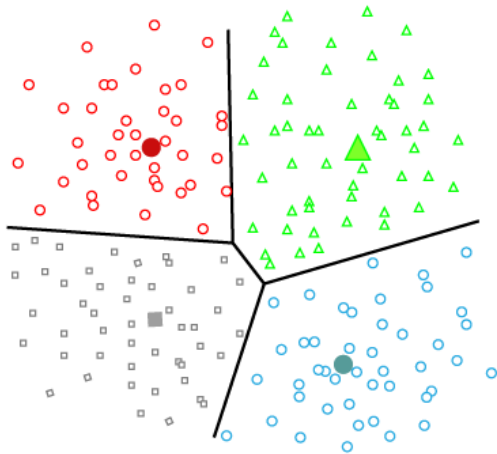
The clustering methods are broadly divided into **Hard clustering** (datapoint belongs to only one group) and **Soft Clustering** (data points can belong to another group also). But there are also other various approaches of Clustering exist.

Below are the main clustering methods used in Machine learning:

5.11] Partitioning Clustering:

It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the **centroid-based method**. The most common example of partitioning clustering is the **K-Means Clustering algorithm**.

In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.

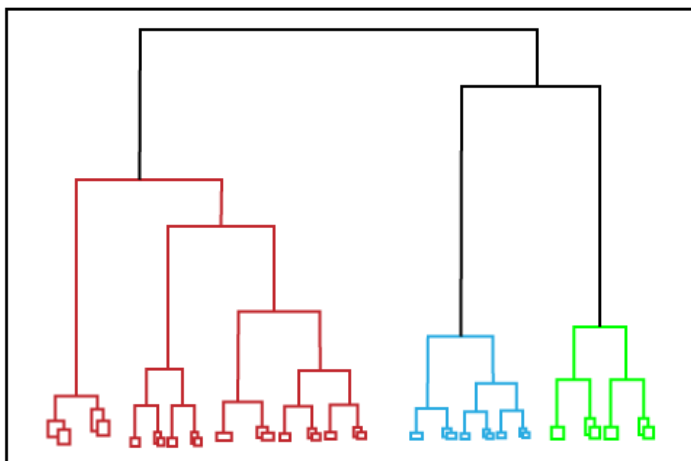


5.12] Hierarchical Methods:

Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created.

In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a **dendrogram**. The observations or any number of clusters can be selected by cutting the tree at the correct level.

The most common example of this method is the **Agglomerative Hierarchical algorithm**.



Unit VI: Exploring Data in R

6.1] Basic features of R:

The **R Language** is renowned for its extensive features that make it a powerful tool for data analysis, statistical computing, and visualization. Here are some of the key features of R:

1. Comprehensive Statistical Analysis:

- R language provides a wide array of statistical techniques, including linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, and clustering.

2. Advanced Data Visualization:

- With packages like ggplot2, plotly, and lattice, R excels at creating complex and aesthetically pleasing data visualizations, including plots, graphs, and charts.

3. Extensive Packages and Libraries:

- The Comprehensive R Archive Network (CRAN) hosts thousands of packages that extend R's capabilities in areas such as machine learning, data manipulation, bioinformatics, and more.

4. Open Source and Free:

- R is free to download and use, making it accessible to everyone. Its open-source nature encourages community contributions and continuous improvement.

5. Platform Independence:

- R is platform-independent, running on various operating systems, including Windows, macOS, and Linux, which ensures flexibility and ease of use across different environments.

6. Integration with Other Languages:

- R language can integrate with other programming languages such as C, C++, Python, Java, and SQL, allowing for seamless interaction with various data sources and computational processes.

7. Powerful Data Handling and Storage:

- R efficiently handles and stores data, supporting various data types and structures, including vectors, matrices, data frames, and lists.

8. Robust Community and Support:

- R has a vibrant and active community that provides extensive support through forums, mailing lists, and online resources, contributing to its rich ecosystem of packages and documentation.

9. Interactive Development Environment (IDE):

- RStudio, the most popular IDE for R, offers a user-friendly interface with features like syntax highlighting, code completion, and integrated tools for plotting, history, and debugging.

10. Reproducible Research:

- R supports reproducible research practices with tools like R Markdown and Knitr, enabling users to create dynamic reports, presentations, and documents that combine code, text, and visualizations.



6.2] Exploring R GUI:

The R Graphical User Interface (R GUI) is the standard graphical interface that comes with the base installation of R, a powerful programming language widely used for statistical computing and data visualization. Exploring R GUI involves understanding its components, features, and capabilities for running R scripts and managing your work. Here's a detailed explanation:

1. Accessing R GUI

- When you install R, the R GUI is included by default. It can be accessed by:
 - Clicking on the R shortcut installed on your system.
 - Running the R executable from your operating system's program directory.

2. Components of R GUI

a. Console

- The console is the primary interface for executing R commands interactively.
- You can type commands directly into the console, and the results are displayed immediately.
- It also displays error messages and diagnostic information.

b. Script Editor (Optional in R GUI)

- Allows you to write, edit, and save R scripts (code files with `.R` extension).
- Scripts can be executed line-by-line or in batches.
- Accessible through menus or toolbars depending on your R GUI version.

c. Menus and Toolbars

- The R GUI includes menus for commonly used tasks:
 - **File:** Open, save, and manage R scripts and workspaces.
 - **Edit:** Basic text editing features like copy, paste, and undo.
 - **Misc:** Access utilities such as viewing objects in memory.
 - **Packages:** Install, load, and manage R packages.
 - **Help:** Documentation and guidance for R functions.

d. Workspace and Environment

- The workspace displays all active variables, functions, and datasets.
- The environment can be managed using the GUI or commands (`ls()`, `rm()`).

e. Graphics Device

- A window that displays visual outputs such as plots and charts.
- Multiple graphics devices can be opened to compare visualizations.

f. Help System

- The GUI has built-in access to extensive documentation.
 - Type `?function_name` in the console or use the Help menu to search for help on specific functions.
-

3. Key Features of R GUI

a. Running Commands

- Directly execute R commands in the console.
- Example: `print("Hello, World!")` or `summary(data)`.

b. Managing Workspaces

- Save your current workspace for future use using `save.image()` or **File > Save Workspace**.
- Load previous workspaces to continue your analysis.

c. Script Execution

- Use the script editor to write and save longer scripts.
- Run the script by selecting portions of the code and pressing `Ctrl+R` or using menu options.

d. Installing Packages

- Install additional R packages using `install.packages("package_name")` or through the Packages menu.

e. Plotting and Visualization

- Generate plots and graphs that appear in a separate graphics window.
- Example: `plot(x, y)`.

f. Customization

- You can customize R GUI settings like fonts, colors, and console behavior for a more comfortable experience.
-

4. Advanced Alternatives to R GUI

While R GUI is sufficient for basic tasks, there are more advanced Integrated Development Environments (IDEs) that enhance functionality:

- **RStudio:** A popular and user-friendly IDE with features like syntax highlighting, integrated help, version control, and multiple panes for projects.
 - **Jupyter Notebook:** Allows interactive coding in R with visual outputs embedded inline.
-

5. Tips for Effective Use

- **Shortcuts:** Familiarize yourself with keyboard shortcuts for faster navigation.
 - **Help System:** Use the help system extensively to understand R functions and error messages.
 - **Save Progress:** Regularly save your scripts and workspace to avoid data loss.
 - **Package Management:** Keep your packages updated for new features and bug fixes.
-

6. Limitations of R GUI

- **Limited Features:** Compared to IDEs like RStudio, the R GUI lacks advanced features like project management and version control.
 - **Basic Text Editor:** The script editor in R GUI is functional but lacks features like syntax highlighting and auto-completion.
-

6.3] Data Frames & Lists:

A data frame is a table or a two-dimensional array-like structure in which each column contains values of one variable and each row contains one set of values from each column.

Following are the characteristics of a data frame.

- The column names should be non-empty.
- The row names should be unique.
- The data stored in a data frame can be of numeric, factor or character type.
- Each column should contain same number of data items.

Create Data Frame

```
# Create the data frame.
emp.data <- data.frame(
  emp_id = c (1:5),
  emp_name = c("Rick","Dan","Michelle","Ryan","Gary"),
  salary = c(623.3,515.2,611.0,729.0,843.25),

  start_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15",
"2014-05-11",
  "2015-03-27")),
  stringsAsFactors = FALSE
)
# Print the data frame.
print(emp.data)
```

When we execute the above code, it produces the following result –

	emp_id	emp_name	salary	start_date
1	1	Rick	623.30	2012-01-01
2	2	Dan	515.20	2013-09-23
3	3	Michelle	611.00	2014-11-15
4	4	Ryan	729.00	2014-05-11
5	5	Gary	843.25	2015-03-27

##List:

Lists are the R objects which contain elements of different types like – numbers, strings, vectors and another list inside it. A list can also contain a matrix or a function as its elements.

Created using **list ()** function.

Creating a List

```
# Create a list containing strings, numbers, vectors and a logical  
# values.  
list_data <- list("Red", "Green", c(21,32,11), TRUE, 51.23, 119.1)  
print(list_data)
```

When we execute the above code, it produces the following result –

```
[[1]]  
[1] "Red"  
  
[[2]]  
[1] "Green"  
  
[[3]]  
[1] 21 32 11  
  
[[4]]  
[1] TRUE  
  
[[5]]  
[1] 51.23  
  
[[6]]  
[1] 119.1  
Ⓢ
```

Naming List Elements:

The list elements can be given names and they can be accessed using these names.

```
# Create a list containing a vector, a matrix and a list.
```

```
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),
  list("green",12.3))

# Give names to the elements in the list.
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")

# Show the list.
print(list_data)
```

When we execute the above code, it produces the following result –

```
$`1st_Quarter`
[1] "Jan" "Feb" "Mar"
```

```
$A_Matrix
  [,1] [,2] [,3]
[1,]  3   5  -2
[2,]  9   1   8
```

```
$A_Inner_list
$A_Inner_list[[1]]
[1] "green"
```

```
$A_Inner_list[[2]]
[1] 12.3
```

Accessing List Elements:

Elements of the list can be accessed by the index of the element in the list. In case of named lists it can also be accessed using the names.

We continue to use the list in the above example –

```
# Create a list containing a vector, a matrix and a list.
list_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),
  list("green",12.3))

# Give names to the elements in the list.
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")

# Access the first element of the list.
print(list_data[1])

# Access the third element. As it is also a list, all its elements will be printed.
print(list_data[3])
```

```
# Access the list element using the name of the element.  
print(list_data$A_Matrix)
```

When we execute the above code, it produces the following result –

```
$`1st_Quarter`  
[1] "Jan" "Feb" "Mar"
```

```
$A_Inner_list  
$A_Inner_list[[1]]  
[1] "green"
```

```
$A_Inner_list[[2]]  
[1] 12.3
```

```
      [,1] [,2] [,3]  
[1,]   3   5  -2  
[2,]   9   1   8
```

Manipulating List Elements:

We can add, delete and update list elements as shown below. We can add and delete elements only at the end of a list. But we can update any element.

```
# Create a list containing a vector, a matrix and a list.  
list_data <- list(c("Jan", "Feb", "Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),  
  list("green", 12.3))  
  
# Give names to the elements in the list.  
names(list_data) <- c("1st Quarter", "A_Matrix", "A Inner list")  
  
# Add element at the end of the list.  
list_data[4] <- "New element"  
print(list_data[4])  
  
# Remove the last element.  
list_data[4] <- NULL  
  
# Print the 4th Element.  
print(list_data[4])  
  
# Update the 3rd Element.  
list_data[3] <- "updated element"  
print(list_data[3])
```

When we execute the above code, it produces the following result –

```
[[1]]
```

```
[1] "New element"
```

```
$<NA>
```

```
NULL
```

```
$`A Inner list`
```

```
[1] "updated element"
```

Merging Lists

You can merge many lists into one list by placing all the lists inside one `list()` function.

```
# Create two lists.  
list1 <- list(1,2,3)  
list2 <- list("Sun", "Mon", "Tue")  
  
# Merge the two lists.  
merged.list <- c(list1, list2)  
  
# Print the merged list.  
print(merged.list)
```

When we execute the above code, it produces the following result –

```
[[1]]
```

```
[1] 1
```

```
[[2]]
```

```
[1] 2
```

```
[[3]]
```

```
[1] 3
```

```
[[4]]
```

```
[1] "Sun"
```

```
[[5]]
```

```
[1] "Mon"
```

```
[[6]]  
[1] "Tue"
```

Converting List to Vector:

A list can be converted to a vector so that the elements of the vector can be used for further manipulation. All the arithmetic operations on vectors can be applied after the list is converted into vectors. To do this conversion, we use the **unlist()** function. It takes the list as input and produces a vector

```
# Create lists.  
list1 <- list(1:5)  
print(list1)  
  
list2 <- list(10:14)  
print(list2)  
  
# Convert the lists to vectors.  
v1 <- unlist(list1)  
v2 <- unlist(list2)  
  
print(v1)  
print(v2)  
  
# Now add the vectors  
result <- v1+v2  
print(result)
```

When we execute the above code, it produces the following result –

```
[[1]]  
[1] 1 2 3 4 5  
  
[[1]]  
[1] 10 11 12 13 14  
  
[1] 1 2 3 4 5  
[1] 10 11 12 13 14  
[1] 11 13 15 17 19
```

6.4] Handling Data in R Workspace:

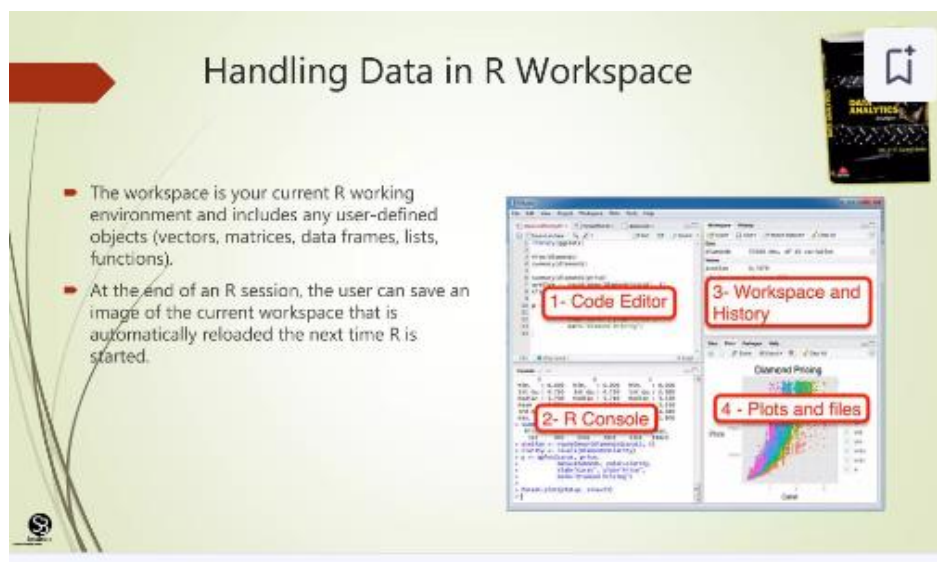
-R Programming Language is used for statistics and data analytics purposes. Importing and exporting of data is often used in all these applications of R

programming.

-R language has the ability to read different types of files such as comma-separated values (CSV) files, text files, excel sheets and files, SPSS files, SAS files, etc.

-R allows its users to work smoothly with the systems directories with the help of some pre-defined functions that take the path of the directory as the argument or return the path of the current directory that the user is working on. Below are some directory functions in R:

- **getwd():** This function is used to get the current working directory being used by R.
- **setwd():** This function in R is used to change the path of current working directory and the path of the directory is passed as argument in the function.



- **Example:**
- `setwd("C:/RExamples/")`

OR

```
setwd("C:\\RExamples\\")
```

- **list.files():** This function lists all files and folders present in current working directory.

Importing Files in R

Let us take some basic files to import in R to learn the approach of importing data files:

❖ *Importing Text Files*

In R language, text files can be read using **read.table()** function.

Syntax:

```
read.table(filename, header = FALSE, sep = "")
```

Parameters:

header represents if the file contains header row or not

sep represents the delimiter value used in file

To know about all the arguments of **read.table()**, execute below command in R:

```
help("read.table")
```

Example:

Suppose a file is present in the current working directory and using R programming, import the data from that particular text file and the content of text file is as shown:

```
100 A a
200 B b
300 C c
400 D d
500 E e
600 F f
```

Values are separated by white spaces.

```
# Check current working directory
```

```
getwd()
```

```
# Get content into a data frame
```

```
data <- read.table("TextFileExample.txt",
```

```
                header = FALSE, sep = " ")
```

```
# Printing content of Text File
```

```
print(data)
```



```
# Print the class of data
```

```
print(class(data))
```

Output:

```
[1] "C:/Users/xyz/Documents"
```

```
  V1 V2 V3
```

```
1 100 A a
```

```
2 200 B b
```

```
3 300 C c
```

```
4 400 D d
```

```
5 500 E e
```

```
6 600 F f
```

```
[1] "data.frame"
```

❖ Importing CSV Files

Comma separated values or CSV files can be imported and read in R using `read.csv()` function.

Syntax:

```
read.csv(filename, header = FALSE, sep = "")
```

Parameters:

header represents if the file contains header row or not

sep represents the delimiter value used in file

To know about all the arguments of `read.csv()`, execute below command in R:

```
help("read.csv")
```

Example:

Suppose a csv file is present in the current working directory and the content of file is as shown:

	A	B	C	D	E
1	100	AB	ab		
2	200	CD	cd		
3	300	EF	ef		
4	400	GH	gh		
5	500	IJ	ij		
6					

```
# Check current working directory

getwd()

# Get content into a data frame

data <- read.csv("CSVFileExample.csv",

                header = FALSE, sep = "\t")

# Printing content of Text File

print(data)

# Print the class of data

print(class(data))
```

Output:

```
[1] "C:/Users/xyz/Documents"
  V1 V2 V3
1 100 AB ab
2 200 CD cd
3 300 EF ef
4 400 GH gh
5 500 IJ ij
[1] "data.frame"
```

❖ *Importing Excel File:*

To read and import the excel files, “**xlsx**” package is required to use the **read.xlsx()** function. To read “.xls” excel files, “**gdata**” package is required to use **read.xls()** function.

Syntax:

```
read.xlsx(filename, sheetIndex)
```

OR

```
read.xlsx(filename, sheetName)
```

Parameters:

sheetIndex specifies number of sheet

sheetName specifies name of sheet

To know about all the arguments of `read.xlsx()`, execute below command in R:

```
help("read.xlsx")
```

Example:

Suppose a `xlsx` file is present in the current working directory and the content of file is as shown:

	A	B	C	D
1	1000	ABC	abc	
2	2000	DEF	def	
3	3000	GHI	ghi	
4	4000	JKL	jkl	
5	5000	MNO	mno	
6				

```
# Install xlsx package
```

```
install.packages("xlsx")
```

```
library(xlsx)
```

```
# Check current working directory
```

```
getwd()
```

```
# Get content into a data frame
```

```
data <- read.xlsx("ExcelExample.xlsx",
```

```
    sheetIndex = 1,
```

```
    header = FALSE)
```

```
# Printing content of Text File
```

```
print(data)
```

```
# Print the class of data
```

```
print(class(data))
```

Output:

```
[1] "C:/Users/xyz/Documents"
```

```
  X1 X2 X3
```

```
1 1000 ABC abc
```

```
2 2000 DEF def
```

```
3 3000 GHI ghi
```

```
4 4000 JKL jkl
```

```
5 5000 MNO mno
```

```
[1] "data.frame"
```

❖ *Exporting files in R*

Below are some methods to export the data to a file in R:

- **Using console**

cat() function in R is used to output the object to console. It can be also used as redirecting the output to a particular file.

Syntax:

```
cat(..., file)
```

Parameter:

file specifies the filename to which output has to be redirected

To know about all the arguments of **cat()**, execute the below command in R:

```
help("cat")
```

Example:

```
str = "World"
```

```
# Redirect Output to file
```

```
cat("Hello, ", str, file = "catExample.txt")
```

Output:

Above code creates a new file and redirects the output of **cat()**. The contents of the file are shown below after executing the code-

```
Hello, World
```

- **Using sink() function:**
sink() function is used to redirect all the outputs from **cat()** and **print()** to the given filename.

Syntax:

- `sink(filename)` # begins redirecting output to file

- `sink()`

To know about all the arguments of **sink()**, execute below command in R:

```
help("sink")
```

Example:

```
# Begin redirecting output

sink("SinkExample.txt")

x <- c(1, 3, 4, 5, 10)

print(mean(x))

print(class(x))

print(median(x))

sink()
```

Output:

The above code creates a new file and redirects the output. The contents of the file are shown below after executing the code-

```
[1] 4.6
[1] "numeric"
[1] 4
```

❖ **Writing to CSV files**

A matrix or data-frame object can be redirected and written to csv file using **write.csv()** function.

Syntax: *write.csv(x, file)*

Parameter:

file specifies the file name used for writing

To know about all the arguments of `write.csv()`, execute below command in R:

```
help("write.csv")
```

Example:

```
# Create vectors

x <- c(1, 3, 4, 5, 10)

y <- c(2, 4, 6, 8, 10)

z <- c(10, 12, 14, 16, 18)

# Create matrix

data <- cbind(x, y, z)

# Writing matrix to CSV File

write.csv(data, file = "CSVWrite.csv", row.names = FALSE)
```

Output:

Above code creates a new file and redirects the output. The contents of the file is shown below after executing the code-

	A	B	C	D
1	x	y	z	
2	1	2	10	
3	3	4	12	
4	4	6	14	
5	5	8	16	
6	10	10	18	
7				

4.5] Reading Data Sets & Exporting Data from R:



In R, reading datasets and exporting data are essential tasks for working with data. R supports multiple file formats for input and output, and it provides functions that make importing and exporting data straightforward. Below is a detailed guide on reading datasets into R and exporting data from R:

1. Reading Data into R

R provides several functions to read datasets from different file formats like CSV, Excel, and more.

a. Reading CSV Files

CSV (Comma Separated Values) files are one of the most commonly used formats for storing data. R can easily read CSV files using the `read.csv()` function.

```
data <- read.csv("path_to_file.csv")
```

- **Arguments:**
 - `"path_to_file.csv"`: Path to the CSV file (can be local or URL).
 - `header = TRUE`: Default is `TRUE`, which assumes the first row contains column names.
 - `sep = ","`: The separator (default is a comma for CSV files).
 - `stringsAsFactors = FALSE`: To prevent strings from being converted into factors (especially in older versions of R).

b. Reading Excel Files

For reading Excel files, you can use the `readxl` or `openxlsx` package. Here's how to use `readxl` to read `.xlsx` files:

1. **Install the package (if not already installed):**
2. `install.packages("readxl")`
3. **Read Excel files:**
4. `library(readxl)`

```
5. data <- read_excel("path_to_file.xlsx")
```

- **Arguments:**
 - "path_to_file.xlsx": Path to the Excel file.
 - sheet = 1: Specify which sheet to load (if there are multiple sheets).

c. Reading Text Files

You can read tab-delimited text files using the `read.table()` function.

```
data <- read.table("path_to_file.txt", header = TRUE, sep = "\t")
```

- **Arguments:**
 - header = TRUE: Assumes the first line contains column names.
 - sep = "\t": Specifies that the columns are tab-separated.

d. Reading RData Files

RData files are R's native format for saving and loading R objects. You can load `.RData` files into R using the `load()` function.

```
load("path_to_file.RData")
```

- This function will load all objects stored in the `.RData` file into the R environment.

2. Exporting Data from R

Once you have manipulated or analyzed your data in R, you can export it to various formats. Below are common methods for exporting data:

a. Exporting to CSV Files

To export a data frame to a CSV file, use the `write.csv()` function:

```
write.csv(data, "output_file.csv", row.names = FALSE)
```

- **Arguments:**
 - data: The data frame to be written to a CSV file.
 - "output_file.csv": The name of the output file.
 - row.names = FALSE: Avoid writing row names (index) into the file.

b. Exporting to Excel Files

To export data to an Excel file, you can use the `openxlsx` or `writexl` package. Here's how to use the `writexl` package:

1. **Install the package (if not already installed):**
2. `install.packages("writexl")`

3. Export data to Excel:

```
4. library(writexl)
5. write_xlsx(data, "output_file.xlsx")
```

- **Arguments:**

- `data`: The data frame to be written to an Excel file.
- `"output_file.xlsx"`: The name of the Excel file.

c. Exporting to Text Files

You can export data to a tab-delimited text file using `write.table()`:

```
write.table(data, "output_file.txt", sep = "\t", row.names = FALSE)
```

- **Arguments:**

- `sep = "\t"`: Tab-separated values.
- `row.names = FALSE`: Do not include row names in the file.

d. Exporting to RData Files

To save your R workspace or specific objects, use the `save()` function, which saves objects to an RData file:

```
save(data, file = "output_file.RData")
```

- **Arguments:**

- `data`: The object to save.
- `file = "output_file.RData"`: The path to the output file.

e. Exporting to JSON

For exporting data to JSON format, you can use the `jsonlite` package.

1. **Install the package:**

```
2. install.packages("jsonlite")
```

3. **Export data to JSON:**

```
4. library(jsonlite)
5. write_json(data, "output_file.json")
```

- **Arguments:**

- `data`: The data frame or list to be written to a JSON file.
- `"output_file.json"`: The name of the output file.

3. Additional Considerations

a. Handling Large Files

- When reading large datasets, you can use the `data.table` package's `fread()` function, which is faster than `read.csv()` for large CSV files:

- `install.packages("data.table")`
- `library(data.table)`
- `data <- fread("large_file.csv")`

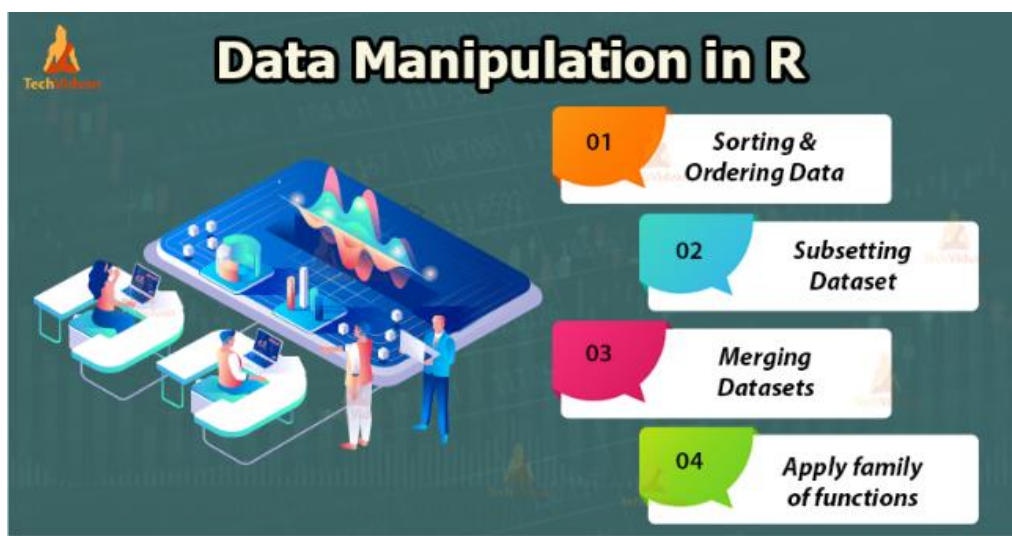
b. Checking the Data after Import

- After reading data, it's good practice to inspect it:
- `head(data)` # View the first few rows
- `summary(data)` # Get summary statistics
- `str(data)` # View structure of the data

c. Data Pre-processing

- Before exporting, ensure your data is cleaned, and unnecessary columns or rows are removed.
 - Use functions like `subset()`, `select()`, or `filter()` from the `dplyr` package for such tasks.

6.6] Manipulating & Processing Data in R:



Manipulating and processing data in R are central tasks in data analysis. R provides powerful tools and libraries for cleaning, transforming, summarizing, and visualizing data. Below is a detailed guide to common data manipulation and processing techniques in R.

1. Basic Data Manipulation

a. Selecting and Subsetting Data

- **Selecting specific columns:**
- `selected_data <- data[, c("column1", "column2")]`
- **Subsetting rows based on conditions:**
- `subset_data <- data[data$column1 > 10, ]`
- **Using `subset()` function:**
- `subset_data <- subset(data, column1 > 10 & column2 == "value")`

b. Adding and Modifying Columns

- **Creating new columns:**
- `data$new_column <- data$column1 + data$column2`
- **Modifying existing columns:**
- `data$column1 <- data$column1 * 2`

c. Renaming Columns

- Using `names()`:
- `names(data)[names(data) == "old_name"] <- "new_name"`
- Using `dplyr`:
- `library(dplyr)`
- `data <- rename(data, new_name = old_name)`

d. Removing Columns or Rows

- **Removing columns:**
 - `data$column1 <- NULL`
 - **Removing rows based on conditions:**
 - `data <- data[data$column1 <= 10, ]`
-

2. Data Transformation with `dplyr`

The `dplyr` package provides a grammar of data manipulation, making complex transformations straightforward.

a. Installing and Loading `dplyr`

```
install.packages("dplyr")  
library(dplyr)
```

b. Common `dplyr` Functions

- **`select()`:** Choose specific columns.
- `selected_data <- select(data, column1, column2)`
- **`filter()`:** Filter rows based on conditions.
- `filtered_data <- filter(data, column1 > 10, column2 == "value")`
- **`mutate()`:** Add or modify columns.
- `data <- mutate(data, new_column = column1 + column2)`
- **`arrange()`:** Sort data by columns.
- `sorted_data <- arrange(data, column1, desc(column2))`

- **summarize() and group_by():** Aggregate data.
- `summary_data <- data %>%`
- `group_by(group_column) %>%`
- `summarize(mean_value = mean(column1, na.rm = TRUE),`
- `count = n())`

c. Chaining with the Pipe Operator (%>%)

The pipe operator %>% passes the output of one function to the next, enabling concise and readable code.

```
result <- data %>%
  filter(column1 > 10) %>%
  mutate(new_column = column1 + column2) %>%
  select(new_column, column3)
```

3. Handling Missing Data

a. Identifying Missing Values

- **Check for missing values:**
- `any(is.na(data))` # Returns TRUE if any NA values exist
- **Count missing values per column:**
- `colSums(is.na(data))`

b. Removing Missing Values

- **Remove rows with missing values:**
- `cleaned_data <- na.omit(data)`
- **Remove specific columns with many missing values:**
- `data <- data[, colSums(is.na(data)) < threshold]`

c. Replacing Missing Values

- **Impute with a fixed value:**
 - `data$column1[is.na(data$column1)] <- 0`
 - **Impute with mean or median:**
 - `data$column1[is.na(data$column1)] <- mean(data$column1, na.rm = TRUE)`
-

4. Data Reshaping

a. Reshaping Data from Wide to Long Format

Using tidyr:

```
install.packages("tidyr")
library(tidyr)

long_data <- pivot_longer(data, cols = c(column1, column2),
                          names_to = "variable", values_to = "value")
```

b. Reshaping Data from Long to Wide Format

```
wide_data <- pivot_wider(data, names_from = variable, values_from = value)
```

5. Sorting and Ordering Data

a. Sorting with Base R

- **Sort rows:**
- `sorted_data <- data[order(data$column1, decreasing = TRUE), ]`

b. Sorting with dplyr

```
sorted_data <- arrange(data, desc(column1))
```

6. Combining and Merging Data

a. Row Binding

- Combine datasets with the same columns:
- `combined_data <- rbind(data1, data2)`

b. Column Binding

- Combine datasets with the same rows:
- `combined_data <- cbind(data1, data2)`

c. Merging Datasets

- Merge two datasets based on a common column:
 - `merged_data <- merge(data1, data2, by = "common_column")`
 - Use dplyr for joins:
 - `left_joined_data <- left_join(data1, data2, by = "common_column")`
-

7. Data Aggregation and Summarization

a. Using Base R

- Aggregate data:
- `aggregated_data <- aggregate(column1 ~ group_column, data, mean)`

b. Using dplyr

- Summarize grouped data:
 - `summary_data <- data %>%`
 - `group_by(group_column) %>%`
 - `summarize(mean_value = mean(column1, na.rm = TRUE))`
-

8. Advanced Processing with `data.table`

For large datasets, `data.table` is more efficient.

a. Installing and Loading `data.table`

```
install.packages("data.table")  
library(data.table)
```

b. Converting Data Frame to `data.table`

```
data <- as.data.table(data)
```

c. Syntax in `data.table`

- Filter rows:
`filtered_data <- data[column1 > 10]`
 - Select columns:
`selected_data <- data[, .(column1, column2)]`
 - Add/modify columns:
`data[, new_column := column1 + column2]`
 - Group by and summarize:
`summary_data <- data[, .(mean_value = mean(column1, na.rm = TRUE)),
by = group_column]`
-

.

Unit I :

Introduction of DBMS (Database Management System)

A **Database Management System (DBMS)** is a software solution designed to efficiently **manage, organize, and retrieve data** in a structured manner. It serves as a critical component in **modern computing, enabling** organizations to **store, manipulate, and secure** their data effectively. From small applications to enterprise systems, DBMS plays a vital role in supporting **data-driven decision-making** and operational efficiency.

In this article, we will explain the **key concepts, benefits, and types of Database Management Systems (DBMS)**. We'll also cover how DBMS solutions work, why they're important for modern applications, and what features they offer to ensure **data integrity, security, and efficient retrieval**.

What is a DBMS?

A [DBMS](#) is a system that allows users to **create, modify, and query** databases while ensuring **data integrity, security, and efficient data access**. Unlike **traditional file systems**, DBMS minimizes **data redundancy**, prevents inconsistencies, and simplifies data management with features like concurrent access and **backup mechanisms**. It organizes data into **tables, views, schemas, and reports**, providing a structured approach to data management.

Example:

A university database can store and manage student information, faculty records, and administrative data, allowing seamless retrieval, insertion, and deletion of information as required.

Key Features of DBMS

1. **Data Modeling:** Tools to create and modify data models, defining the structure and relationships within the database.
2. **Data Storage and Retrieval:** Efficient mechanisms for storing data and executing queries to retrieve it quickly.
3. **Concurrency Control:** Ensures multiple users can access the database simultaneously without conflicts.
4. **Data Integrity and Security:** Enforces rules to maintain accurate and secure data, including access controls and encryption.
5. **Backup and Recovery:** Protects data with regular backups and enables recovery in case of system failures.

Types of DBMS

There are several types of Database Management Systems (DBMS), each tailored to different data structures, scalability requirements, and application needs. The most common types are as follows:

1. Relational Database Management System (RDBMS)

[RDBMS](#) organizes data into tables (relations) composed of rows and columns. It uses primary keys to uniquely identify rows and foreign keys to establish relationships between tables. Queries are written in **SQL (Structured Query Language)**, which allows for efficient data manipulation and retrieval.

Examples: MySQL, Oracle, Microsoft SQL Server and Postgre SQL.

2. NoSQL DBMS

[NoSQL systems](#) are designed to handle **large-scale data** and provide high performance for scenarios where **relational models** might be restrictive. They store data in various non-relational formats, such as **key-value pairs**, **documents**, **graphs**, or **columns**. These flexible data models enable rapid scaling and are well-suited for unstructured or semi-structured data.

Examples: MongoDB, Cassandra, DynamoDB and Redis.

3. Object-Oriented DBMS (OODBMS)

[OODBMS](#) integrates object-oriented programming concepts into the **database environment**, allowing data to be stored as objects. This approach supports complex data types and relationships, making it ideal for applications requiring advanced data modeling and **real-world simulations**.

Examples: ObjectDB, db4o.

Database Languages

Database languages are specialized sets of **commands** and **instructions** used to define, manipulate, and control data within a database. Each language type plays a distinct role in database management, ensuring efficient **storage**, **retrieval**, and security of data. The primary database languages include:

1. Data Definition Language (DDL)

[DDL](#) is the short name for Data Definition Language, which deals with database schemas and descriptions, of how the data should reside in the database.

- **CREATE:** to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- **ALTER:** alters the structure of the existing database
- **DROP:** delete objects from the database
- **TRUNCATE:** remove all records from a table, including all spaces allocated for the records are removed
- **COMMENT:** add comments to the data dictionary
- **RENAME:** rename an object

2. Data Manipulation Language (DML)

[DML](#) focuses on manipulating the data stored in the database, enabling users to retrieve, add, update, and delete data.

- **SELECT:** retrieve data from a database
- **INSERT:** insert data into a table
- **UPDATE:** updates existing data within a table
- **DELETE:** Delete all records from a database table
- **MERGE:** UPSERT operation (insert or update)
- **CALL:** call a PL/SQL or Java subprogram
- **EXPLAIN PLAN:** interpretation of the data access path
- **LOCK TABLE:** concurrency Control

3. Data Control Language (DCL)

[DCL](#) commands manage access permissions, ensuring data security by controlling who can perform certain actions on the database.

- **GRANT:** Provides specific privileges to a user (e.g., SELECT, INSERT).
- **REVOKE:** Removes previously granted permissions from a user

4. Transaction Control Language (TCL)

[TCL](#) commands oversee transactional data to maintain consistency, reliability, and atomicity.

- **ROLLBACK:** Undoes changes made during a transaction.
- **COMMIT:** Saves all changes made during a transaction.
- **SAVEPOINT:** Sets a point within a transaction to which one can later roll back.

5. Data Query Language (DQL)

[DQL](#) is a subset of DML, specifically focused on data retrieval.

- **SELECT:** The primary DQL command, used to query data from the database without altering its structure or contents.

Advantages of Database Management System

The advantages of database management systems are:

1. **Data Security:** The more accessible and usable the database, the more it is prone to security issues. As the number of users increases, the data transferring or data sharing rate also increases thus increasing the risk of data security. It is widely used in the corporate world where companies invest large amounts of money, time, and effort to ensure data is secure and used properly. A Database Management System (DBMS) provides a better platform for data privacy and security policies thus, helping companies to improve Data Security.
2. **Data integration:** Due to the Database Management System we have access to well-managed and synchronized forms of data thus it makes data handling very easy and gives an integrated view of how a particular organization is working and also helps to keep track of how one segment of the company affects another segment.
3. **Data abstraction:** The major purpose of a [database](#) system is to provide users with an abstract view of the data. Since many complex algorithms are used by the developers to increase the efficiency of databases that are being hidden by the users through various data abstraction levels to allow users to easily interact with the system.
4. **Reduction in data Redundancy:** When working with a structured database, DBMS provides the feature to prevent the input of duplicate items in the database. for e.g. – If there are two same students in different rows, then one of the duplicate data will be deleted.
5. **Data sharing:** A DBMS provides a platform for sharing data across multiple applications and users, which can increase productivity and collaboration.
6. **Data consistency and accuracy:** DBMS ensures that data is consistent and accurate by enforcing data integrity constraints and preventing data duplication. This helps to eliminate data discrepancies and errors that can occur when data is stored and managed manually.
7. **Data organization:** A DBMS provides a systematic approach to organizing data in a structured way, which makes it easier to retrieve and manage data efficiently.
8. **Efficient data access and retrieval:** DBMS allows for efficient data access and retrieval by providing indexing and query optimization techniques that speed up data

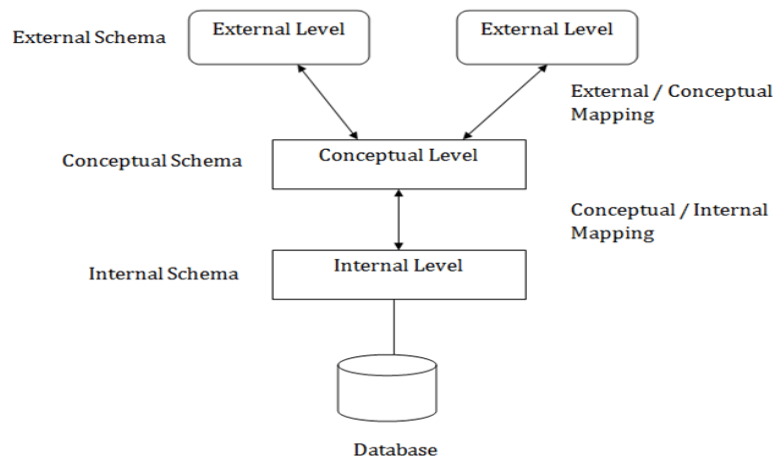
retrieval. This reduces the time required to process large volumes of data and increases the overall performance of the system.

9. **Concurrency and maintained Atomicity** : That means, if some operation is performed on one particular table of the database, then the change must be reflected for the entire database. The DBMS allows concurrent access to multiple users by using the synchronization technique.
10. **Scalability and flexibility**: DBMS is highly scalable and can easily accommodate changes in data volumes and user requirements. DBMS can easily handle large volumes of data, and can scale up or down depending on the needs of the organization. It provides flexibility in data storage, retrieval, and manipulation, allowing users to easily modify the structure and content of the database as needed.

Three schema Architecture

- The three schema architecture is also called ANSI/SPARC architecture or three-level architecture.
- This framework is used to describe the structure of a specific database system.
- The three schema architecture is also used to separate the user applications and physical database.
- The three schema architecture contains three-levels. It breaks the database down into three different categories.

The three-schema architecture is as follows:



In the above diagram:

- It shows the DBMS architecture.
- Mapping is used to transform the request and response between various database levels of architecture.
- Mapping is not good for small DBMS because it takes more time.
- In External / Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- In Conceptual / Internal mapping, DBMS transform the request from the conceptual to internal level.

Objectives of Three schema Architecture

The main objective of three level architecture is to enable multiple users to access the same data with a personalized view while storing the underlying data only once. Thus it separates the user's view from the physical structure of the database. This separation is desirable for the following reasons:

- Different users need different views of the same data.
- The approach in which a particular user needs to see the data may change over time.
- The users of the database should not worry about the physical implementation and internal workings of the database such as data compression and encryption techniques, hashing, optimization of the internal structures etc.
- All users should be able to access the same data according to their requirements.
- DBA should be able to change the conceptual structure of the database without affecting the user's
- Internal structure of the database should be unaffected by changes to physical aspects of the storage.
-

What is Data Independence in DBMS?

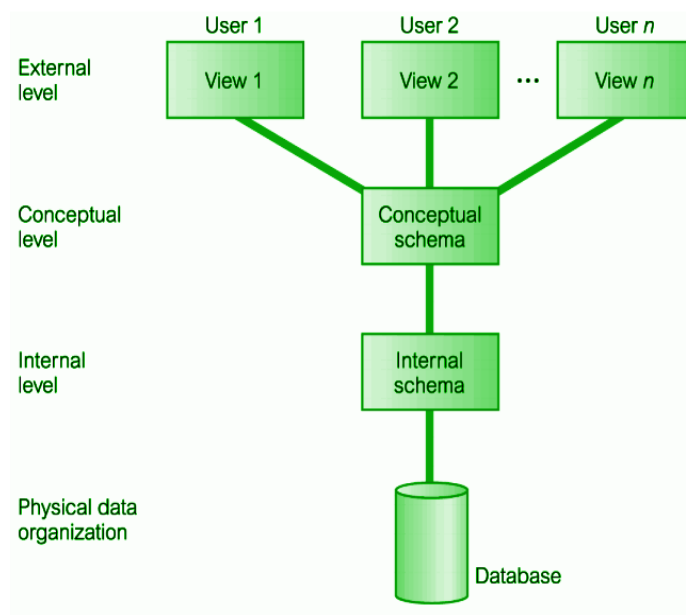
Data independence is a property of a database management system by which we can change the database schema at one level of the database system without changing the database schema at the next higher level. In this article, we will learn in full detail about data independence and will also see its types. If you read it completely, you will understand it easily.

What is Data Independence in DBMS?

In the context of a database management system, data independence is the feature that allows the schema of one layer of the database system to be changed without any impact on the schema of the next higher level of the database system. ” Through data independence, we can build an environment in which data is independent of all programs, and through the three schema architectures, data independence will be more understandable. Data via two card stencils along with centralized DBMS data is a form of transparency that has value for someone.

It can be summed up as a sort of immunity of user applications that adjusts correctly and does not change addresses, imparting the class of data and their order. I want the separate applications not to be forced to deal with data representation and storage specifics because this decreases quality and flexibility. DBMS permits you to see data with such a generalized sight. It actually means that the ability to change the structure of the lower-level schema

without presenting the upper-level schema is called data



independence.

Types of Data Independence

There are two types of data independence.

- logical data independence
- Physical data independence

Logical Data Independence

- Changing the logical schema (conceptual level) without changing the external schema (view level) is called logical data independence.
- It is used to keep the external schema separate from the logical schema.
- If we make any changes at the conceptual level of data, it does not affect the view level.
- This happens at the user interface level.
- For example, it is possible to add or delete new entities, attributes to the conceptual schema without making any changes to the external schema.

Physical Data Independence

- Making changes to the physical schema without changing the logical schema is called physical data independence.
- If we change the storage size of the database system server, it will not affect the conceptual structure of the database.
- It is used to keep the conceptual level separate from the internal level.
- This happens at the logical interface level.
- Example – Changing the location of the database from C drive to D drive.

Difference between Schema and Instance in DBMS

“**Schema**” and “**Instance**” are key ideas in a database management system (DBMS) that help organize and manage data. A schema can be referred to as the blueprint of the database while an instance is the actual contents of the database at a given point of time. This article will look at these ideas in detail to understand their importance, their difference, and how they relate to each other in a DBMS.

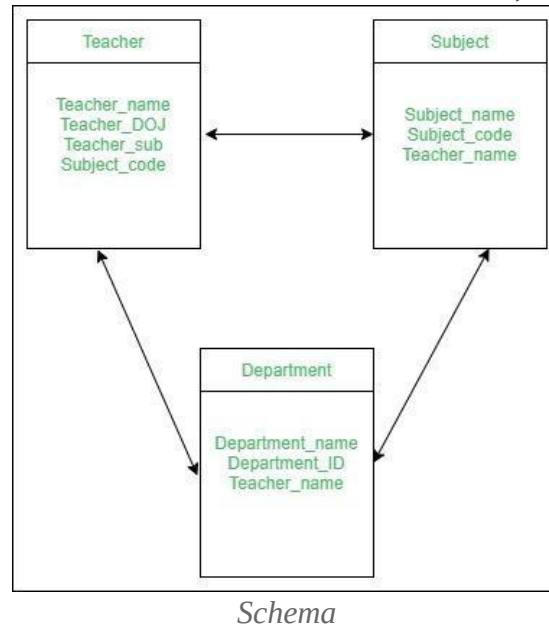
In a Database Management System (DBMS), the **schema** refers to the overall design or blueprint of the database, describing its structure (like tables, columns, and relationships). It remains relatively stable over time. On the other hand, an **instance** represents the actual data

within the database at any particular moment, which can change frequently as the database is updated.

For more in-depth learning about DBMS concepts and other computer science topics, consider exploring [GeeksforGeeks' GATE CS Self-Paced course](#) to enhance your knowledge further.

What is Schema?

Schema is the overall description of the database. The basic structure of how the data will be stored in the database is called schema. In DBMS, the term schema refers to the architecture of the database which describes how it will appear or will be constructed. It describes the organization of data such as tables, relationships as well as constraints. A schema is a template that dictates how data items in a database will be stored, arranged, and accessed.



Schema is of three types: Logical Schema, Physical Schema and view Schema.

- **Logical Schema** – It describes the database designed at a logical level.
- **Physical Schema** – It describes the database designed at the physical level.
- **View Schema** – It defines the design of the database at the view level.

Advantages of Schema

- **Consistency:** Guarantees proper storage of data in order to allow easy access and expandability.
- **Structure:** Helps in easy arrangement of the data base in an organized manner and hence makes it easy to comprehend.
- **Data Integrity:** Puts in place restrictions that ensure the data's maintaining of its accuracy and subsequent reliability.

Disadvantages of Schema

- **Rigidity:** Schemas, defined, may be rigid for alteration, and may take a huge amount of effort in order to alter the scheme.
- **Complexity:** Developing a schema may be difficult or time consuming in case of large databases.

What is Instance?

An instance of [DBMS](#) refers to real data in a database coming at some particular point in time. Instance on the other hand refers to the content in the database in as much as it refers to the structure defined under a particular schema at a given point.

Example

Let's say a table teacher in our database whose name is School, suppose the table has 50 records so the instance of the database has 50 records for now and tomorrow we are going to add another fifty records so tomorrow the instance has a total of 100 records. This is called an instance.

Advantages of Instance

- **Real-Time Representation:** It is a return of the data in the database at a certain point in time as may be required for analysis or for performing operations.
- **Flexibility:** While a schema remains fixed in time, instances can be quite volatile, as data is written, updated, or deleted.

Disadvantages of Instance

- **Volatility:** Those occurrences may be dynamic in a way they are different over time and this may make it a challenge to keep track without the necessary intervention.
- **Data Integrity Issues:** If not well regulated, it is evident that the data in an instance could become very inconsistent and at times even incorrect.

Interfaces in DBMS

A database management system (DBMS) interface is a user interface that allows for the ability to input queries to a database without using the query language itself. User-friendly interfaces provided by [DBMS](#) may include the following:

Menu-Based Interfaces

These interfaces present the user with lists of options (called menus) that lead the user through the formation of a request. The basic advantage of using menus is that they remove the tension of remembering specific commands and syntax of any query language. The query is basically composed step by step by collecting or picking options from a menu that is shown by the system. Pull-down menus are a very popular technique in **Web-based interfaces**. They are also often used in browsing interfaces which allow a user to look through the contents of a database in an exploratory and unstructured manner.

Forms-Based Interfaces

A forms-based interface displays a form to each user. Users can fill out all of the form entries to insert new data, or they can fill out only certain entries, in which case the DBMS will redeem the same type of data for other remaining entries. These types of forms are usually designed or created and programmed for users that have no expertise in operating systems.

Example: [SQL Forms](#) is a form-based language that specifies queries using a form designed in conjunction with the relational database schema.

Graphical User Interface

A GUI typically displays a schema to the user in diagrammatic form. The user then can specify a query by manipulating the diagram. In many cases, GUI utilise both menus and forms. Most GUI use a pointing device such as a mouse, to pick a certain part of the displayed schema diagram.

Natural Language Interfaces

These interfaces accept requests written in English or some other language and attempt to understand them. A Natural language interface has its own schema, which is similar to the database conceptual schema as well as a dictionary of important words.

Speech Input and Output Interfaces

There is limited use of speech be it for a query or an answer to a question or being a result of a request it is becoming commonplace. Applications with limited vocabulary such as inquiries for telephone directory, flight arrival/departure, and bank account information are allowed speech for input and output to enable ordinary folks to access this information.

The Speech input is detected using predefined words and used to set up the parameters that are supplied to the queries. For output, a similar conversion from text or numbers into speech takes place.

Interface for Parametric Users

Interfaces for Parametric Users contain some commands that can be handled with a minimum of keystrokes. It is generally used in bank transactions for transferring money. These operations are performed repeatedly.

Interfaces for Database Administrators (DBA)

Most database system contains privileged commands that can be used only by the DBA's staff. These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of databases.

DBMS and data characteristics:

- **Data Organization:**

A DBMS structures data into tables, rows, and columns, allowing for logical relationships between different data points.

- **Data Integrity:**

By enforcing data constraints, a DBMS ensures the accuracy and consistency of data within the database.

- **Data Independence:**

The separation of data structure from application programs, allowing changes to the database schema without affecting applications that access it.

- **Multi-user Access:**

A DBMS enables multiple users to simultaneously access and modify data in the database, with appropriate access controls.

- **ACID Properties:**

To guarantee reliable data transactions, a DBMS adheres to ACID principles:

- **Atomicity:** An entire transaction is completed or none of it is.
- **Consistency:** Data must remain in a valid state before and after a transaction.
- **Isolation:** Concurrent transactions are isolated from each other.
- **Durability:** Committed changes to the database are permanently stored.

Data Models in DBMS

A Data Model in Database Management System (DBMS) is the concept of tools that are developed to summarize the description of the database. Data Models provide us with a transparent picture of data which helps us in creating an actual database. It shows us from the design of the data to its proper implementation of data.

Types of Relational Models

1. Conceptual Data Model
2. Representational Data Model
3. Physical Data Model

It is basically classified into 3 types:-

1. Conceptual Data Model

The conceptual data model describes the database at a very high level and is useful to understand the needs or requirements of the database. It is this model, that is used in the requirement-gathering process i.e. before the Database Designers start making a particular database. One such popular model is the [entity/relationship model \(ER model\)](#). The E/R model specializes in entities, relationships, and even attributes that are used by database designers. In terms of this concept, a discussion can be made even with non-computer science(non-technical) users and stakeholders, and their requirements can be understood.

Entity-Relationship Model(ER Model): It is a high-level data model which is used to define the data and the relationships between them. It is basically a conceptual design of any database which is easy to design the view of data.

Components of ER Model:

1. **Entity:** An entity is referred to as a real-world object. It can be a name, place, object, class, etc. These are represented by a rectangle in an ER Diagram.
2. **Attributes:** An attribute can be defined as the description of the entity. These are represented by Ellipse in an ER Diagram. It can be Age, Roll Number, or Marks for a Student.
3. **Relationship:** Relationships are used to define relations among different entities. Diamonds and Rhombus are used to show Relationships.

Characteristics of a conceptual data model

- Offers Organization-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the “real world.”

Conceptual data models known as Domain models create a common vocabulary for all stakeholders by establishing basic concepts and scope

2. Representational Data Model

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database. A popular representational model is a [Relational model](#). The relational Model consists of [Relational Algebra](#) and [Relational Calculus](#). In the Relational Model, we basically use tables to represent our data and the relationships between them. It is a theoretical concept whose practical implementation is done in Physical Data Model.

The advantage of using a Representational data model is to provide a foundation to form the base for the Physical model.

Characteristics of Representational Data Model

- Represents the logical structure of the database.
- Relational models like Relational Algebra and Relational Calculus are commonly used.
- Uses tables to represent data and relationships.
- Provides a foundation for building the physical data model.

3. Physical Data Model

The physical Data Model is used to practically implement Relational Data Model. Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes. This is stored in the form of files, records, and certain other data structures. It has all the information on the format in which the files are present and the structure of the databases, the presence of external data structures, and their relation to each other. Here, we basically save tables in memory so they can be accessed efficiently. In order to come up with

a good physical model, we have to work on the relational model in a better way. [Structured Query Language \(SQL\)](#) is used to practically implement Relational Algebra.

This Data Model describes HOW the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.

Characteristics of a physical data model:

- The physical data model describes data need for a single project or application though it maybe integrated with other physical data models based on project scope.
- Data Model contains relationships between tables that which addresses cardinality and nullability of the relationships.
- Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
- Columns should have exact datatypes, lengths assigned and default values.
- Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined

Unit II

Relational Model and NoSQL

Relational Model

The Relational Model represents data and their relationships through a collection of tables. Each table also known as a relation consists of rows and columns. Every column has a unique name and corresponds to a specific attribute, while each row contains a set of related data values representing a real-world entity or relationship. This model is part of the record-based models which structure data in fixed-format records each belonging to a particular type with a defined set of attributes.

E.F. Codd introduced the Relational Model to organize data as relations or tables. After creating the conceptual design of a database using an [ER diagram](#), this design must be transformed into a relational model which can then be implemented using relational database systems like Oracle SQL or MySQL.

Important Terminologies

- **Attribute:** Attributes are the properties that define an entity. e.g. ROLL_NO, NAME, ADDRESS.
- **Relation Schema:** A relation schema defines the structure of the relation and represents the name of the relation with its attributes. e.g. STUDENT (ROLL_NO, NAME, ADDRESS, PHONE, and AGE) is the relation schema for STUDENT. If a schema has more than 1 relation it is called Relational Schema.
- **Tuple:** Each row in the relation is known as a tuple. The above relation contains 4 tuples one of which is shown as:

1	RAM	DELHI	9455123451	18
---	-----	-------	------------	----

- **Relation Instance:** The set of tuples of a relation at a particular instance of time is called a relation instance. It can change whenever there is an insertion, deletion or update in the database.
- **Degree:** The number of attributes in the relation is known as the degree of the relation. The STUDENT relation defined above has degree 5.
- **Cardinality:** The number of tuples in a relation is known as [cardinality](#). The STUDENT relation defined above has cardinality 4.
- **Column:** The column represents the set of values for a particular attribute. The column ROLL_NO is extracted from the relation STUDENT.
- **NULL Values:** The value which is not known or unavailable is called a NULL value. It is represented by NULL. e.g. PHONE of STUDENT having ROLL_NO 4 is NULL.
- **Relation Key:** These are basically the keys that are used to identify the rows uniquely or also help in identifying tables. These are of the following types:
 - [Primary Key](#)
 - [Candidate Key](#)
 - [Super Key](#)
 - [Foreign Key](#)
 - [Alternate Key](#)
 - [Composite Key](#)

Relational Model Notation

- Relation schema R of degree n is denoted by $R(A_1, A_2, \dots, A_n)$.
- Uppercase letters Q, R, S denote relation names.
- Lowercase letters q, r, s denote relation states.
- Letters t, u, v denote tuples.
- In general, the name of a relation schema such as **STUDENT** also indicates the current set of tuples in that relation.
- An attribute A can be qualified with the relation name R to which it belongs by using the dot notation $R.A$ for example, **STUDENT.Name** or **STUDENT.Age**.
- An n -tuple t in a relation $r(R)$ is represented as $t = \langle v_1, v_2, \dots, v_n \rangle$ where v_i is the value corresponding to the attribute A_i . The value v_i for attribute A_i in tuple t can be accessed using $t[A_i]$ or $t.A_i$.

Characteristics of the Relational Model

- **Data Representation:**
Data is organized in tables (relations), with rows (tuples) representing records and columns (attributes) representing data fields.
- **Atomic Values:**
Each attribute in a table contains atomic values, meaning no multi-valued or nested data is allowed in a single cell.
- **Unique Keys:**
Every table has a primary key to uniquely identify each record, ensuring no duplicate rows.
- **Attribute Domain:**
Each attribute has a defined domain, specifying the valid data types and constraints for the values it can hold.
- **Tuples as Rows:**
Rows in a table, called tuples, represent individual records or instances of real-world entities or relationships.
- **Relation Schema:**
A table's structure is defined by its schema, which specifies the table name, attributes, and their domains.
- **Data Independence:**
The model ensures logical and physical data independence, allowing changes in the database schema without affecting the application layer.
- **Integrity Constraints:**
The model enforces rules like:
 - **Domain constraints:** Attribute values must match the specified domain.
 - **Entity integrity:** No primary key can have NULL values.
 - **Referential integrity:** Foreign keys must match primary keys in the referenced table or be NULL.
- **Relational Operations:**
Supports operations like **selection, projection, join, union, and intersection**, enabling powerful data retrieval and manipulation.
- **Data Consistency:**
Ensures data consistency through constraints, reducing redundancy and anomalies.
- **Set-Based Representation:**
Tables in the relational model are treated as sets, and operations follow mathematical set theory principles.

Constraints in Relational Model

While designing the Relational Model, we define some conditions which must hold for data present in the database are called Constraints. These constraints are checked before

performing any operation (insertion, deletion, and updation) in the database. If there is a violation of any of the constraints, the operation will fail.

Domain Constraints

Domain Constraints ensure that the value of each attribute A in a tuple must be an atomic value derived from its specified domain, $\text{dom}(A)$. Domains are defined by the data types associated with the attributes. Common data types include:

- **Numeric types:** Integers (e.g., short integer, integer, long integer) and real numbers (e.g., float, double-precision float).
- **Character types:** Fixed-length and variable-length strings.
- **Boolean values.**
- **Specialized types:** Date, time, timestamp, monetary values, and others.

Key Integrity

Every relation in the database should have at least one set of attributes that defines a tuple uniquely. Those set of attributes is called keys. e.g.; ROLL_NO in STUDENT is key. No two students can have the same roll number. So a key has two properties:

- It should be unique for all tuples.
- It can't have NULL values.

Relation Schema in DBMS

Relation schema defines the design and structure of the relation or table in the database. It is the way of representation of relation states in such a way that every relation database state fulfills the integrity constraints set (Like Primary key, Foreign Key, Not null, Unique constraints) on a relational schema.

It consists of the relation name, set of attributes/field names/column names. every attribute would have an associated domain. In this article, we will discuss relational schema in detail along with proper examples.

Components of a Relation Schema

- **Relation Name:** Name of the table that is stored in the database. It should be unique and related to the data that is stored in the table. For example- The name of the table can be Employee store the data of the employee.
- **Attributes Name:** [Attributes](#) specify the name of each column within the table. Each attribute has a specific data type.
- **Domains:** The set of possible values for each attribute. It specifies the type of data that can be stored in each column or attribute, such as integer, string, or date.
- **Primary Key:** The [primary key](#) is the key that uniquely identifies each tuple. It should be unique and not be null.
- **Foreign Key:** The [foreign key](#) is the key that is used to connect two tables. It refers to the primary key of another table.
- **Constraints:** Rules that ensure the integrity and validity of the data. Common constraints include NOT NULL, UNIQUE, CHECK, and DEFAULT.

The relation schema defines the structure of data stored in a database, specifying attributes and relationships.

Constraints on Relational Database Mode

In modeling the design of the relational database we can put some restrictions like what values are allowed to be inserted in the relation, and what kind of modifications and deletions are allowed in the relation. These are the restrictions we impose on the relational database.

In models like Entity-Relationship models, we did not have such features. Database Constraints can be categorized into 3 main categories:

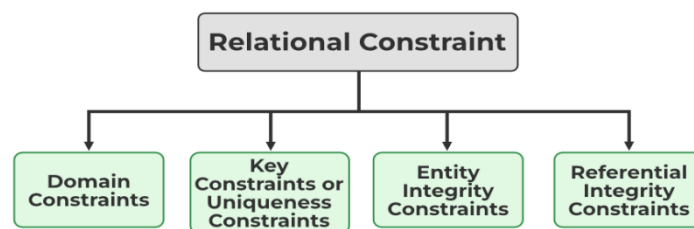
1. Constraints that are applied in the data model are called **Implicit Constraints**.
2. Constraints that are directly applied in the schemas of the data model, by specifying them in the [DDL\(Data Definition Language\)](#). These are called **Schema-Based Constraints or Explicit Constraints**.
3. Constraints that cannot be directly applied in the schemas of the data model. We call these Application-based or **Semantic Constraints**.

So here we are going to deal with **Implicit constraints**.

Relational Constraints

These are the restrictions or sets of rules imposed on the database contents. It validates the quality of the database. It validates the various operations like data insertion, updation, and other processes that have to be performed without affecting the integrity of the data. It protects us against threats/damages to the database. Mainly Constraints on the relational database are of 4 types

- Domain constraints
- Key constraints or Uniqueness Constraints
- Entity Integrity constraints
- Referential integrity constraints



Types of Relational Constraints

Update Operations And Transactions

In a Database Management System (DBMS), "update operations" refer to the actions that modify existing data within a database, like changing values in a table row, while a "transaction" is a logical unit of work that encompasses a set of related update operations, ensuring that all changes within the transaction are either fully committed to the database or completely rolled back if any errors occur, maintaining data consistency and integrity; essentially, a transaction guarantees that a series of updates happen as a single, atomic unit.

Key points about update operations and transactions:

- **Update operations:**
- **Types:** Include "insert" (adding new data), "delete" (removing data), and "update" (modifying existing data).
- **Functionality:** These operations are performed using SQL statements like INSERT, DELETE, and UPDATE to modify data within a table.
- **Example:** Updating a customer's address in a customer database.
- **Transactions:**

- **Concept:** A transaction is a group of related database operations that must be completed successfully as a single unit to maintain data integrity.
- **ACID properties:** Transactions are typically governed by the ACID properties:
- **Atomicity:** All operations within a transaction are treated as a single unit, either all succeed or all fail.
- **Consistency:** A transaction brings the database from one valid state to another.
- **Isolation:** Concurrent transactions appear to execute independently, preventing interference with each other.
- **Durability:** Once a transaction is committed, the changes are permanently stored in the database.

What are the Constraints of DBMS

In DBMS, constraints refer to limitations placed on data or data processes. This indicates that only a particular type of data may be entered into the database or that only a particular sort of operation can be performed on the data inside.

Constraints thereby guarantee data accuracy in a database management system (DBMS).

The following can be guaranteed via constraints

Data Accuracy – Data accuracy is guaranteed by constraints, which make sure that only true data is entered into a database. For example, a limitation may stop a user from entering a negative value into a field that only accepts positive numbers.

Data Consistency – The consistency of data in a database can be upheld by using constraints. These constraints are able to ensure that the primary key value in one table is followed by the foreign key value in another table.

Data integrity – The accuracy and completeness of the data in a database are ensured by constraints. For example, a constraint can stop a user from putting a null value into a field that requires one.

Types of Constraints in DBMS

- Domain Constraints
- Key Constraints
- Entity Integrity Constraints
- Referential Integrity Constraints
- Tuple Uniqueness Constraints

Domain Constraints

In a database table, domain constraints are guidelines that specify the acceptable values for a certain property or field. These restrictions guarantee data consistency and aid in preventing the entry of inaccurate or inconsistent data into the database. The following are some instances of domain restrictions in a DBMS –

- **Data type constraints** – These limitations define the kinds of data that can be kept in a column. A column created as VARCHAR can take string values, but a column specified as INTEGER can only accept integer values.
- **Length Constraints** – These limitations define the largest amount of data that may be put in a column. For instance, a column with the definition VARCHAR(10) may only take strings that are up to 10 characters long.
- **Range constraints** – The allowed range of values for a column is specified by range restrictions. A column designated as DECIMAL(5,2), for example, may only take decimal values up to 5 digits long, including 2 decimal places.
- **Nullability constraints** – Constraints on a column's capacity to accept NULL values are known as nullability constraints. For instance, a column that has the NOT NULL definition cannot take NULL values.
- **Unique constraints** – Constraints that require the presence of unique values in a column or group of columns are known as unique constraints. For instance, duplicate values are not allowed in a column with the UNIQUE definition.
- **Check constraints** – Constraints for checking data: These constraints outline a requirement that must hold for any data placed into the column. For instance, a column with the definition CHECK (age > 0) can only accept ages that are greater than zero.
- **Default constraints** – Constraints by default: Default constraints automatically assign a value to a column in case no value is provided. For example, a column with a DEFAULT value of 0 will have 0 as its value if no other value is specified.

Key Constraints

Key constraints are regulations that a DBMS uses to ensure data accuracy and consistency in a database. They define how the values in a table's one or more columns are related to the values in other tables, making sure that the data remains correct.

In DBMS, there are several key constraint kinds, including –

- **Primary Key Constraint** – A **primary key** constraint is an individual identifier for each record in a database. It guarantees that each database entry contains a single, distinct value—or a pair of values—that cannot be null—as its method of identification.
- **Foreign Key Constraint** – Reference to the primary key in another table is a **foreign key constraint**. It ensures that the values of a column or set of columns in one table correspond to the primary key column(s) in another table.
- **Unique Constraint** – In a database, a **unique constraint** ensures that no two values inside a column or collection of columns are the same.

Entity Integrity Constraints

A database management system uses entity integrity constraints (EICs) to enforce rules that guarantee a table's primary key is unique and not null. The consistency and integrity of the data in a database are maintained by EICs, which are created to stop the formation of duplicate or incomplete entries.

Each item in a table in a relational database is uniquely identified by one or more fields known as the primary key. EICs make a guarantee that every row's primary key value is distinct and not null. Take the "Employees" table, for instance, which has the columns "EmployeeID" and "Name." The table's primary key is the EmployeeID column. An EIC on this table would make sure that each row's unique EmployeeID value is there and that it is not null.

If you try to insert an entry with a duplicate or null EmployeeID, the database management system will reject the insertion and produce an error. This guarantees that the information in the table is correct and consistent.

EICs are a crucial component of database architecture and assist guarantee the accuracy and dependability of the data contained in a database.

Referential Integrity Constraints

A database management system will apply referential integrity constraints (RICs) in order to preserve the consistency and integrity of connections between tables. By preventing links between entries that don't exist from being created or by removing records that have related records in other tables, RICs guarantee that the data in a database is always consistent.

By the use of foreign keys, linkages between tables are created in relational databases. A column or collection of columns in one table that is used as a foreign key to access the primary key of another table. RICs make sure there are no referential errors and that these relationships are legitimate.

Consider the "Orders" and "Customers" tables as an illustration. The primary key column in the "Customers" database corresponds to the foreign key field "CustomerID" in the "Orders" dataset. A RIC on this connection requires that each value in the "CustomerID" column of the "Orders" database exist in the "Customers" table's primary key column.

If an attempt was made to insert a record into the "Orders" table with a non-existent "CustomerID" value, the database management system would reject the insertion and notify the user of an error.

Similar to this, the database management system would either prohibit the deletion or cascade the deletion in order to ensure referential integrity if a record in the "Customers" table was removed and linked entries in the "Orders" table.

In general, RICs are a crucial component of database architecture and assist guarantee that the information contained in a database is correct and consistent throughout time.

Tuple Uniqueness Constraints

A database management system uses constraints called tuple uniqueness constraints (TUCs) to make sure that every entry or tuple in a table is distinct. TUCs impose uniqueness on the whole row or tuple, in contrast to Entity Integrity Constraints (EICs), which only enforce uniqueness on certain columns or groups of columns.

TUCs, then, make sure that no two rows in a table have the same values for every column. Even if the individual column values are not unique, this can be helpful in cases when it is vital to avoid the production of duplicate entries.

Consider the "Sales" table, for instance, which has the columns "TransactionID," "Date," "CustomerID," and "Amount." Even if individual column values could be duplicated, a TUC on this table would make sure that no two rows have the same values in all four columns.

Data Types?

An [SQL](#) developer must know what data type will be stored inside each column while creating a table. The data type guideline for SQL is to understand what type of data is expected inside each column and it also identifies how SQL will interact with the stored data.

- Numeric Datatypes
- Character and String Data Types
- Date and Time Data Types
- Binary Data Types
- Boolean Data Types
- Special Data Types

Numeric Data Types

AI-DS Department

Numeric data types are fundamental to database design and are used to store numbers, whether they are integers, decimals, or floating-point numbers. These data types allow for mathematical operations like addition, subtraction, multiplication, and division, which makes them essential for managing financial, scientific, and analytical data.

Exact Numeric Datatype

Exact numeric types are used when precise numeric values are needed, such as for financial data, quantities, and counts.

Data Type	From	To
BigInt	-263 (-9,223,372,036,854,775,808)	263 -1 (9,223,372,036,854,775,807)
Int	-231 (-2,147,483,648)	231-1 (2,147,483,647)
smallint	-215 (-32,768)	215-1 (32,767)
tinyint	0	28-1 (255)
bit	0	1
decimal	-1038+1	1038-1
numeric	-1038+1	1038-1
money	-922,337,203,685,477.5808	922,337,203,685,477.5807
smallmoney	-214,748.3648	214,748.3647

Approximate Numeric Datatype

These types are used to store approximate values, such as scientific measurements or large ranges of data that don't need exact precision.

Data Type	From	To
Float	-1.79E+308	1.79E+308
Real	-3.40E+38	3.40E+38

String Data Types

Character data types are used to store text or character-based data.

Character String Datatype

The subtypes are given in below table

Data Type	Description
-----------	-------------

Data Type	Description
char	The maximum length of 8000 characters.(Fixed-Length non-Unicode Characters)
varchar	The maximum length of 8000 characters.(Variable-Length non-Unicode Characters)
varchar(max)	The maximum length of 231 characters(SQL Server 2005 only).(Variable Length non-Unicode data)
text	The maximum length of 2,127,483,647 characters(Variable Length non-Unicode data)

Unicode Character String Datatype

Unicode data types are used to store characters from any language, supporting a wider variety of characters. These are given in below table.

Data Type	Description
nchar	The maximum length of 4000 characters(Fixed-Length Unicode Characters)
Nvarchar	The maximum length of 4000 characters.(Variable-Length Unicode Characters)
nvarchar(max)	The maximum length of 231 characters(SQL Server 2005 only).(Variable Length Unicode data)

Date and Time Data Type in SQL

SQL provides several data types for storing date and time information. These are given in the below table.

Data Type	Description
DATE	A data type is used to store the data of date in a record
TIME	A data type is used to store the data of time in a record
DATETIME	A data type is used to store both the data,date, and time in the record.

Binary Data Types in SQL

Binary data types are used to store binary data such as images, videos, or other file types.

Data Type	Description	Max Length
BINARY	Fixed-length binary data.	8000 bytes
VARBINARY	Variable-length binary data.	8000 bytes
IMAGE	Stores binary data as images.	2,147,483,647 bytes

Boolean Data Type in SQL

Boolean data types are used to store logical values, typically TRUE or FALSE.

Data Type	Description
BOOLEAN	Stores a logical value (TRUE/FALSE).

Other Special Data Types

XML Datatype

XML data type allows storage of XML documents and fragments in a SQL Server database

Data Type	Description
XML Datatype	A Datatype used to store data in the format of XML datatype

Spatial Datatype

A datatype is used for storing planar spatial data, such as points, lines, and polygons, in a database table.

Data Type	Description
Geometry	A datatype is used for storing planar spatial data, such as points, lines, and polygons, in a database table.

Array Datatype

SQL Server does not have a built-in array datatype. However, it is possible to simulate arrays using tables or XML data types.

Introduction of ER Model

•

We typically follow the below steps for **designing** a database for an application.

- Gather the requirements (functional and data) by asking questions to the database users.
- Do a logical or conceptual design of the database. This is where **ER model** plays a role. It is the most used graphical representation of the conceptual design of a database.
- Physical Database Design (Like indexing) and external design (like views)

The Entity Relationship Model is a model for identifying entities (like student, car or company) to be represented in the database and representation of how those entities are related. The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.

Why Use ER Diagrams In DBMS?

- ER diagrams represent the E-R model in a database, making them easy to convert into relations (tables).
- ER diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
- ER diagrams require no technical knowledge of the underlying DBMS used.
- It gives a standard solution for visualizing the data logically.

Symbols Used in ER Model

ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

- **Rectangles:** Rectangles represent Entities in the ER Model.
- **Ellipses:** Ellipses represent Attributes in the ER Model.
- **Diamond:** Diamonds represent Relationships among Entities.
- **Lines:** Lines represent attributes to entities and entity sets with other relationship types.
- **Double Ellipse:** Double Ellipses represent Multi-Valued Attributes.
- **Double Rectangle:** Double Rectangle represents a Weak Entity.

Components of ER Diagram

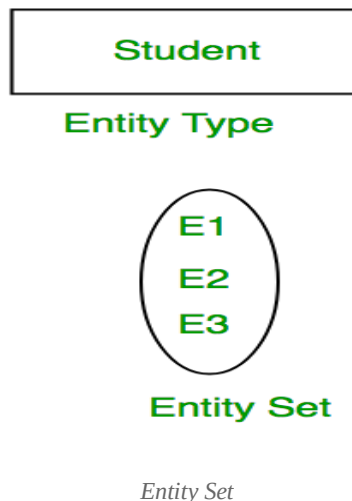
ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.

What is Entity?

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

What is Entity Set?

An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



We can represent the entity set in ER Diagram but can't represent entity in ER Diagram because entity is row and column in the relation and ER Diagram is graphical representation of data.

Types of Entity

There are two types of entity:

1. Strong Entity

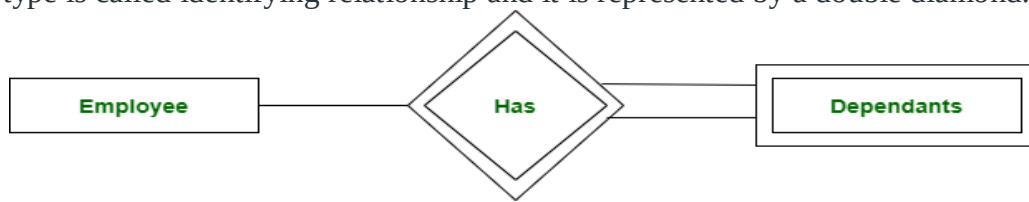
A [Strong Entity](#) is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can't be defined. These are called [Weak Entity types](#).

For Example, A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents can't exist without the employee. So Dependent will be a **Weak Entity Type** and Employee will be Identifying Entity type for Dependent, which means it is **Strong Entity Type**.

A weak entity type is represented by a Double Rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



Strong Entity and Weak Entity

What is Attributes?

[Attributes](#) are the properties that define the entity type. For example, Roll_No, Name, DOB, Age, Address, and Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.



Attribute

Types of Attributes

1. Key Attribute

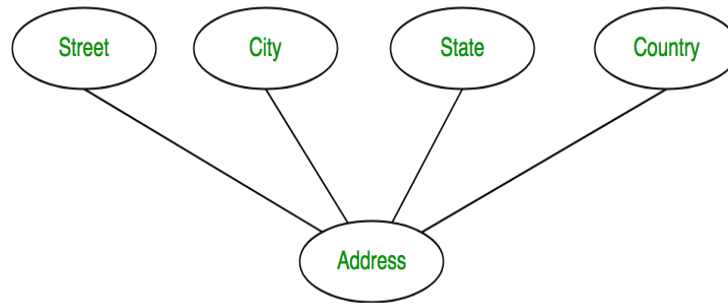
The attribute which **uniquely identifies each entity** in the entity set is called the key attribute. For example, Roll_No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines.



Key Attribute

2. Composite Attribute

An attribute **composed of many other attributes** is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. In ER diagram, the composite attribute is represented by an oval comprising of ovals.



Composite Attribute

3. Multivalued Attribute

An attribute consisting of more than one value for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.



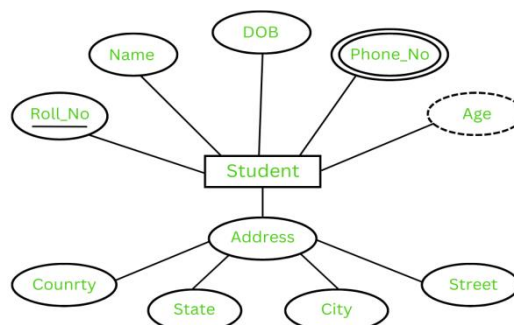
Multivalued Attribute

4. Derived Attribute

An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.



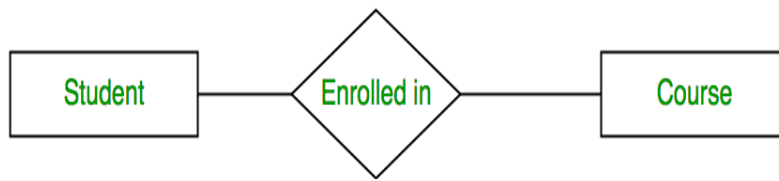
Derived Attribute



Entity and Attributes

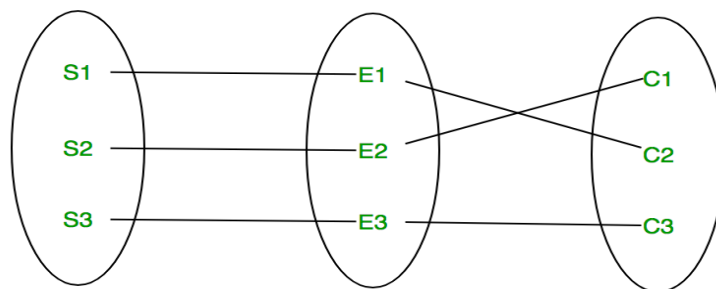
Relationship Type and Relationship Set

A Relationship Type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course. In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



Entity-Relationship Set

A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.

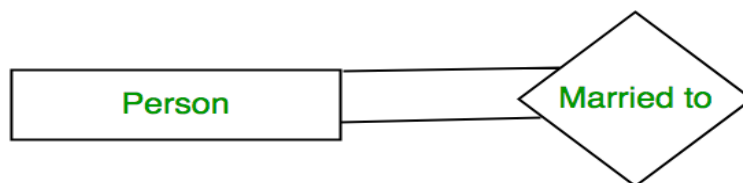


Relationship Set

Degree of a Relationship Set

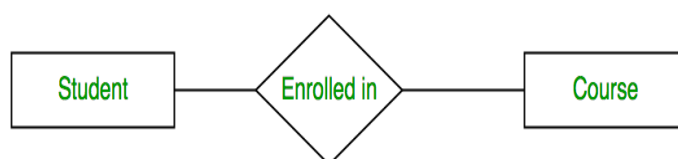
The number of different entity sets participating in a relationship set is called the [degree of a relationship set](#).

1. Unary Relationship: When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



Unary Relationship

2. Binary Relationship: When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.



Binary Relationship

3. Ternary Relationship: When there are three entity sets participating in a relationship, the relationship is called a ternary relationship.

4. N-ary Relationship: When there are n entities set participating in a relationship, the relationship is called an n-ary relationship.

What is Cardinality?

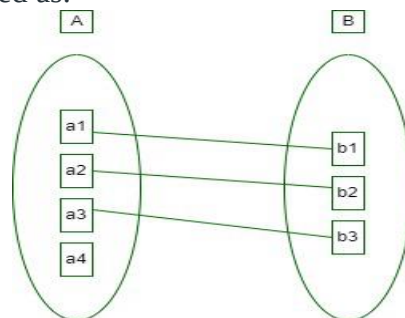
The number of times an entity of an entity set participates in a relationship set is known as [cardinality](#). Cardinality can be of different types:

1. One-to-One: When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.
the total number of tables that can be used in this is 2.



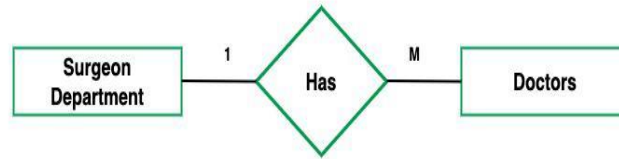
one to one cardinality

Using Sets, it can be represented as:

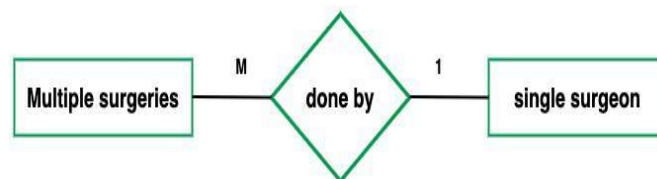


Set Representation of One-to-One

2. One-to-Many: In one-to-many mapping as well where each entity can be related to more than one entity and the total number of tables that can be used in this is 2. Let us assume that one surgeon department can accommodate many doctors. So the Cardinality will be 1 to M. It means one department has many Doctors.
total number of tables that can be used is 3.

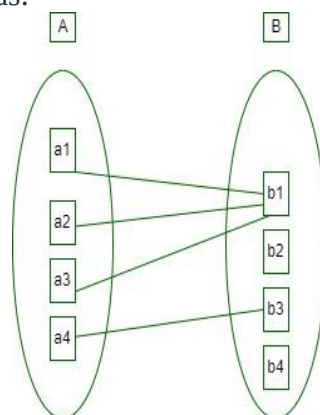


3. Many-to-One: When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1 . It means that for one course there can be n students but for one student, there will be only one course. The total number of tables that can be used in this is 3.



many to one cardinality

Using Sets, it can be represented as:

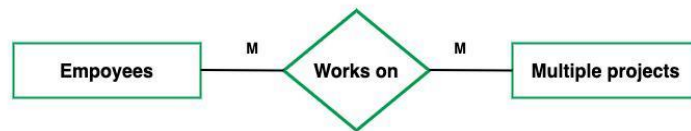


Set Representation of Many-to-One

In this case, each student is taking only 1 course but 1 course has been taken by many students.

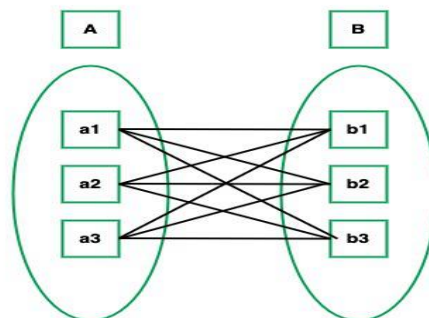
4. Many-to-Many: When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

the total number of tables that can be used in this is 3.



many to many cardinality

Using Sets, it can be represented as:



Many-to-Many Set Representation

In this example, student S1 is enrolled in C1 and C3 and Course C3 is enrolled by S1, S3, and S4. So it is many-to-many relationships.

Introduction to NoSQL

NoSQL is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data. Unlike traditional relational databases that use tables with pre-defined schemas to store data, NoSQL databases use flexible data models that can adapt to changes in data structures and are capable of scaling horizontally to handle growing amounts of data.

The term NoSQL originally referred to “non-SQL” or “non-relational” databases, but the term has since evolved to mean “not only SQL,” as NoSQL databases have expanded to include a wide range of different database architectures and data models.

NoSQL databases are generally classified into four main categories:

1. **Document databases:** These databases store data as semi-structured documents, such as JSON or XML, and can be queried using document-oriented query languages.
2. **Key-value stores:** These databases store data as key-value pairs, and are optimized for simple and fast read/write operations.
3. **Column-family stores:** These databases store data as column families, which are sets of columns that are treated as a single entity. They are optimized for fast and efficient querying of large amounts of data.
4. **Graph databases:** These databases store data as nodes and edges, and are designed to handle complex relationships between data.

NoSQL databases are often used in applications where there is a high volume of data that needs to be processed and analyzed in real-time, such as social media analytics, e-commerce, and gaming. They can also be used for other applications, such as content management systems, document management, and customer relationship management.

However, NoSQL databases may not be suitable for all applications, as they may not provide the same level of data consistency and transactional guarantees as traditional relational databases. It is important to carefully evaluate the specific needs of an application when choosing a database management system.

NoSQL originally referring to non SQL or non relational is a database that provides a mechanism for storage and retrieval of data. This data is modeled in means other than the tabular relations used in relational databases. Such databases came into existence in the late 1960s, but did not obtain the NoSQL moniker until a surge of popularity in the early twenty-first century. NoSQL databases are used in real-time web applications and big data and their use are increasing over time.

- NoSQL systems are also sometimes called Not only SQL to emphasize the fact that they may support SQL-like query languages. A NoSQL database includes simplicity of design, simpler horizontal scaling to clusters of machines, **has** and finer control over availability. The data structures used by NoSQL databases are different from those used by default in relational databases which makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it should solve.
- NoSQL databases, also known as “not only SQL” databases, are a new type of database management system that has, gained popularity in recent years. Unlike traditional relational databases, NoSQL databases are designed to handle large amounts of unstructured or semi-structured data, and they can accommodate dynamic changes to the data model. This makes NoSQL databases a good fit for modern web applications, real-time analytics, and big data processing.
- Data structures used by NoSQL databases are sometimes also viewed as more flexible than relational database tables. Many NoSQL stores compromise consistency in favor of availability, speed, and partition tolerance. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, lack of standardized interfaces, and huge previous investments in existing relational databases.
- Most NoSQL stores lack true ACID (Atomicity, Consistency, Isolation, Durability) transactions but a few databases, such as MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (though technically a NewSQL database), Symas LMDB, and OrientDB have made them central to their designs.
- Most NoSQL databases offer a concept of eventual consistency in which database changes are propagated to all nodes so queries for data might not return updated data immediately or might result in reading data that is not accurate which is a problem known as stale reads. Also, **has** some NoSQL systems may exhibit lost writes and other forms of

data loss. Some NoSQL systems provide concepts such as write-ahead logging to avoid data loss.

- One simple example of a NoSQL database is a document database. In a document database, data is stored in documents rather than tables. Each document can contain a different set of fields, making it easy to accommodate changing data requirements
- For example, “Take, for instance, a database that holds data regarding employees.”. In a relational database, this information might be stored in tables, with one table for employee information and another table for department information. In a document database, each employee would be stored as a separate document, with all of their information contained within the document.
- NoSQL databases are a relatively new type of database management system that has gained popularity in recent years due to their scalability and flexibility. They are designed to handle large amounts of unstructured or semi-structured data and can handle dynamic changes to the data model. This makes NoSQL databases a good fit for modern web applications, real-time analytics, and big data processing.

Document Data Model:

A Document Data Model is a lot different than other data models because it stores data in JSON, BSON, or XML documents. In this data model, we can move documents under one document and apart from this, any particular elements can be indexed to run queries faster. Often documents are stored and retrieved in such a way that it becomes close to the data objects which are used in many applications which means very less translations are required to use data in applications. JSON is a native language that is often used to store and query data too.

So in the document data model, each document has a key-value pair below is an example for the same.

```
{  
  "Name" : "Yashodhra",  
  "Address" : "Near Patel Nagar",  
  "Email" : "yahoo123@yahoo.com",  
  "Contact" : "12345"  
}
```

Working of Document Data Model:

This is a data model which works as a semi-structured data model in which the records and data associated with them are stored in a single document which means this data model is not completely unstructured. The main thing is that data here is stored in a document.

Violation of constraints in relational database

Here, we will learn about the violations that can occur on a database as a result of any changes made in the relation.

There are mainly three operations that have the ability to change the state of relations, these modifications are given below:

1. **Insert –**
To insert new tuples in a relation in the database.
2. **Delete –**
To delete some of the existing relation on the database.
3. **Update (Modify) –**
To make changes in the value of some existing tuples.

Whenever we apply the above modification to the relation in the database, the constraints on the relational database should not get violated.

Insert operation:

On inserting the tuples in the relation, it may cause violation of the constraints in the following way:

1. Domain constraint :

Domain constraint gets violated only when a given value to the attribute does not appear in the corresponding domain or in case it is not of the appropriate datatype.

Example:

Assume that the domain constraint says that all the values you insert in the relation should be greater than 10, and in case you insert a value less than 10 will cause you violation of the domain constraint, so gets rejected.

2. Entity Integrity constraint :

On inserting NULL values to any part of the primary key of a new tuple in the relation can cause violation of the Entity integrity constraint.

Example:

Insert (NULL, 'Bikash', 'M', 'Jaipur', '123456') into EMP

The above insertion violates the entity integrity constraint since there is NULL for the primary key EID, it is not allowed, so it gets rejected.

3. Key Constraints :

On inserting a value in the new tuple of a relation which is already existing in another tuple of the same relation, can cause violation of Key Constraints.

Example:

Insert ('1200', 'Arjun', '9976657777', 'Mumbai') into EMPLOYEE

This insertion violates the key constraint if EID=1200 is already present in some tuple in the same relation, so it gets rejected.

Referential integrity :

On inserting a value in the foreign key of relation 1, for which there is no corresponding value in the Primary key which is referred to in relation 2, in such case Referential integrity is violated.

Example:

When we try to insert a value say 1200 in EID (foreign key) of table 1, for which there is no corresponding EID (primary key) of table 2, then it causes violation, so gets rejected. Solution that is possible to correct such violation is if any insertion violates any of the constraints, then the default action is to reject such operation.

Deletion operation:

On deleting the tuples in the relation, it may cause only violation of Referential integrity constraints.

Referential Integrity Constraints :

It causes violation only if the tuple in relation 1 is deleted which is referenced by foreign

key from other tuples of table 2 in the database, if such deletion takes place then the values in the tuple of the foreign key in table 2 will become empty, which will eventually violate Referential Integrity constraint.

Solutions that are possible to correct the violation to the referential integrity due to deletion are listed below:

1. **Restrict** –
Here we reject the deletion.
2. **Cascade** –
Here if a record in the parent table(referencing relation) is deleted, then the corresponding records in the child table(referenced relation) will automatically be deleted.
3. **Set null or set default** –
Here we modify the referencing attribute values that cause violation and we either set NULL or change to another valid value

Unit III

Normalization

In database design, a "functional dependency" refers to a relationship between attributes in a table where the value of one attribute (or set of attributes) uniquely determines the value of another attribute, and it is a crucial concept used in the process of "normalization" to structure a database efficiently, minimizing data redundancy and ensuring data integrity by identifying and eliminating unnecessary dependencies between columns.

Key points about functional dependencies and normalization:

- **Definition:**

A functional dependency is represented as " $X \rightarrow Y$," where X is the determinant attribute (which determines the value) and Y is the dependent attribute (whose value is determined).

- **Importance in database design:**

By understanding functional dependencies, database designers can identify potential redundancy and anomalies (like insertion, update, deletion anomalies) in a table, allowing them to split the table into smaller, more normalized tables to improve data quality and consistency.

- **Normalization process:**

Normalization involves analyzing the functional dependencies in a table and progressively breaking it down into smaller tables based on different normal forms (1NF, 2NF, 3NF, BCNF) to eliminate unwanted dependencies.

Example:

Consider a table "Student" with columns "StudentID", "FirstName", "LastName", "Course", and "Grade".

- **Functional Dependency:**

"StudentID $\rightarrow$ FirstName, LastName, Course, Grade" - This means that the "StudentID" uniquely determines the values of all other attributes in the row.

- **Potential issue:**

If a student takes multiple courses, storing all course details in a single row can lead to redundancy.

- **Normalization solution:**

Create separate tables "Student" (with "StudentID", "FirstName", "LastName") and "Enrollment" (with "StudentID", "Course", "Grade") to eliminate redundancy and ensure data integrity.

Types of functional dependencies:

- **Trivial functional dependency:**

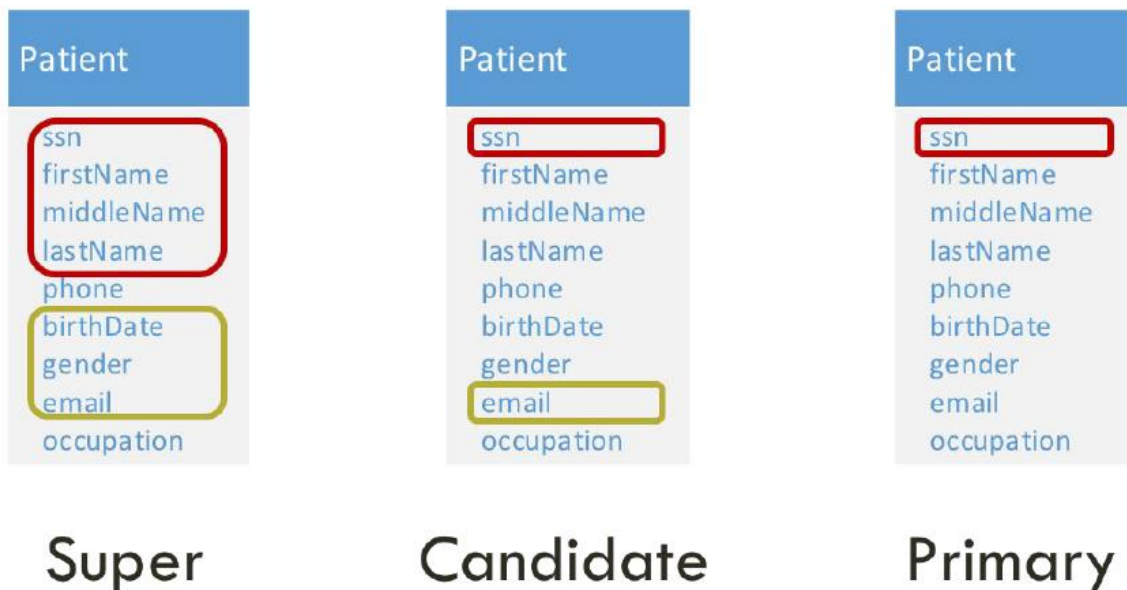
When an attribute is functionally dependent on itself (e.g., "StudentID $\rightarrow$ StudentID")

- **Partial functional dependency:**

When an attribute is dependent on only a part of a candidate key (e.g., "Course $\rightarrow$ Instructor" if "Course" is part of a composite key)

- **Transitive functional dependency:**

When the value of one attribute determines the value of another attribute through an intermediary attribute (e.g., "Department -> Instructor -> Course")



Rules of Inference

Rules of Inference: Rules of inference are logical tools used to derive conclusions from premises. They form the foundation of logical reasoning, allowing us to build arguments, prove theorems, and solve problems in mathematics, computer science, and philosophy. Understanding these rules is crucial for constructing valid arguments and ensuring that conclusions follow logically from given information.

Table of Content

- [What are Rules of Inference?](#)
- [Basic Rules of Inference](#)
 - [Modus Ponens \(Law of Detachment\)](#)
 - [Modus Tollens \(Law of Contrapositive\)](#)
 - [Hypothetical Syllogism](#)
 - [Disjunctive Syllogism](#)
 - [Conjunction](#)
 - [Simplification](#)
 - [Addition](#)
- [Rule of Inference Table](#)
- [Rules of Inference](#)
- [Resolution Principle](#)
- [GATE Questions – Rule of Inference](#)

What are Rules of Inference?

Rules of inference are essential tools in logic and mathematics, enabling the derivation of conclusions from premises. They form the backbone of logical reasoning, and proof techniques, and are extensively used in fields such as computer science, engineering, and mathematics.

Basic Rules of Inference

1. Modus Ponens (Law of Detachment)

If a conditional statement (“if-then” statement) is true, and its antecedent (the “if” part) is true, then its consequent (the “then” part) must also be true.

Form: If $p \rightarrow q$ and p , then q .

Example:

- Premise: If it rains, the ground will be wet.
- Premise: It is raining.
- Conclusion: The ground is wet.

2. Modus Tollens (Law of Contrapositive)

If a conditional statement is true, and its consequent is false, then its antecedent must also be false.

Form: If $p \rightarrow q$ and $\neg q$, then $\neg p$.

Example:

- Premise: If it rains, the ground will be wet.
- Premise: The ground is not wet.
- Conclusion: It is not raining.

3. Hypothetical Syllogism

If two conditional statements are true, where the consequent of the first is the antecedent of the second, then a third conditional statement combining the antecedent of the first and the consequent of the second is also true.

Form: If $p \rightarrow q$ and $q \rightarrow r$, then $p \rightarrow r$.

Example:

- Premise: If it rains, the ground will be wet.
- Premise: If the ground is wet, the plants will grow.
- Conclusion: If it rains, the plants will grow.

4. Disjunctive Syllogism

If a disjunction (an “or” statement) is true, and one of the disjuncts (the parts of the “or” statement) is false, then the other disjunct must be true.

Form: If $p \vee q$ and $\neg p$, then q .

Example:

- Premise: It is either raining or sunny.
- Premise: It is not raining.
- Conclusion: It is sunny.

5. Conjunction

If two statements are true, then their conjunction (an “and” statement) is also true.

Form: If p and q , then $p \wedge q$.

Example:

- Premise: It is raining.
- Premise: It is windy.
- Conclusion: It is raining and windy.

6. Simplification

If a conjunction (an “and” statement) is true, then each of its conjuncts is also true.

Form:

If $p \wedge q$, then p

If $p \wedge q$, then q

Example:

- Premise: It is raining and windy.
- Conclusion: It is raining.

7. Addition

If a statement is true, then the disjunction (an “or” statement) of that statement with any other statement is also true.

Form:

If p, then $p \vee q$

Example:

- Premise: It is raining.
- Conclusion: It is raining or sunny.

Rule of Inference Table

Rule of Inference	Description
Modus Ponens (MP)	If P implies Q, and P is true, then Q is true.
Modus Tollens (MT)	If P implies Q, and Q is false, then P is false.
Hypothetical Syllogism (HS)	If P implies Q and Q implies R, then P implies R.
Disjunctive Syllogism (DS)	If P or Q is true, and P is false, then Q is true.
Addition (Add)	If P is true, then P or Q is true.
Simplification (Simp)	If P and Q are true, then P is true
Conjunction (Conj)	If P and Q are true, then P and Q are true.

Rules of Inference

Simple arguments can be used as building blocks to construct more complicated valid arguments. Certain simple arguments that have been established as valid are very important in terms of their usage. These arguments are called Rules of Inference. The most commonly used Rules of Inference are tabulated below –

Rules of Inference	Tautology	Name
$p, p \rightarrow q, \therefore q$	$(p \wedge (p \rightarrow q)) \rightarrow q$	Modus Ponens
$\neg q, p \rightarrow q, \therefore \neg p$	$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus Tollens
$p \rightarrow q, q \rightarrow r, \therefore p \rightarrow r$	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	Hypothetical Syllogism
$\neg p, p \vee q, \therefore q$	$(\neg p \wedge (p \vee q)) \rightarrow q$	Disjunctive Syllogism
$p, \therefore (p \vee q)$	$p \rightarrow (p \vee q)$	Addition
$(p \wedge q) \rightarrow r, \therefore p \rightarrow (q \rightarrow r)$	$((p \wedge q) \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$	Exportation

Rules of Inference	Tautology	Name
$p \vee q, \neg p \vee r, \therefore q \vee r$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$	Resolution

Similarly, we have Rules of Inference for quantified statements –

Rule of Inference	Name
$\forall xP(x)$	Universal instantiation
$P(c)$ for an arbitrary c	Universal generalization
$\exists xP(x)$	Existential instantiation
$P(c)$ for some c	Existential generalization

How To Find Minimal Set?

Last Updated : 25 Jul, 2024

•

If we have a set of functional dependencies, we get the simplest and irreducible form of functional dependencies after reducing these functional dependencies. This is called the **Minimal Cover** or **Irreducible Set** (as we can't reduce the set further). It is also called a **Canonical Cover**.

Let us understand the procedure to find the minimal cover by this example:

The Given Functional Dependencies are – $A \rightarrow B, B \rightarrow C, D \rightarrow ABC, AC \rightarrow D$

1. Steps to Find Minimal Cover

Step 1: First split all the right-hand attributes of all FDs such that RHS contains a single attribute.

Example: $D \rightarrow ABC$ is split into $D \rightarrow A, D \rightarrow B$ and $D \rightarrow C$

$A \rightarrow B, B \rightarrow C, D \rightarrow A, D \rightarrow B, D \rightarrow C, AC \rightarrow D$

[Note: We can't split $AC \rightarrow D$ as $A \rightarrow D, C \rightarrow D$]

Step 2: Now remove all redundant FDs.

[Redundant FD is if we derive one FD from another FD]

Let, 's test the redundancy of $A \rightarrow B$

$A^+ = A$ (A is only closure contains to A, simply we can derive A from A) So, $A \rightarrow B$ is not redundant.

Similarly, $B \rightarrow C$ is not redundant.

But, $D \rightarrow B$ and $D \rightarrow C$ is redundant

because $D^+ = A$ and $A^+ = B$, So $D^+ = B$ can be derived which means $D \rightarrow B$ is redundant. So, We remove $D \rightarrow B$ from the FDs set.

Now, check for $D \rightarrow C$, it is not redundant because we can't $D^+ = B$ and $B^+ = C$ as we remove $D \rightarrow B$ from the list.

At last, we check for $AC \rightarrow D$. This is also not redundant.

$AC \neq AC$

So, the final FDs are: $A \rightarrow B$, $B \rightarrow C$, $D \rightarrow A, D \rightarrow C$, $AC \rightarrow D$

Step 3: Find the Extraneous attribute and remove it.

In this case, we should only check $AC \rightarrow D$. Simply we can say the right-hand attributes are pointed by only one attribute at one time.

$AC \rightarrow D$, either A or C, or none can be extraneous.

If $A \neq C$ then C is extraneous and it can be removed.

If $C \neq A$ then A is extraneous and it can be removed.

So, the final FDs are: $A \rightarrow B$, $B \rightarrow C$, $D \rightarrow A, D \rightarrow C$, $AC \rightarrow D$

Hence, we can write it as $A \rightarrow B$, $B \rightarrow C$, $D \rightarrow AC$, $AC \rightarrow D$ this is the minimum cover.

Find the Minimal Cover

Given Functional Dependencies

- $A \rightarrow B$
- $B \rightarrow C$
- $D \rightarrow ABC$
- $AC \rightarrow D$

Step 1: Split the Functional Dependencies

- $A \rightarrow B$
- $B \rightarrow C$
- $D \rightarrow A$
- $D \rightarrow B$
- $D \rightarrow C$
- $AC \rightarrow D$

Step 2: Remove Redundant FDs

- $A \rightarrow B$ (not redundant)
- $B \rightarrow C$ (not redundant)
- $D \rightarrow A$ (not redundant)
- $D \rightarrow B$ (redundant, because $D \rightarrow A$ and $A \rightarrow B$)
- $D \rightarrow C$ (not redundant, as $D \rightarrow B$ was removed)
- $AC \rightarrow D$ (not redundant)

After removing redundancies FDs set became

- $A \rightarrow B$
- $B \rightarrow C$
- $D \rightarrow A$
- $D \rightarrow C$
- $AC \rightarrow D$

Step 3: Remove Extraneous Attributes

- In $AC \rightarrow D$, check if A or C is extraneous.
- Compute closures

- $A^+ = \{A, B, C\}$
- $C^+ = \{C\}$
- Since **A** alone does not give D, A is not extraneous
- Since **C** alone does not give D, C is not extraneous

So Minimal cover of $(A \rightarrow B, B \rightarrow C, D \rightarrow ABC, AC \rightarrow D) \Rightarrow (A \rightarrow B, B \rightarrow C, D \rightarrow A, AC \rightarrow D)$

By ensuring all functional dependencies are minimal and non-redundant, we obtain the correct minimal cover.

Properties of Relational Decomposition

When a relation in the relational model is not appropriate normal form then the decomposition of a relation is required. In a database, breaking down the table into multiple tables termed as decomposition. The properties of a relational decomposition are listed below :

1. Attribute Preservation:

Using functional dependencies the algorithms decompose the universal relation schema R in a set of relation schemas $D = \{ R_1, R_2, \dots, R_n \}$ relational database schema, where 'D' is called the Decomposition of R.

The attributes in R will appear in at least one relation schema R_i in the decomposition, i.e., no attribute is lost. This is called the *Attribute Preservation* condition of decomposition.

2. [Dependency Preservation](#):

If each functional dependency $X \rightarrow Y$ specified in F appears directly in one of the relation schemas R_i in the decomposition D or could be inferred from the dependencies that appear in some R_i . This is the *Dependency Preservation*.

If a decomposition is not dependency preserving some dependency is lost in decomposition. To check this condition, take the JOIN of 2 or more relations in the decomposition.

For example:

$R = (A, B, C)$

$F = \{A \rightarrow B, B \rightarrow C\}$

Key = {A}

R is not in BCNF.

Decomposition $R_1 = (A, B), R_2 = (B, C)$

R_1 and R_2 are in BCNF, Lossless-join decomposition, Dependency preserving.

Each Functional Dependency specified in F either appears directly in one of the relations in the decomposition.

It is not necessary that all dependencies from the relation R appear in some relation R_i .

It is sufficient that the union of the dependencies on all the relations R_i be equivalent to the dependencies on R.

3. Non Additive Join Property:

Another property of decomposition is that D should possess is the *Non Additive Join*

Property, which ensures that no spurious tuples are generated when a NATURAL JOIN operation is applied to the relations resulting from the decomposition.

4. **No redundancy:**

Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy. If the relation has no proper decomposition, then it may lead to problems like loss of information.

5. Lossless Join:

Lossless join property is a feature of decomposition supported by normalization. It is the ability to ensure that any instance of the original relation can be identified from corresponding instances in the smaller relations.

For example:

R : relation, F : set of functional dependencies on R ,

X, Y : decomposition of R ,

A decomposition $\{R_1, R_2, \dots, R_n\}$ of a relation R is called a lossless decomposition for R if the natural join of $R_1, R_2, \dots, R_n$ produces exactly the relation R .

A decomposition is lossless if we can recover:

$R(A, B, C) \rightarrow \text{Decompose} \rightarrow R_1(A, B) \ R_2(A, C) \rightarrow \text{Recover} \rightarrow R'(A, B, C)$

Thus, $R' = R$

Decomposition is lossless if:

$X \cap Y \rightarrow X$, that is: all attributes common to both X and Y functionally determine ALL the attributes in X .

$X \cap Y \rightarrow Y$, that is: all attributes common to both X and Y functionally determine ALL the attributes in Y

If $X \cap Y$ forms a superkey of either X or Y , the decomposition of R is a lossless decomposition.

Normal Forms Based on Primary Keys

Normalization is a process of organizing data in a database to reduce redundancy and improve data consistency. Primary keys are really important in organizing information in a database. They help to make sure that every row in a table has a unique identification so that nothing gets mixed up or lost.

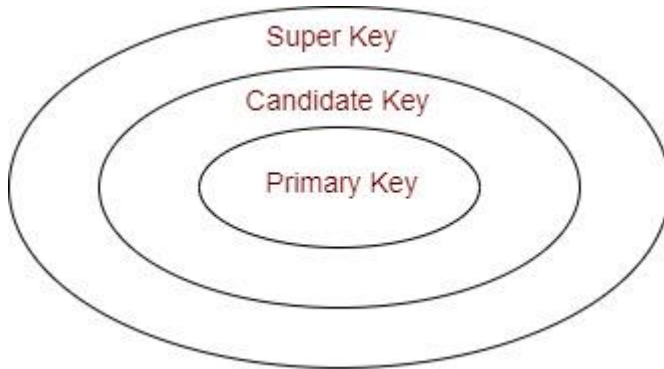
In this article, we will discuss different normal forms based on primary keys and their importance in ensuring data consistency.

Introduction

Let's discuss how we make sure that data in databases is organized properly. We use something called "functional dependencies" to help us do this. Each table in the database has a special identifier called a "primary key." We also have some rules that help us organize the data properly, called "normal forms."

Types of Keys used in Normalization

There are several types of keys used in normalization in database management systems (DBMS). These are explained as follows.



Super Key

A super key is a set of one or more attributes that uniquely identifies each record in a table. A super key may contain more attributes than necessary to uniquely identify each record.

Candidate Key

A candidate key is a minimal super key that can uniquely identify each record in a table. In other words, it is a super key that does not contain any unnecessary attributes.

Primary Key

A primary key is a candidate key that has been selected to uniquely identify each record in a table. It is used to enforce entity integrity, and is typically denoted by an underline or a key symbol.

Alternate Key

An alternate key is a candidate key that is not selected to be the primary key. It is used to enforce uniqueness, and may be used as a reference key in another table.

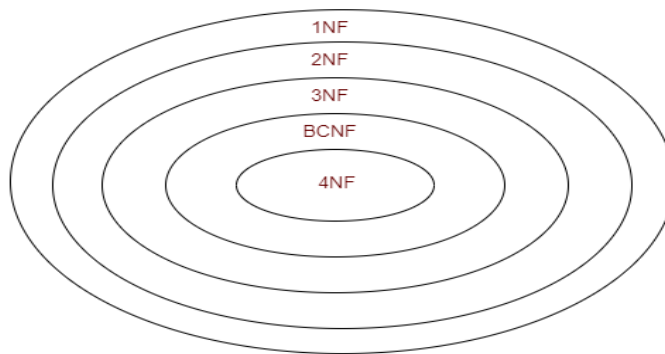
Foreign Key

A foreign key is a key that is used to link two tables together. It is a column (or set of columns) in one table that refers to the primary key of another table.

By using these different types of keys, DBMS designers can ensure that data is organized properly, free of redundancy, and can be retrieved efficiently. Normalization helps to reduce data anomalies and inconsistencies, and ensures that databases are scalable and easy to maintain.

Normal Forms Based on Primary Keys

Normalization is the process of organizing data in a database to minimize redundancy and dependency. In database design, there are different normal forms based on the primary keys of a table. These include –



First Normal Form (1NF)

1NF requires that each column in a table contains atomic values and that each row is uniquely identified. This means that a table cannot have repeating groups or arrays as columns, and each row must have a unique primary key. For example, a table that lists customer orders with a column for order items violates 1NF because the order items are not atomic values.

Second Normal Form (2NF)

2NF builds on 1NF by requiring that each non-primary key column in a table is fully functionally dependent on the primary key. This means that a table should not have partial dependencies, where a non-primary key column depends on only part of the primary key. For example, a table that lists customer orders with a primary key of order ID and non-primary key columns for customer ID and customer name violates 2NF because customer name depends on only customer ID and not on the full primary key.

Third Normal Form (3NF)

3NF builds on 2NF by requiring that each non-primary key column in a table is not transitively dependent on the primary key. This means that a table should not have transitive dependencies, where a non-primary key column depends on another non-primary key

column. For example, a table that lists customer orders with a primary key of order ID and non-primary key columns for customer ID, customer name, and customer city violates 3NF because customer city depends on only customer ID and not on the full primary key.

Boyce-Codd Normal Form (BCNF)

BCNF is a stricter form of 3NF that applies to tables with more than one candidate key. BCNF requires that each non-trivial dependency in a table is a dependency on a candidate key. This means that a table should not have non-trivial dependencies, where a non-primary key column depends on another non-primary key column. BCNF ensures that each table in a database is a separate entity and eliminates redundancies.

Fourth Normal Form (4NF)

4NF builds on BCNF by requiring that a table should not have multi-valued dependencies. A multi-valued dependency occurs when a non-primary key column depends on a combination of other non-primary key columns. For example, a table that lists customer orders with a primary key of order ID and non-primary key columns for customer ID and order items violates 4NF because order items depend on both order ID and customer ID.

Difference Between 2NF and 3NF

2NF(Second Normal Form)	3NF(Third Normal Form)
It is already in 1NF.	It is already in 1NF as well as in 2NF also.
In 2NF non-prime attributes are allowed to be functionally dependent on non-prime attributes.	In 3NF non-prime attributes are only allowed to be functionally dependent on Super key of relation.
No partial functional dependency of non-prime attributes are on any proper subset of candidate key is allowed.	No transitive functional dependency of non-prime attributes on any super key is allowed. .
Stronger normal form than 1NF but lesser than 3NF	Stronger normal form than 1NF and 2NF.

2NF(Second Normal Form)	3NF(Third Normal Form)
It eliminates repeating groups in relation.	It virtually eliminates all the redundancies.
The goal of the second normal form is to eliminate redundant data.	The goal of the third normal form is to ensure referential integrity.

Boyce-Codd Normal Form (BCNF)

- While Third Normal Form (3NF) is generally sufficient for organizing relational databases, it may not completely eliminate redundancy. Redundancy can still occur if there's a dependency $X \rightarrow X$ where X is not a candidate key. This issue is addressed by a stronger normal form known as **Boyce-Codd Normal Form (BCNF)**. Applying the rules of 2NF and 3NF can help identify some redundancies caused by dependencies that violate candidate keys. However, even with these rules, certain dependencies may still lead to redundancy in 3NF. To overcome this limitation, BCNF was introduced by Codd in 1974 as a more robust solution.

Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form (BCNF) is a stricter version of Third Normal Form (3NF) that ensures a more simplified and efficient database design.

It enforces that every non-trivial functional dependency must have a superkey on its left-hand side.

This approach addresses potential issues with candidate keys and ensures the database is free from redundancy.

BCNF eliminates redundancy more effectively than 3NF by strictly requiring that all functional dependencies originate from super-keys.

BCNF is essential for good [database schema design](#) in higher-level systems where consistency and efficiency are important, particularly when there are many candidate keys (as one often finds with a delivery system).

Example 1

Consider a relation R with attributes (student, teacher, subject).

FD: { (student, Teacher) $\rightarrow$ subject, (student, subject) $\rightarrow$ Teacher, (Teacher) $\rightarrow$ subject }

- [Candidate keys](#) are (student, teacher) and (student, subject).
- The above relation is in 3NF (since there is no transitive dependency). A relation R is in BCNF if for every non-trivial FD $X \rightarrow Y$, X must be a key.
- The above relation is not in BCNF, because in the FD (teacher $\rightarrow$ subject), teacher is not a key. This relation suffers with anomalies –

AI-DS Department

- For example, if we delete the student Tahira , we will also lose the information that N.Gupta teaches C. This issue occurs because the teacher is a determinant but not a candidate key.

R is divided into two relations R1(Teacher, Subject) and R2(Student, Teacher).
For more, refer to [BCNF in DBMS](#).

Example 2

Let us consider the student database, in which data of the student are mentioned.

Stu_ID	Stu_Branch	Stu_Course	Branch_Number	Stu_Course_No
101	Computer Science & Engineering	DBMS	B_001	201
101	Computer Science & Engineering	Computer Networks	B_001	202
102	Electronics & Communication Engineering	VLSI Technology	B_003	401
102	Electronics & Communication Engineering	Mobile Communication	B_003	402

Functional Dependency of the above is as mentioned:

Stu_ID → Stu_Branch

Stu_Course → {Branch_Number, Stu_Course_No}

Dependencies and normal Forms

Dependency normal forms are a set of rules that define how normalized a database is. They are used to determine the degree of normalization for a database.

Types of dependency normal forms

- First Normal Form (1NF)**

Each attribute in a table is atomic, meaning it is indivisible.

- **Second Normal Form (2NF)**

A table is in 1NF, and all non-primary key attributes are functionally dependent on the primary key.

- **Third Normal Form (3NF)**

A table is in 2NF, and all non-primary key attributes are not functionally dependent on any non-primary key attributes.

- **Boyce-Codd Normal Form (BCNF)**

A table is in 3NF, and for every Functional Dependency, LHS is the super key.
Benefits of dependency normal forms

- **Data consistency**

Normal forms ensure that data is consistent and doesn't contain redundant information.

- **Data redundancy**

Normal forms minimize data redundancy by organizing data into tables that contain only unique data.

- **Query performance**

Normal forms can improve query performance by reducing the number of joins required to retrieve data.

Unit IV

Indexing and Hashing

Difference between Indexing and Hashing in DBMS

~~Efficient Data Retrieval, Efficient Sorting and Searching, Reduces table space by storing just pointers to data rather than the actual data, hence minimizing storage capacity, Supports Random Lookups, By providing efficient access to ordered data, this feature helps to speed up random lookups.~~
What is Indexing?

Indexing, as the name suggests, is a technique or mechanism generally used to speed up access of data. The index is a type of data structure that is used to locate and access data in a database table quickly. Indexes can easily be developed or created using one or more columns of a [database table](#).

Advantages of Indexing

- **Faster Data Retrieval:** Indexing improves query speed by drastically lowering the amount of disk accesses needed to obtain data.
- **Efficient Sorting and Searching:** This makes it easier to quickly retrieve sorted data, which is useful for activities like organizing, searching, and grouping.
- Reduces table space by storing just pointers to data rather than the actual data, hence minimizing storage capacity.
- **Supports Random Lookups:** By providing efficient access to ordered data, this feature helps to speed up random lookups.

Disadvantages of Indexing

- **Increased Maintenance Overhead:** When indexes are updated often, this may lead to an increase in maintenance overhead and the need for extra storage.
- **Not Suitable for Big Databases:** Performance may suffer if the database is too large or has an excessive number of indexes, which may slow down writes and updates.
- **Performance Impact on Insertions and Updates:** The need to update the indexes after each data insertion, deletion, or update may lead to slower data modification operations.

What is Hashing?

Hashing, as name suggests, is a technique or mechanism that uses hash functions with search keys as parameters to generate address of data record. It calculates direct location of data record on disk without using index structure. A good hash functions only uses one-way hashing algorithm and hash cannot be converted back into original key. In simple words, it is a process of converting given key into another value known as hash value or simply hash.

Advantages of Hashing

- This approach, which determines the precise storage location using a hash function, enables quick and easy access to data.
- efficient with large databases Large databases may benefit from its ability to handle large volumes of data without negatively affecting search performance.
- **Increased Flexibility and Reliability:** By organizing data into easily searchable “buckets,” this method offers a trustworthy means of retrieving information.
- **Fast Search Results:** it allows for efficient comparison of large datasets and is faster than more traditional data structures like lists and [arrays](#).

Disadvantages of Hashing

- **Fixed Hash Values:** When two keys map to the same location, the hash function produces a fixed-length hash result that might cause collisions.
- **Unsuitable for Range Inquiries:** Hashing is ineffective when doing ordered retrievals or range searches.

- **Data Integrity Problems:** Data integrity problems may arise from improper handling of hash collisions.
- **Complex Execution:** To avoid collisions and provide a consistent data distribution, [hash functions](#) need to be carefully selected.

Difference Between Indexing and Hashing in DBMS

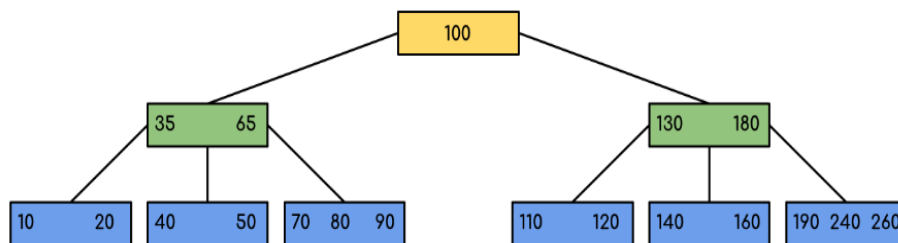
Indexing	Hashing
It is a technique that allows to quickly retrieve records from database file.	It is a technique that allows to search location of desired data on disk without using index structure.
It is generally used to optimize or increase performance of database simply by minimizing number of disk accesses that are required when a query is processed.	It is generally used to index and retrieve items in database as it is faster to search that specific item using shorter hashed key rather than using its original value.
It offers faster search and retrieval of data to users, helps to reduce table space, makes it possible to quickly retrieve or fetch data, can be used for sorting, etc.	It is faster than searching arrays and lists, provides more flexible and reliable method of data retrieval rather than any other data structure , can be used for comparing two files for quality, etc.
Its main purpose is to provide basis for both rapid random lookups and efficient access of ordered records.	Its main purpose is to use math problem to organize data into easily searchable buckets.
It is not considered best for large databases and its good for small databases.	It is considered best for large databases.
Types of indexing includes ordered indexing, primary indexing , secondary indexing , clustered indexing .	Types of hashing includes static hashing and dynamic hashing .
It uses data reference to hold address of disk block.	It uses mathematical functions known as hash function to calculate direct location of records on disk.
It is important because it protects file and documents of large size business organizations, and optimize performance of database.	It is important because it ensures data integrity of files and messages, takes variable length string or messages and compresses and converts it into fixed length value.

Introduction of B-Tree

A B-Tree is a specialized m-way tree designed to optimize data access, especially on disk-based storage systems.

- In a B-Tree of order **m**, each node can have up to m children and m-1 keys, allowing it to efficiently manage large datasets.
- The value of m is decided based on disk block and key sizes.
- One of the standout features of a B-Tree is its ability to store a significant number of keys within a single node, including large key values. It significantly reduces the tree's height, hence reducing costly disk operations.
- B-Trees allow faster data retrieval and updates, making them an ideal choice for systems requiring efficient and scalable data management. By maintaining a balanced structure at all times,
- B-Trees deliver consistent and efficient performance for critical operations such as search, insertion, and deletion.

Following is an example of a B-Tree of order 5 .



Properties of a B-Tree

A B Tree of order m can be defined as an m-way search tree which satisfies the following properties:

1. All leaf nodes of a B tree are at the same level, i.e. they have the same depth (height of the tree).
2. The keys of each node of a B tree (in case of multiple keys), should be stored in the ascending order.
3. In a B tree, all non-leaf nodes (except root node) should have at least $m/2$ children.
4. All nodes (except root node) should have at least $m/2 - 1$ keys.
5. If the root node is a leaf node (only node in the tree), then it will have no children and will have at least one key. If the root node is a non-leaf node, then it will have at least 2 children and at least one key.
6. A non-leaf node with n-1 key values should have n non NULL children.

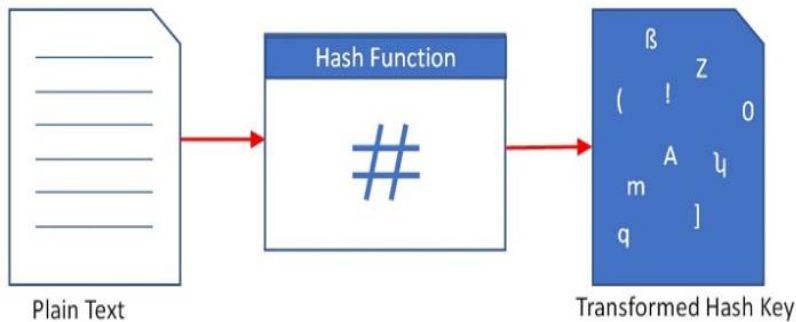
Difference between Static and Dynamic Hashing

Hashing is a computation technique in which hashing functions take variable-length data as input and issue a shortened fixed-length data as output. The output data is often called a "Hash Code", "Key", or simply "Hash". The data on which hashing works is called a "Data Bucket".

Characteristics of Hashing Technique

Hashing techniques come with the following characteristics –

- The first characteristic is, hashing technique is deterministic. Means, whatever number of times you invoke the function on the same test variable, it delivers the same fixed-length result.
- The second characteristic is its unidirectional action. There is no way you can use the Key to retrieve the original data. Hashing is irreversible.



What are Hash Functions?

Hash functions are mathematical functions that are executed to generate addresses of data records. Hash functions use memory locations that store data, called ‘Data Buckets’.

Hash functions are used in cryptographic signatures, securing privacy of vulnerable data, and verifying correctness of the received files and texts. In computation, hashing is used in data processing to locate a single string of data in an array, or to calculate direct addresses of records on the disk by requesting its Hash Code or Key.

Applications of Hashing

Hashing is applicable in the following area –

- Password verification
- Associating filename with their paths in operating systems
- Data Structures, where a key-value pair is created in which the key is a unique value, whereas the value associated with the keys can be either same or different for different keys.
- Board games such as Chess, tic-tac-toe, etc.
- Graphics processing, where a large amount of data needs to be matched and fetched.

Database Management Systems where phenomenal records are required to be searched, queried, and matched for retrieval. For example, DBMS used in banking or large public transport reservation software.

Read through this article to find out more about Hashing and specifically the difference between two important hashing techniques – *static hashing* and *dynamic hashing*.

What is Static Hashing?

It is a hashing technique that enables users to lookup a definite data set. Meaning, the data in the directory is not changing, it is "Static" or fixed. In this hashing technique, the resulting number of data buckets in memory remains constant.

Operations Provided by Static Hashing

Static hashing provides the following operations –

- **Delete** – Search a record address and delete a record at the same address or delete a chunk of records from records for that address in memory.
- **Insertion** – While entering a new record using static hashing, the hash function (h) calculates bucket address " $h(K)$ " for the search key (k), where the record is going to be stored.
- **Search** – A record can be obtained using a hash function by locating the address of the bucket where the data is stored.
- **Update** – It supports updating a record once it is traced in the data bucket.

Advantages of Static Hashing

Static hashing is advantageous in the following ways –

- Offers unparalleled performance for small-size databases.
- Allows Primary Key value to be used as a Hash Key.

Disadvantages of Static Hashing

Static hashing comes with the following disadvantages –

- It cannot work efficiently with the databases that can be scaled.
- It is not a good option for large-size databases.
- Bucket overflow issue occurs if there is more data and less memory.

What is Dynamic Hashing?

It is a hashing technique that enables users to lookup a dynamic data set. Means, the data set is modified by adding data to or removing the data from, on demand hence the name

‘Dynamic’ hashing. Thus, the resulting data bucket keeps increasing or decreasing depending on the number of records.

In this hashing technique, the resulting number of data buckets in memory is ever-changing.

Operations Provided by Dynamic Hashing

Dynamic hashing provides the following operations –

- **Delete** – Locate the desired location and support deleting data (or a chunk of data) at that location.
- **Insertion** – Support inserting new data into the data bucket if there is a space available in the data bucket.
- **Query** – Perform querying to compute the bucket address.
- **Update** – Perform a query to update the data.

Advantages of Dynamic Hashing

Dynamic hashing is advantageous in the following ways –

- It works well with scalable data.
- It can handle addressing large amount of memory in which data size is always changing.
- Bucket overflow issue comes rarely or very late.

Disadvantages of Dynamic Hashing

Dynamic hashing comes with the following disadvantage –

- The location of the data in memory keeps changing according to the bucket size. Hence if there is a phenomenal increase in data, then maintaining the bucket address table becomes a challenge.

Differences between Static and Dynamic Hashing

Here are some prominent differences by which Static Hashing is different than Dynamic Hashing –

Key Factor	Static Hashing	Dynamic Hashing
Form of Data	Fixed-size, non-changing data.	Variable-size, changing data.
Result	The resulting Data Bucket is of fixed-length.	The resulting Data Bucket is of variable-length.
Bucket Overflow	Challenge of Bucket overflow can arise often depending upon memory size.	Bucket overflow can occur very late or doesn't occur at all.
Complexity	Simple	Complex

Difference between Indexing and Hashing in DBMS

Indexing and hashing are two crucial techniques used in databases to improve the efficiency of data retrieval and query performance. You can search and retrieve entries from databases rapidly thanks to a data structure that indexing makes feasible.

What is Indexing?

Indexing, as the name suggests, is a technique or mechanism generally used to speed up access of data. The index is a type of data structure that is used to locate and access data in a database table quickly. Indexes can easily be developed or created using one or more columns of a [database table](#).

Advantages of Indexing

- **Faster Data Retrieval:** Indexing improves query speed by drastically lowering the amount of disk accesses needed to obtain data.
- **Efficient Sorting and Searching:** This makes it easier to quickly retrieve sorted data, which is useful for activities like organizing, searching, and grouping.
- Reduces table space by storing just pointers to data rather than the actual data, hence minimizing storage capacity.
- **Supports Random Lookups:** By providing efficient access to ordered data, this feature helps to speed up random lookups.

Disadvantages of Indexing

- **Increased Maintenance Overhead:** When indexes are updated often, this may lead to an increase in maintenance overhead and the need for extra storage.
- **Not Suitable for Big Databases:** Performance may suffer if the database is too large or has an excessive number of indexes, which may slow down writes and updates.
- **Performance Impact on Insertions and Updates:** The need to update the indexes after each data insertion, deletion, or update may lead to slower data modification operations.

What is Hashing?

Hashing, as name suggests, is a technique or mechanism that uses hash functions with search keys as parameters to generate address of data record. It calculates direct location of data record on disk without using index structure.

Advantages of Hashing

- This approach, which determines the precise storage location using a hash function, enables quick and easy access to data.
- efficient with large databases Large databases may benefit from its ability to handle large volumes of data without negatively affecting search performance.
- **Increased Flexibility and Reliability:** By organizing data into easily searchable “buckets,” this method offers a trustworthy means of retrieving information.

- **Fast Search Results:** it allows for efficient comparison of large datasets and is faster than more traditional data structures like lists and [arrays](#).

Disadvantages of Hashing

- **Fixed Hash Values:** When two keys map to the same location, the hash function produces a fixed-length hash result that might cause collisions.
- **Unsuitable for Range Inquiries:** Hashing is ineffective when doing ordered retrievals or range searches.
- **Data Integrity Problems:** Data integrity problems may arise from improper handling of hash collisions.
- **Complex Execution:** To avoid collisions and provide a consistent data distribution, [hash functions](#) need to be carefully selected.

Difference Between Indexing and Hashing in DBMS

Indexing	Hashing
It is a technique that allows to quickly retrieve records from database file.	It is a technique that allows to search location of desired data on disk without using index structure.
Its main purpose is to provide basis for both rapid random lookups and efficient access of ordered records.	Its main purpose is to use math problem to organize data into easily searchable buckets.
It is not considered best for large databases and its good for small databases.	It is considered best for large databases.
Types of indexing includes ordered indexing, primary indexing , secondary indexing , clustered indexing .	Types of hashing includes static hashing and dynamic hashing .
It uses data reference to hold address of disk block.	It uses mathematical functions known as hash function to calculate direct location of records on disk.
It is important because it protects file and documents of large size business organizations, and optimize performance of database.	It is important because it ensures data integrity of files and messages, takes variable length string or messages and compresses and converts it into fixed length value.

Bitmap Indexing in DBMS

Bitmap Indexing is a data indexing technique used in database management systems (DBMS) to improve the performance of read-only queries that involve large datasets. It involves creating a bitmap index, which is a data structure that represents the presence or absence of data values in a table or column.

In a bitmap index, each distinct value in a column is assigned a bit vector that represents the presence or absence of that value in each row of the table. The bit vector contains one bit for each row in the table, where a set bit indicates the presence of the corresponding value in the row, and a cleared bit indicates the absence of the value.

Bitmap Index Structure

A bitmap is the combination of two words: **bit** and **map**. A bit can be termed as the smallest unit of data in a computer and a map can be termed as a way of organizing things.

Bit: A bit is a basic unit of information used in computing that can have only one of two values either 0 or 1. The two values of a binary digit can also be interpreted as logical values true/false or Yes/No.

Bitmap Indexing is a special type of database indexing that uses **bitmaps**. This technique is used for huge databases when the column is of low cardinality and these columns are most frequently used in the query.

Bitmap indexing is a data structure used in [database management systems \(DBMS\)](#) to efficiently represent and query large datasets with many attributes (columns). Bitmap indexes use a compact binary representation to store the occurrence of each value or combination of values in each attribute, allowing for fast, set-based operations.

Features of Bitmap Indexing in DBMS

- **Space efficiency:** Bitmap indexes are highly space-efficient because they use a compact binary representation to store the occurrence of each value or combination of values in each attribute. This makes them especially useful for large datasets with many attributes.
- **Fast query processing:** Bitmap indexes can be used to quickly answer complex queries involving multiple attributes using set-based operations such as AND, OR, and NOT. This allows for fast query processing and reduces the need for full table scans.
- **Low maintenance overhead:** Bitmap indexes require relatively low maintenance overhead because they can be updated incrementally as data changes. This makes them especially useful for applications where the data is frequently updated.
- **Flexibility:** Bitmap indexes can be used for both numerical and categorical data types, and can also be used to index text data using techniques such as term frequency-inverse document frequency (TF-IDF).
- **Reduced I/O overhead:** Bitmap indexes can be used to avoid expensive I/O operations by using a compressed representation of the data. This reduces the amount of data that needs to be read from the disk, improving query performance.
- **Ideal Choice:** Bitmap indexing is a powerful technique for efficiently querying large datasets with many attributes. It's compact representation and set-based operations make it an ideal choice for data warehousing and other applications where fast query processing is critical.

Applications of Bitmap Indexing in DBMS

- **Fast queries on large datasets:** Bitmap indexing is particularly useful for fast queries on large datasets. When querying a table with a bitmap index, the database engine can quickly determine which rows satisfy the query by performing a bitwise operation on the corresponding bitmaps. This can greatly reduce query execution time, especially for queries that involve multiple columns or complex conditions.
- **Efficient range queries:** Bitmap indexing can also be used for efficient range queries on numeric or date columns. The bitmap for a range of values can be constructed by performing bitwise operations on the bitmaps for individual values. This allows the database engine to quickly identify all rows that fall within a given range.
- **Space efficiency:** Bitmap indexing can be more space-efficient than other indexing techniques, especially for columns with a small number of distinct values. For example, a column with only two distinct values (such as gender) can be represented using a single bitmap, while other indexing techniques would require multiple index entries.
- **Multi-dimensional indexing:** Bitmap indexing can be used for multi-dimensional indexing by creating a bitmap for each dimension. The bitmaps can then be combined using bitwise operations to identify rows that satisfy a query with multiple conditions.
- **Data warehousing applications:** Bitmap indexing is commonly used in [data warehousing applications](#) where the focus is on fast query performance. In these applications, tables often have large numbers of columns with a relatively small number of distinct values, making bitmap indexing a particularly effective technique.

What is the Need for Bitmap Indexing?

The need for Bitmap Indexing will be clear through the below-given example:

For example, Let us say that a company holds an employee table with entries like EmpNo, EmpName, Job, New_Emp, and salary. Let us assume that the employees are hired once a year, therefore the table will be updated very less and will remain static most of the time. But the columns will be frequently used in queries to retrieve data like: No. of female employees in the company etc. In this case, we need a [file organization](#) method that should be fast enough to give quick results. But any of the traditional file organization methods are not that fast, therefore we switch to a better method of storing and retrieving data known as Bitmap Indexing.

Bitmap indexing can be used to improve the performance of queries that involve complex logical operations, such as AND, OR, and NOT. The logical operations are performed on the bit vectors, which can be efficiently combined using bitwise operators.

Bitmap indexing is particularly useful for datasets with low cardinality columns, where each value appears in a large number of rows. It is also useful for read-only queries that involve aggregations, such as counting or summing values, as the bit vectors can be quickly scanned to retrieve the required data.

How Bitmap Indexing is Done?

In the above example of table employee, we can see that the column New_Emp has only two values **Yes** and **No** based upon the fact that the employee is new to the company or not. Similarly let us assume that the Job of the Employees is divided into 4 categories only i.e Manager, Analyst, Clerk, and Salesman. Such columns are called columns with low cardinality. Even though these columns have fewer unique values, they can be queried very often.

Bit: A bit is a basic unit of information used in computing that can have only one of two values either 0 or 1. The two values of a binary digit can also be interpreted as logical values true/false or yes/no.

AI-DS Department

In Bitmap Indexing these bits are used to represent the unique values in those low [cardinality](#) columns. This technique of storing the low cardinality rows in the form of bits is called bitmap indices.

Continuing the Employee example, Given below is the Employee table.

EmpNo	EmpName	Job	New_Emp	Salary
1	Alice	Analyst	Yes	15000
2	Joe	Salesperson	No	10000
3	Katy	Clerk	No	12000
4	Annie	Manager	Yes	25000

Bitmap_Indexing_1

If New_Emp is the data to be indexed, the content of the bitmap index is shown as four(As we have four rows in the above table) columns under the heading Bitmap Indices. Here Bitmap Index “Yes” has a value of 1001 because rows 1 and row four have the value “Yes” in column New_Emp.

New_Emp Values	Bitmap Indices
Yes	1001
No	0110

Bitmap_Indexing_2

In this case, there are two such bitmaps, one for “New_Emp” Yes and one for “New_Emp” NO. It is easy to see that each bit in bitmap indices shows whether a particular row refers to a person who is New to the company or not.

The above scenario is the simplest form of Bitmap Indexing. Most columns will have more distinct values. For example, the column Job here will have only 4 unique values (As mentioned earlier). Variations on the bitmap index can effectively index this data as well. For the Job column the bitmap Indexing is shown below:

Job Values	Bitmap Indices
Analyst	1000
Salesperson	0100
Clerk	0010
Manager	0001

Bitmap_Indexing_3

Now Suppose, If we want to find out the details for the Employee who is not new in the company and is a salesperson then we will run the query.

```
SELECT *  
FROM Employee  
WHERE New_Emp = "No" and Job = "Salesperson";
```

For this query, the DBMS will search the bitmap index of both columns and perform logical AND operation on those bits and find out the actual result:

Bitmap Index for "NO"	→	0 1 1 1
		AND
Bitmap Index for Salesperson	→	0 1 0 0

Result	→	0 1 0 0

Bitmap_Indexing_4

Here the result 0100 represents that the second row has to be retrieved as a result.

Bitmap Indexing in SQL

The syntax for creating a bitmap index in SQL is given below.

```
CREATE BITMAP INDEX Index_Name
```

```
ON Table_Name (Column_Name);
```

For the above example of the employee table, the bitmap index on column New_Emp will be created as follows:

```
CREATE BITMAP INDEX index_New_Emp  
ON Employee (New_Emp);
```

Advantages of Bitmap Indexing

- Efficiency in terms of insertion deletion and updation.
- Faster retrieval of records

Disadvantages of Bitmap Indexing

- Only suitable for large tables
- Bitmap Indexing is time-consuming

Unit V

Query processing and Optimization

Measures of Query Cost in DBMS

Query Cost is a cost in which the enhancer considers what amount of time your query will require (comparative with absolute clump time).

Then the analyzer attempts to pick the most ideal query plan by taking a glance at your inquiry and insights of your information, attempting a few execution designs, and choosing the most inexpensive of them.

The measures of query cost in DBMS can be done by creating a framework that can make numerous designs for an inquiry. It tends to be finished by the means of contrasting every conceivable arrangement as far as their assessed cost. For working out the net assessed cost of any arrangement, the expense of every activity inside an arrangement ought to be set in a deterministic and consolidated cost to get the net assessed cost of the query assessment plan.

Example: We utilize the number of square exchanges that is basically the block from the disk and the quantity of the disk seeks to appraise the expense of a query assessment plan. Assuming that the disk subsystem takes a normal of tT seconds to move a square of information and has a normal block access time (disk lookup time in addition to rotational idleness) of tS seconds, then, at that point, an activity that moves b obstructs and performs S looks for would take $b * tT + S * tS$ seconds. The upsides of tT and tS should be aligned for the disk framework utilization, however, normal qualities for top-end disk today would be $tS = 4$ milliseconds and $tT = 0.1$ milliseconds, expecting a 4-kilobyte block size and an exchange pace of 40 megabytes each second.

tT – time to transfer one block

tS – time for one to seek

Cost for b block transfers plus S seeks

$b * tT + S * tS$

The expense assessment of a query assessment plan is determined by keeping in mind the different assets that follow as:

1. The number of disk accesses.
2. Time of Execution taken by the CPU to execute a query.
3. The involved Communication costs in either distributed or parallel database systems.

To gauge the expense of a query assessment plan, utilizing the number of blocks moved from the disk, and the quantity of disk seeks looks for. For the most part, for assessing the expense, we must consider the most pessimistic scenario that could occur which is the worst-case scenario. The clients accept that at first, the information is perused from the disk as it were. However, there should be an opportunity that the data is now present in the principle memory also called the main memory. Notwithstanding, the clients generally

disregard this impact, and because of this, the genuine expense of execution comes out lesser than the assessed esteem.

The reaction time, i.e., the time that is expected so as to execute the arrangement could be utilized for assessing the expense of the query assessment plan. Yet, because of the accompanying reasons, it becomes hard to work out the reaction time without really executing the query assessment plan. As soon as the query starts the process of execution, the time of reaction becomes reliant upon the substance put away in support. In any case, this data is so hard to recover that when the query is in its upgraded mode, or it isn't accessible.

Whenever there is a framework with the various presence of disks available, the time of reaction relies upon a cross-examination that is ready "what is the way that accesses gets to be circulated among the disk available?". It is hard to gauge without having itemized information on the information format present over the disk. Thus, rather than limiting the time of reaction for any query assessment plan, the analyzers observe that it is better to lessen the wholesome asset utilization of the query plan. Along with these lines to appraise the expense of a query assessment plan, it is great to limit the assets utilized for getting to the disk or utilization of the additional assets.

Query operations in a database management system (DBMS) include indexing, query rewriting, and query optimization. These operations help improve the efficiency of queries and the performance of the database.

Query operations in DBMS

Indexing

Helps the database engine find and retrieve relevant rows quickly.



Query rewriting

Modifies queries to make them more efficient. This can include using subqueries, optimizing join operations, and eliminating unnecessary computations.

Query optimization

Improves the performance of a database query by modifying the query's execution plan. The goal is to reduce the time and resources needed to retrieve data.



Join minimization

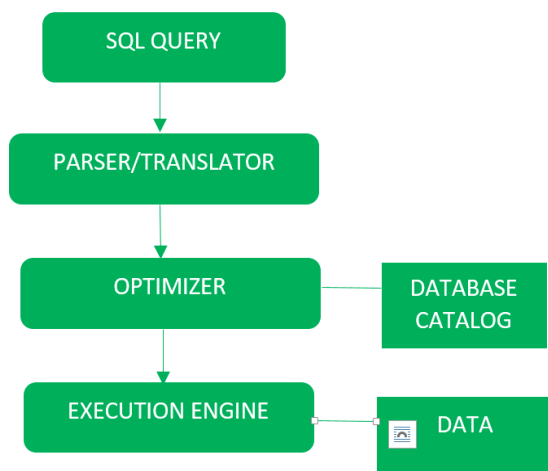
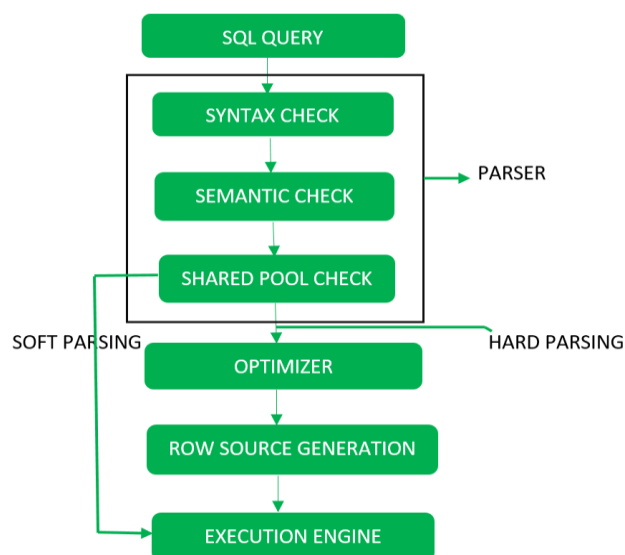
Involves dropping relations from a query that are not required for computing the query.



Query execution plans

Help developers identify bottlenecks and inefficiencies in the query processing system.
Other DBMS operations

- Concurrency control Manages simultaneous operations on the database without them interfering with each other.
- ACID compliance Ensures a safe and secure way of sharing data among multiple users.
- Cost estimation Measures the time taken by a query to hit the database and return the result.



Transformation of Relational Expressions

When a SQL query is submitted to DB, it can be evaluated in number of ways. For example, consider the below case:

```
SELECT EMP_ID, DEPT_NAME
FROM EMP, DEPT
WHERE EMP.DEPT_ID = DEPT.DEPT_ID
AND EMP.DEPT_ID = 10;
```

Above query selects the EMP_ID and DEPT_NAME from EMP and DEPT table for DEPT_ID = 10. But when it is given to the DBMS, it divides the query into tokens and sees how it can be put together so that performance will be better. This is the duty of query optimizer. But altering the order of tokens in the query should not change the result. In either way it should give same result. Order of records can change and are least important. This is called equivalent query. There is set of rules to put tokens in the query. This is called equivalence rule.

Above query can be broken down by the DBMS in either ways below :

- Select the records of EMP with DEPT_ID = 10 first then join them with DEPT table to get all matching records of EMP and DEPT. Then select only the columns EMP_ID and DEPT_NAME to display the result.
- Select all matching records from EMP and DEPT, from which filter on DEPT_ID = 10 and select only EMP_ID and DEPT_NAME to display.

Both the steps above are same irrespective of how it is performed. Hence both are called equivalent query. These are not written in SQL, but using relational algebra, graph or tree.

Π EMP_ID, DEPT_NAME (σ DEPT_ID = 10 (EMP $\bowtie$ DEPT))

or

σ DEPT_ID = 10 (Π EMP_ID, DEPT_NAME, DEPT_ID (EMP $\bowtie$ DEPT))

Above relational algebra and tree shows how DBMS depicts the query inside it. But the cost of both of them may vary.

This is because the number of records in each step changes depending on the join and filters we use, and the algorithms used to evaluate them.

For example we may have huge number of records in second case tree above to filter.

But in the first case we are filtering the record first; hence number of records to join would have been reduced.

This makes lots of difference and query optimizer calculates this difference and selects the optimal tree for query evaluation.

Estimating Statistics of Expression results in DBMS

In order to determine ideal plan for evaluating the query, it checks various details about the tables that are stored in the data dictionary. These informations about tables are collected when a table is created and when various DDL / DML operations are performed on it. The optimizer checks data dictionary for :

- Total number of records in a table, nr . This will help to determine which table needs to be accessed first. Usually smaller **tables** are executed first to reduce the size of the intermediary tables. Hence it is one of the important factors to be checked.
- Total number of records in each block, fr . This will be useful in determining blocking factor and is required to determine if the table fits in the memory or not.
- Total number of blocks assigned to a table, br . This is also an important factor to calculate number of records that can be assigned to each block. Suppose we have 100 records in a table and total number of blocks are 20, then fr can be calculated as $nr/b r = 100/20 = 5$.
- Total length of the records in the table, $l r$. This is an important factor when the size of the records varies significantly between any two tables in the query. If the record length is fixed, there is no significant affect. But when a variable length records are involved in the query, average length or actual length needs to be used depending upon the type of operations.
- Number of unique values for a column, $d Ar$. This is useful when a query uses aggregation operation or projection. It will provide an estimate on distinct number of columns selected while projection. Number groups of records can be determined using this when Aggregation operation is used in the query. E.g.; SUM, MAX, MIN, COUNT etc.
- Levels of index, x . This data provides the information like whether the single level of index like primary key index, secondary key indexes are used or multi-level indexes like B+ tree index, merge-sort index etc are used. These index levels will provide details about number of block access required to retrieve the data.
- Selection cardinality of a column, $s A$. This is the number of records present with same column value as A . This is calculated as $nr/d Ar$. i.e.; total number of records with distinct value of A . For example, suppose EMP table has 500 records and DEPT_ID has 5 distinct values. Then the selection cardinality of DEPT_ID in EMP table is $500/ 5 = 100$. That means, on an average 100 employees are distributed among each department. This is helpful in determining average number of records that would satisfy selection criteria.
- There many other factors too like index type, data file type, sorting order, type of sorting etc.

Query-Evaluation Plan in SQL

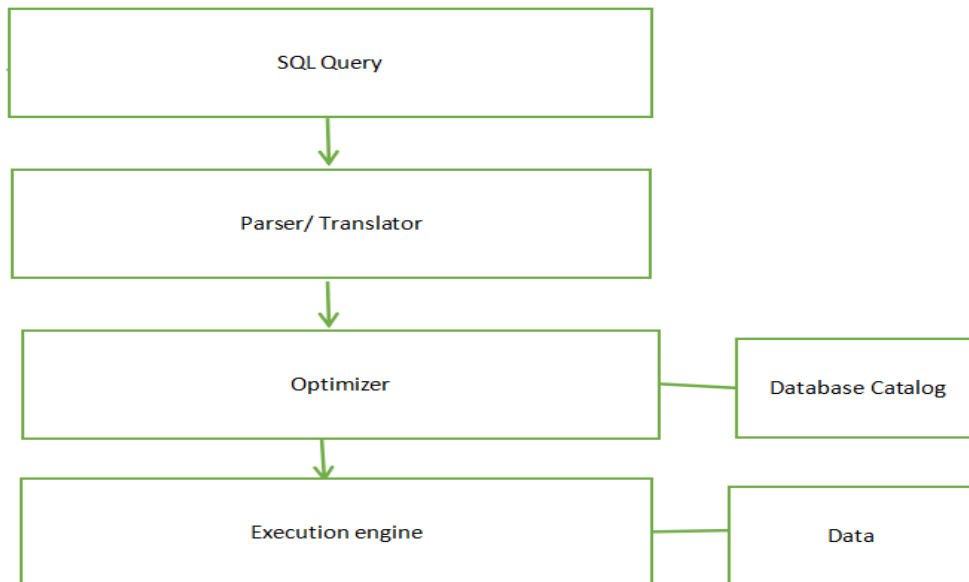
•

Pre requisites: [Query Execution Engine in SQL](#), [Query-Execution Plan in SQL](#)

In this article, we will see about Query Evaluation Plan in SQL and how the system optimizes the given query. Basically, Query Processing in SQL is extracting data from the datasets. There are various steps involved like Parsing and translation, Optimization, and Query Evaluation.

Query Evaluation Plan :

It is nothing but a program for an abstract machine inside the DBMS. It is produced by the query optimizer. At times this can also be termed as access plan because DBMS decide's how to access the rows. Query evaluation plans are very much similar to relational algebra expressions in most of the systems.



Executing a Query Evaluation Plan :

In many relational algebra expressions it is very necessary to translate expression and also to annotate the translated relational algebra expression with the commands used for stating and evaluating operation. And immediately after translating the user query, the system uses the query program.

Characteristics of Query Evaluation Plan:

- It can also be called the Query Questionnaire.
- Systems must create a query system to fully analyze the query.
- Annotations in the test program sometimes refer to the steps used to perform.
- Related algebra with annotations named Evaluation Primitives containing the instructions needed to evaluate the work.
- There is therefore nothing wrong with the fact that the questionnaire program describes a sequence of previous works that are very useful in questioning.
- The result of the query is generated by the query engine.

Optimization :

AI-DS Department

- Since we know that there are different types of questions so the cost of testing will vary, one thing is that the user does not need to write his question correctly because the system does it automatically.
- If the website system produces an effective query program that leads to cost reduction then this type of activity is known as Quiz Development.
- Since all tasks are cost-effective in allocating memory the query caller should have a limited cost analysis for each task.

Unit VI

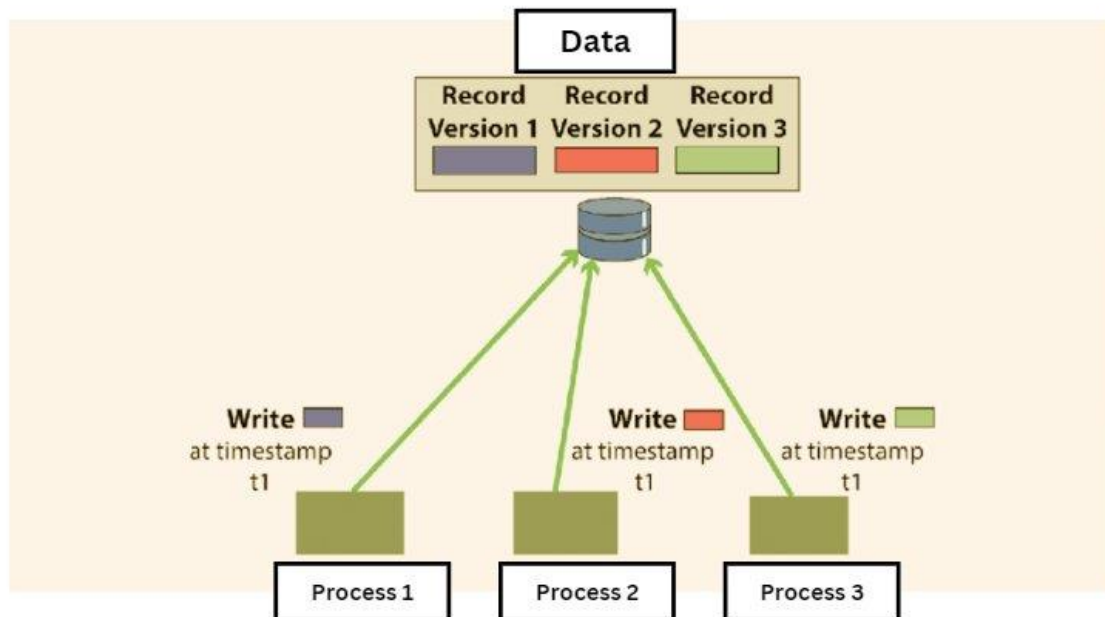
Transaction processing and concurrency control

"Transaction processing" refers to managing a series of database operations as a single unit, ensuring that either all operations within the unit succeed (commit) or none do (rollback), while "concurrency control" is the mechanism within a database management system (DBMS) to manage simultaneous transactions happening on the same data, preventing conflicts and maintaining data consistency when multiple users access the database at the same time; essentially, transaction processing ensures the integrity of individual operations, while concurrency control ensures the integrity of the database when multiple transactions are happening concurrently.

Key points about transaction processing:

- **Atomicity:**
A transaction is treated as a single indivisible unit, meaning all operations within it must complete successfully or none of them will.
 - **Consistency:**
A transaction must bring the database from one valid state to another.
 - **Isolation:**
Concurrent transactions should not interfere with each other, preventing issues like dirty reads and lost updates.
 - **Durability:**
Once a transaction is committed, the changes should persist even in case of system failure.
- Key points about concurrency control:
- **Managing concurrent access:**
Concurrency control mechanisms ensure that multiple transactions accessing the same data do not conflict with each other.
 - **Locking techniques:**
A common approach to concurrency control is using locks on data items, preventing other transactions from accessing and modifying the data while it's locked.
 - **Serializability:**
A concurrency control mechanism aims to ensure that the execution of concurrent transactions appears as if they were executed in a serial order, maintaining data integrity.
Common concurrency control techniques:
 - **Two-phase locking (2PL):**
A transaction acquires all necessary locks in the first phase (growing phase) and releases them only in the second phase (shrinking phase), preventing deadlocks.
 - **Timestamp ordering:**
Each transaction is assigned a timestamp, and operations are ordered based on their timestamps to maintain serializability.
 - **Optimistic concurrency control:**

Transactions assume no conflicts will occur, validating their updates before committing to the database, potentially leading to transaction restarts if conflicts are detected.



Concurrency Control in DBMS

In a database management system (DBMS), allowing transactions to run concurrently has significant advantages, such as better system resource utilization and higher throughput. However, it is crucial that these transactions do not conflict with each other. The ultimate goal is to ensure that the database remains consistent and accurate. For instance, if two users try to book the last available seat on a flight at the same time, the system must ensure that only one booking succeeds. Concurrency control is a critical mechanism in DBMS that ensures the consistency and integrity of data when multiple operations are performed at the same time.

- Concurrency control is a concept in Database Management Systems (DBMS) that ensures multiple transactions can simultaneously access or modify data without causing errors or inconsistencies. It provides mechanisms to handle the concurrent execution in a way that maintains [ACID properties](#).
- By implementing [concurrency control](#), a DBMS allows transactions to execute concurrently while avoiding issues such as deadlocks, race conditions, and conflicts between operations.
- The main goal of concurrency control is to ensure that simultaneous transactions do not lead to data conflicts or violate the consistency of the database. The concept of [serializability](#) is often used to achieve this goal.

In this article, we will explore the various [concurrency control techniques](#) in DBMS, understand their importance, and learn how they enable reliable and efficient database operations.

Concurrent Execution and Related Challenges in DBMS

In a multi-user system, several users can access and work on the same database at the same time. This is known as concurrent execution, where the database is used simultaneously by

different users for various operations. For instance, one user might be updating data while another is retrieving it.

When multiple transactions are performed on the database simultaneously, it is important that these operations are executed in an interleaved manner. This means that the actions of one user should not interfere with or affect the actions of another. This helps in maintaining the consistency of the database. However, managing such simultaneous operations can be challenging, and certain problems may arise if not handled properly. These challenges need to be addressed to ensure smooth and error-free concurrent execution.

Concurrent Execution can lead to various challenges:

- **Dirty Reads:** One transaction reads uncommitted data from another transaction, leading to potential inconsistencies if the changes are later rolled back.
- **Lost Updates:** When two or more transactions update the same data simultaneously, one update may overwrite the other, causing data loss.
- **Inconsistent Reads:** A transaction may read the same data multiple times during its execution, and the data might change between reads due to another transaction, leading to inconsistency.

To read more about Concurrency Problems in DBMS Transactions Refer, [Here](#).

Why is Concurrency Control Needed?

Consider the following example:

- **Without Concurrency Control:** Transactions interfere with each other, causing issues like lost updates, dirty reads or inconsistent results.
- **With Concurrency Control:** Transactions are properly managed (e.g., using locks or timestamps) to ensure they execute in a consistent, isolated manner, preserving data accuracy.

Concurrency control is critical to maintaining the accuracy and reliability of databases in multi-user environments. By preventing conflicts and inconsistencies during concurrent transactions, it ensures the database remains consistent and correct, even under high levels of simultaneous activity.

Concurrency Control Protocols

Concurrency control protocols are the set of rules which are maintained in order to solve the concurrency control problems in the database. It ensures that the concurrent transactions can execute properly while maintaining the database consistency. The concurrent execution of a transaction is provided with atomicity, consistency, isolation, durability, and serializability via the concurrency control protocols.

In transaction processing, "recovery" refers to the mechanism that allows a database system to restore itself to a consistent state after a system failure, by utilizing logs and checkpoints to undo or redo incomplete transactions, ensuring that no data is lost even if a crash occurs during processing; essentially, it's the process of rolling back uncommitted changes or reapplying committed ones to maintain data integrity.

Key points about recovery in transaction processing:

- **Importance of ACID properties:**

Recovery is crucial to uphold the ACID properties (Atomicity, Consistency, Isolation, Durability) of transactions, guaranteeing that all changes within a transaction are either fully committed or completely undone in case of failure.

- **Transaction logs:**

A critical component of recovery, where all transaction operations are recorded in a log file, allowing the system to reconstruct the state of the database during recovery.

- **Checkpointing:**
Periodic snapshots of the database state used to minimize the amount of data that needs to be recovered during a failure.
- **Recovery techniques:**
- **Rollback:** Undoing the effects of a failed transaction by applying the corresponding log entries in reverse order.
- **Redolog:** Re-applying committed transaction updates after a system crash using the transaction log.
- **Concurrency control interaction:**
Recovery mechanisms must consider concurrent transactions and their locking schemes to ensure consistent data restoration.

Characterizing Schedules Based on Recoverability

Introduction

In the field of database systems, one of the most important concepts is that of recoverability. Recoverability refers to the ability of a system to restore its state in the event of failure. In order to ensure that a system is recoverable, it is necessary to understand the different types of schedules that can be used, and how they impact the recoverability of a system.

In this article, we will explore the various types of schedules that can be used in database systems, and how they impact the recoverability of a system. We will also provide examples of each type of schedule, and discuss the pros and cons of each.

Transactions and Schedules

Before diving into the different types of schedules, it is important to understand the concept of a transaction. In the context of database systems, a transaction is a series of operations that are executed as a single unit of work. These operations can include reading and writing data, and they must be atomic, consistent, isolated, and durable (ACID).

A schedule, on the other hand, is a sequence of transactions that are executed by a system. The order in which these transactions are executed can have a major impact on the recoverability of a system.

Explore our **latest online courses** and learn new skills at your own pace. Enroll and become a certified expert to boost your career.

Types of Schedules

Serial Schedule

A serial schedule is one in which all transactions are executed one at a time, in a specific order. This means that no two transactions can be executed simultaneously. This type of schedule is considered to be the most recoverable, as there is only one transaction executing at a time, and it is easy to determine the state of the system at any given point in time.

Example

Transaction 1: Read A, Write A

Transaction 2: Read B, Write B

Transaction 3: Read C, Write C

Parallel Schedule

A parallel schedule is one in which multiple transactions are executed simultaneously. This type of schedule is less recoverable than a serial schedule, as it can be more difficult to determine the state of the system at any given point in time.

Example

Transaction 1: Read A, Write A

Transaction 2: Read B, Write B

Transaction 3: Read C, Write C

Concurrent Schedule

A concurrent schedule is one in which multiple transactions are executed simultaneously, and their operations may overlap. This type of schedule is the least recoverable, as it can be very difficult to determine the state of the system at any given point in time.

Example

Transaction 1: Read A, Write A

Transaction 2: Read A, Write B

Transaction 3: Read B, Write C

Recoverability

As mentioned earlier, recoverability refers to the ability of a system to restore its state in the event of a failure. The recoverability of a system is directly impacted by the type of schedule that is used.

A serial schedule is considered to be the most recoverable, as there is only one transaction executing at a time, and it is easy to determine the state of the system at any given point in time.

A parallel schedule is less recoverable than a serial schedule, as it can be more difficult to determine the state of the system at any given point in time.

A concurrent schedule is the least recoverable, as it can be very difficult to determine the state of the system at any given point in time.

Real-Life Examples

Online Retail

An online retail store would typically use a concurrent schedule, as multiple customers can be browsing and making purchases at the same time. This type of schedule is necessary in order to handle the high volume of transactions that occur during peak shopping times. However, it also means that the system must be designed to handle conflicts and ensure recoverability in case of failure.

Banking

A banking system would typically use a serial schedule, as transactions must be processed in a specific order to ensure the integrity of the data. For example, a transfer of funds from one account to another must be processed in a specific order to ensure that the funds are properly deducted from the source account and added to the destination account. This type of schedule is necessary to ensure that the system is recoverable in case of failure.

Airline Reservation

An airline reservation system would typically use a parallel schedule, as multiple customers can be booking flights at the same time. This type of schedule is necessary in order to handle the high volume of transactions that occur during peak travel times. However, it also means

that the system must be designed to handle conflicts and ensure recoverability in case of failure.

Distributed Systems

In distributed systems, different nodes may execute transactions concurrently and their execution order may not be the same across all nodes. This leads to the possibility of conflicting transactions and the need for more advanced techniques for recoverability, such as distributed transactions and two-phase commit protocols. Distributed systems often rely on parallel schedules to handle high-volume transactions, but they also need to be designed with consideration for how to handle conflicts and ensure recoverability.

Recovery Techniques

Recoverability in database systems can be achieved through various techniques such as database backups, replication, and log-based recovery. Database backups allow for the system to be restored to a previous state in case of failure, while replication allows for multiple copies of the database to be maintained for failover. Log-based recovery uses a log of all the transactions to undo or redo actions in case of failure.

Characterizing Schedules Based on Serializability

Introduction

In the field of database management systems, concurrency control is an important aspect of maintaining the consistency of data. One of the most widely used methods for ensuring consistency is serializability, which is a concept that characterizes schedules based on their ability to produce the same results as if the transactions were executed one at a time in some order. In this article, we will explore the concept of serializability in scheduling and how it can be used to ensure consistency in a concurrent system.

What is Serializability?

Serializability is a concept that is used to ensure that the concurrent execution of multiple transactions does not result in inconsistencies or conflicts in a database management system. In other words, it ensures that the results of concurrent execution of transactions are the same as if the transactions were executed one at a time in some order.

A schedule is considered to be serializable if it is equivalent to some serial schedule, which is a schedule where all transactions are executed one at a time. This means that if a schedule is serializable, it does not result in any inconsistencies or conflicts in the database.

Explore our **latest online courses** and learn new skills at your own pace. Enroll and become a certified expert to boost your career.

Types of Schedules

There are two types of schedules: serial schedules and concurrent schedules. A serial schedule is one where all transactions are executed one at a time, and a concurrent schedule is one where multiple transactions are executed simultaneously.

A schedule is considered to be conflict serializable if it is equivalent to some serial schedule. A schedule is considered to be view serializable if it is equivalent to some serial schedule, but the order of the transactions may be different.

Testing for Serializability

There are several methods for testing if a schedule is serializable, including –

- **Conflict serializability** – A schedule is conflict serializable if it is equivalent to some serial schedule and does not contain any conflicting operations.
- **View serializability** – A schedule is a view serializable if it is equivalent to some serial schedule, but the order of the transactions may be different.

To check for conflict serializability, we can use the conflict graph method which involves creating a graph where each transaction is represented by a node and each conflicting operation is represented by an edge. A schedule is considered to be conflict serializable if there are no cycles in the graph.

To check for view serializability, we can use the view equivalence method which involves comparing the results of a schedule with the results of a serial schedule. If the results are the same, the schedule is considered to be view serializable.

Concurrency Control Techniques

There are several techniques that can be used to ensure that a schedule is serializable, including –

- **Locking** – This technique involves placing locks on specific data items to prevent conflicting operations from being executed simultaneously.
- **Time stamping** – This technique involves assigning a unique time stamp to each transaction, and transactions are executed in order of their time stamp.
- **Optimistic concurrency control** – This technique involves allowing transactions to be executed concurrently, but checking for conflicts at commit time.

Example

```
BEGIN TRANSACTION
  READ(A)
  READ(B)
  A = A + 1
  B = B + 1
  WRITE(A)
  WRITE(B)
COMMIT
```

In the above example, we have two transactions T1 and T2 that are executed concurrently. T1 reads the value of A and B, increments them by 1, and writes the new values back to the database. T2 also reads the value of A and B, increments them by 1, and writes the new values back to the database.

If T1 and T2 are executed in a serial schedule, the final values of A and B will be 2. But if they are executed concurrently, there is a possibility of a conflict as both T1 and T2 are trying to update the same values simultaneously. This can result in an inconsistent state.

To prevent this type of conflict, we can use concurrency control techniques like locking or time stamping. For example, if we use locking, T1 would acquire a lock on A and B before reading and updating them, and T2 would have to wait until T1 releases the lock before it can acquire the lock and execute its operations. This ensures that T1 and T2 are not executing their operations on the same data simultaneously and thus avoiding conflict.

Another example of serializability in real life would be in an e-commerce platform. Two customers, A, and B are trying to purchase the last available item in a store. If the transactions were executed in a serial schedule, customer A would purchase the item and customer B would be notified that the item is out of stock.

But if the transactions were executed concurrently, there is a possibility of a conflict as both A and B are trying to purchase the same item simultaneously. To prevent this type of conflict,

we can use a technique like time stamping, where the first customer to click the “purchase” button gets the item and the other customer is notified that the item is no longer available.

Two Phase Locking Protocol

-

The Two-Phase Locking (2PL) Protocol is an essential concept in database management systems used to maintain data consistency and ensure smooth operation when multiple transactions are happening simultaneously. It helps to prevent issues like data conflicts where two or more transactions try to access or modify the same data at the same time, potentially causing errors.

Two-Phase Locking is widely used to ensure serializability, meaning transactions occur in a sequence that maintains data accuracy. This article will explore the workings of the 2PL protocol, its types, advantages and its role in maintaining a reliable database system

Two Phase Locking

The Two-Phase Locking (2PL) Protocol is a key technique used in database management systems to manage how multiple transactions access and modify data at the same time. When many users or processes interact with a database, it's important to ensure that data remains consistent and error-free. Without proper management, issues like data conflicts or corruption can occur if two transactions try to use the same data simultaneously.

The Two-Phase Locking Protocol resolves this issue by defining clear rules for managing data locks. It divides a transaction into two phases:

1. **Growing Phase:** In this step, the transaction gathers all the locks it needs to access the required data. During this phase, it cannot release any locks.
2. **Shrinking Phase:** Once a transaction starts releasing locks, it cannot acquire any new ones. This ensures that no other transaction interferes with the ongoing process.

Types of Lock

Shared Lock (S): Shared Lock is also called a read-only lock, allows multiple transactions to access the same data item for reading at the same time. However, transactions with this lock cannot make changes to the data. A shared lock is requested using the lock-S instruction.

Exclusive Lock (X): An Exclusive Lock allows a transaction to both read and modify a data item. This lock is exclusive, meaning no other transaction can access the same data item while this lock is held. An exclusive lock is requested using the lock-X instruction.

Read more about the [Locks](#).

Lock Conversions

In the Two-Phase Locking Protocol, lock conversion means changing the type of lock on data while a transaction is happening. This process is carefully controlled to maintain consistency in the database.

1. **Upgrading a Lock:** This means changing a shared lock (S) to an exclusive lock (X). For example, if a transaction initially only needs to read data (S) but later decides it needs to update the same data, it can request an upgrade to an exclusive lock (X). However, this can only happen during the Growing Phase, where the transaction is still acquiring locks.
 - Example: A transaction reads a value (S lock) but then realizes it needs to modify the value. It upgrades to an X lock during the Growing Phase.

2. **Downgrading a Lock:** This means changing an exclusive lock (X) to a shared lock (S). For instance, if a transaction initially planned to modify data (X lock) but later decides it only needs to read it, it can downgrade the lock. However, this must happen during the Shrinking Phase, where the transaction is releasing locks.
 - Example: A transaction modifies a value (X lock) but later only needs to read the value, so it downgrades to an S lock during the Shrinking Phase.

ules ensure that the Two-Phase Locking Protocol maintains consistency and avoids conflicts between transactions. By limiting when upgrades and downgrades can occur, the system prevents situations where multiple transactions interfere with each other's operations.

Read more about [Shared and Exclusive Locks](#).

Let's see a transaction implementing 2-PL.

	T1	T2
1	lock-S(A)	
2		lock-S(A)
3	lock-X(B)	
4		
5	Unlock(A)	
6		Lock-X(C)
7	Unlock(B)	
8		Unlock(A)
9		Unlock(C)
10		

This is a basic outline of a transaction that demonstrates how locking and unlocking work in the Two-Phase Locking Protocol (2PL).

Transaction T1

- The growing Phase is from steps 1-3
- The shrinking Phase is from steps 5-7
- Lock Point at 3

Transaction T2

- The growing Phase is from steps 2-6
- The shrinking Phase is from steps 8-9
- Lock Point at 6

Lock Point

The lock point in a transaction is the moment when the transaction finishes acquiring all the locks it needs. After this point, no new locks can be added, and the transaction starts releasing locks. It's a key step in the Two-Phase Locking Protocol to ensure the rules of growing and shrinking phases are followed.

Example of 2PL

Imagine a library system where multiple users can borrow or return books. Each action (like borrowing or returning) is treated as a transaction. Here's how the Two-Phase Locking Protocol (2PL) works, including the lock point:

User A wants to:

1. Check the availability of Book X.
2. Borrow Book X if it's available.
3. Update the library's record.

Growing Phase (Locks are Acquired):

1. User A locks Book X with a shared lock (S) to check its availability.
2. After confirming the book is available, User A upgrades the lock to an exclusive lock (X) to borrow it.
3. User A locks the library's record to update the borrowing details.

Lock Point: Once User A has acquired all the necessary locks (on Book X and the library record), the transaction reaches the lock point. No more locks can be acquired after this.

Shrinking Phase (Locks are Released):

1. User A updates the record and releases the lock on the library's record.
 2. User A finishes borrowing and releases the exclusive lock on Book X.
- This process ensures that no other user can interfere with Book X or the library record during the transaction, maintaining data accuracy and consistency. The lock point ensures that all locks are acquired before any are released, following the 2PL rules.

Drawbacks of 2PL

Two-phase locking (2PL) ensures that transactions are executed in the correct order by using two phases: acquiring and releasing locks. However, it has some drawbacks:

- **Deadlocks:** Transactions can get stuck waiting for each other's locks, causing them to freeze indefinitely.
- **Cascading Rollbacks:** If one transaction fails, others that depend on it might also fail leading to inefficiency and potential data issues.
- **Lock Contention:** Too many transactions competing for the same locks can slow down the system, especially when many users are working at the same time.

- **Limited Concurrency:** The strict rules of 2PL can reduce how many transactions can run at once, resulting in slower performance and longer wait times.

Cascading Rollbacks in 2-PL

Let's see the following Schedule:

Schedule

The image illustrates a transaction schedule using the Two-Phase Locking (2PL) protocol, showing the sequence of actions for three transactions T1, T2 and T3.

Key Points:

1. **Transaction T1:**
 - T1 acquires an exclusive lock (X) on data item A, performs a write operation on A and then acquires a shared lock (S) on B.
 - T1 reaches its lock point (LP) after acquiring all locks.
 - Eventually, T1 fails and a rollback is triggered, undoing its changes.
2. **Transaction T2:**
 - T2 reads A after T1 writes A. This is called a [dirty read](#) because T1's write is not committed yet.
 - When T1 rolls back, T2's operations become invalid, and it is also forced to rollback.
3. **Transaction T3:**
 - T3 reads A after T2 reads A. Since T2 depends on the uncommitted changes of T1, T3 indirectly relies on T1's changes.
 - When T1 rolls back, T3 is also forced to rollback even though it was not directly interacting with T1's operations.

Cascading Rollback Problem:

- Here, T2 and T3 both depend on uncommitted data from T1 (either directly or indirectly).
- When T1 fails and rolls back all dependent transactions (T2 and T3) must also rollback because their operations were based on invalid data.
- Cascading rollbacks waste system resources, reduce concurrency and lead to inefficiency.
- In large systems, this can significantly affect performance and make recovery more complex.

How to avoid it?

- **Strict 2PL:** Ensure that transactions release their locks only after they commit, preventing other transactions from accessing uncommitted changes.
- **Rigorous 2PL:** Extend strict 2PL to hold both read and write locks until the transaction commits, offering even stronger guarantees.

Deadlock in 2-PL

Consider this simple example. We have two transactions T1 and T2.

Schedule: Lock-X1(A) Lock-X2(B) Lock-X1(B) Lock-X2(A)

This sequence represents a locking scenario where two transactions, T1 and T2 are trying to lock two resources, A and B in a particular order. Here's what each step means:

1. **Lock-X1(A):**
Transaction T1 acquires an exclusive lock on resource A. This means T1 has full control over A and no other transaction can use it until T1 releases the lock.
2. **Lock-X2(B):**
Transaction T2 acquires an exclusive lock on resource B. Similarly, T2 now has full control over B and no other transaction can access B until T2 releases the lock.
3. **Lock-X1(B):**
Transaction T1 tries to acquire an exclusive lock on resource B but T2 already holds the lock on B. So, T1 must wait for T2 to release the lock.
4. **Lock-X2(A):**
At the same time, Transaction T2 tries to acquire an exclusive lock on resource A but T1 already holds the lock on A. So, T2 must wait for T1 to release the lock.

Handling Deadlocks

Deadlock is a situation where a process or a set of processes is blocked, waiting for some other resource that is held by some other waiting process. It is an undesirable state of the system. In other words, Deadlock is a critical situation in computing where a process, or a group of processes, becomes unable to proceed because each is waiting for a resource that is held by another process in the same group. This scenario leads to a complete standstill, rendering the affected processes inactive and the system inefficient.

Necessary Condition for a Deadlock

The following are the [four conditions that must hold simultaneously](#) for a deadlock to occur.

1. **Mutual Exclusion:** A resource can be used by only one process at a time. If another process requests for that resource, then the requesting process must be delayed until the resource has been released.
2. **Hold and wait:** Some processes must be holding some resources in the non-shareable mode and at the same time must be waiting to acquire some more resources, which are currently held by other processes in the non-shareable mode.
3. **No pre-emption:** Resources granted to a process can be released back to the system only as a result of voluntary action of that process after the process has completed its task.
4. **Circular wait:** Deadlocked processes are involved in a circular chain such that each process holds one or more resources being requested by the next process in the chain.

Methods of Handling Deadlocks

Understanding how to handle deadlocks is vital for mastering operating systems, a key area in GATE CS.

There are four approaches to dealing with deadlocks.

1. Deadlock Prevention
2. Deadlock avoidance (Banker's Algorithm)
3. Deadlock detection & recovery
4. Deadlock Ignorance (Ostrich Method)

These are explained below.

1. Deadlock Prevention

The strategy of [deadlock prevention](#) is to design the system in such a way that the possibility of deadlock is excluded. The indirect methods prevent the occurrence of one of three necessary conditions of deadlock i.e., mutual exclusion, no pre-emption, and hold and wait. The direct method prevents the occurrence of circular wait. **Prevention techniques -Mutual**

exclusion – are supported by the OS. **Hold and Wait** – the condition can be prevented by requiring that a process requests all its required resources at one time and blocking the process until all of its requests can be granted at the same time simultaneously. But this prevention does not yield good results because:

- long waiting time required
- inefficient use of allocated resource
- A process may not know all the required resources in advance

No pre-emption – techniques for ‘no pre-emption are’

- If a process that is holding some resource, requests another resource that can not be immediately allocated to it, all resources currently being held are released and if necessary, request again together with the additional resource.
- If a process requests a resource that is currently held by another process, the OS may pre-empt the second process and require it to release its resources. This works only if both processes do not have the same priority.

Circular wait One way to ensure that this condition never holds is to impose a total ordering of all resource types and to require that each process requests resources in increasing order of enumeration, i.e., if a process has been allocated resources of type R, then it may subsequently request only those resources of types following R in ordering.

2. Deadlock Avoidance

The deadlock avoidance Algorithm works by proactively looking for potential deadlock situations before they occur. It does this by tracking the resource usage of each process and identifying conflicts that could potentially lead to a deadlock. If a potential deadlock is identified, the algorithm will take steps to resolve the conflict, such as rolling back one of the processes or pre-emptively allocating resources to other processes. The Deadlock Avoidance Algorithm is designed to minimize the chances of a deadlock occurring, although it cannot guarantee that a deadlock will never occur. This approach allows the three necessary conditions of deadlock but makes judicious choices to assure that the deadlock point is never reached. It allows more concurrency than avoidance detection A decision is made dynamically whether the current resource allocation request will, if granted, potentially lead to deadlock. It requires knowledge of future process requests. Two techniques to avoid deadlock :

1. Process initiation denial
2. Resource allocation denial

Advantages

- Not necessary to pre-empt and rollback processes
- Less restrictive than deadlock prevention

Disadvantages

- Future resource requirements must be known in advance
- Processes can be blocked for long periods
- Exists a fixed number of resources for allocation

Multiple Granularity Locking in DBMS

The various Concurrency Control schemes have used different methods and every individual Data item is the unit on which synchronization is performed. A certain drawback of this technique is if a transaction T_i needs to access the entire database, and a locking protocol is used, then T_i must lock each item in the database. It is less efficient, it would be simpler if T_i could use a single lock to lock the entire database. But, if it considers the second proposal, this should not in fact overlook certain flaws in the proposed method. Suppose another transaction just needs to access a few data items from a database, so locking the entire database seems to be unnecessary moreover it may cost us a loss of Concurrency, which was our primary goal in the first place. To bargain between Efficiency and Concurrency. Use Granularity.

Let's start by understanding what is meant by Granularity.

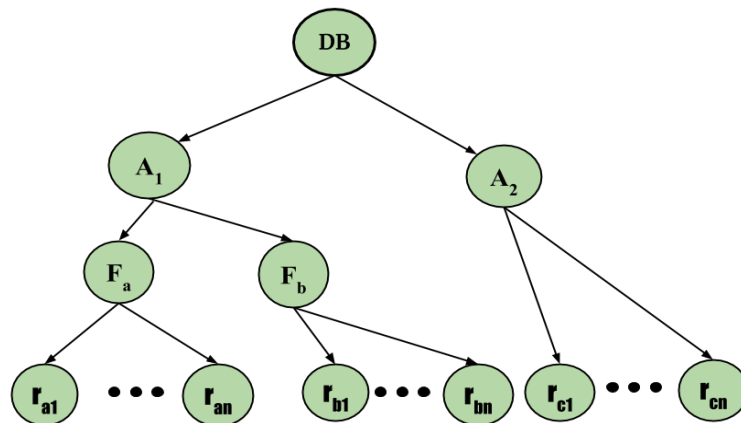
Granularity

It is the size of the data item allowed to lock. Now *Multiple Granularity* means hierarchically breaking up the database into blocks that can be locked and can be tracked what needs to lock and in what fashion. Such a hierarchy can be represented graphically as a tree.

For example, consider the tree, which consists of four levels of nodes. The highest level represents the entire database. Below it is nodes of type area; the database consists of exactly these areas. The area has children nodes which are called files. Every area has those files that are its child nodes. No file can span more than one area.

Finally, each file has child nodes called records. As before, the file consists of exactly those records that are its child nodes, and no record can be present in more than one file. Hence, the levels starting from the top level are:

- database
- area
- file
- record



Multi Granularity tree Hiererchy

Consider the above diagram for the example given, each node in the tree can be locked individually. As in the [2-phase locking protocol](#), it shall use shared and exclusive lock modes. When a transaction locks a node, in either shared or exclusive mode, the transaction also implicitly locks all the descendants of that node in the same lock mode. For example, if transaction T_i gets an explicit lock on file F_c in exclusive mode, then it has an implicit lock in exclusive mode on all the records belonging to that file. It does not need to lock the

individual records of F_c explicitly. this is the main difference between Tree-Based [Locking](#) and Hierarchical locking for multiple granularities.

Now, with locks on files and records made simple, how does the system determine if the root node can be locked? One possibility is for it to search the entire tree but the solution nullifies the whole purpose of the multiple-granularity locking scheme. A more efficient way to gain this knowledge is to introduce a new lock mode, called *Intention lock mode*.

Intention Mode Lock

In addition to **S** and **X** lock modes, there are three additional lock modes with multiple granularities:

- **Intention-Shared (IS)**: explicit locking at a lower level of the tree but only with shared locks.
- **Intention-Exclusive (IX)**: explicit locking at a lower level with exclusive or shared locks.
- **Shared & Intention-Exclusive (SIX)**: the subtree rooted by that node is locked explicitly in shared mode and explicit locking is being done at a lower level with exclusive mode locks.

The compatibility matrix for these lock modes are described below:

	IS	IX	S	SIX	X
IS	✓	✓	✓	✓	✗
IX	✓	✓	✗	✗	✗
S	✓	✗	✓	✗	✗
SIX	✓	✗	✗	✗	✗
X	✗	✗	✗	✗	✗

IS : Intention Shared
IX : Intention Exclusive
S : Shared

X : Exclusive
SIX : Shared & Intention Exclusive

Multi Granularity tree Hierarchy

The multiple-granularity locking protocol uses the intention lock modes to ensure serializability. It requires that a transaction T_i that attempts to lock a node must follow these protocols:

- Transaction T_i must follow the lock-compatibility matrix.
- Transaction T_i must lock the root of the tree first, and it can lock it in any mode.
- Transaction T_i can lock a node in S or IS mode only if T_i currently has the parent of the node-locked in either IX or IS mode.
- Transaction T_i can lock a node in X, SIX, or IX mode only if T_i currently has the parent of the node-locked in either IX or SIX modes.
- Transaction T_i can lock a node only if T_i has not previously unlocked any node (i.e., T_i is two-phase).

- Transaction T_i can unlock a node only if T_i currently has none of the children of the node-locked.

Observe that the multiple-granularity protocol requires that locks be acquired in top-down (root-to-leaf) order, whereas locks must be released in bottom-up (leaf to-root) order. As an illustration of the protocol, consider the tree given above and the transactions:

- Say transaction T_1 reads record R_{a2} in file F_a . Then, T_1 needs to lock the database, area A_1 , and F_a in IS mode (and in that order), and finally to lock R_{a2} in S mode.
- Say transaction T_2 modifies record R_{a9} in file F_a . Then, T_2 needs to lock the database, area A_1 , and file F_a (and in that order) in IX mode, and at last to lock R_{a9} in X mode.
- Say transaction T_3 reads all the records in file F_a . Then, T_3 needs to lock the database and area A_1 (and in that order) in IS mode, and at last to lock F_a in S mode.
- Say transaction T_4 reads the entire database. It can do so after locking the database in S mode.

Note that transactions T_1 , T_3 , and T_4 can access the database concurrently. Transaction T_2 can execute concurrently with T_1 , but not with either T_3 or T_4 . This protocol enhances concurrency and reduces lock overhead. Deadlock is still possible in the multiple-granularity protocol, as it is in the two-phase locking protocol. These can be eliminated by using certain [deadlock elimination techniques](#).

Database Recovery Techniques in DBMS

•

Database Systems like any other computer system, are subject to failures but the data stored in them must be available as and when required. When a database fails it must possess the facilities for fast recovery. It must also have atomicity i.e. either transactions are completed successfully and committed (the effect is recorded permanently in the database) or the transaction should have no effect on the database.

Types of Recovery Techniques in DBMS

Database recovery techniques are used in database management systems (DBMS) to restore a database to a consistent state after a failure or error has occurred. The main goal of recovery techniques is to ensure data integrity and consistency and prevent data loss.

There are mainly two types of recovery techniques used in DBMS

- **Rollback/Undo Recovery Technique**
- **Commit/Redo Recovery Technique**
- **CheckPoint Recovery Technique**

Database recovery techniques ensure data integrity in case of system failures. Understanding how these techniques work is crucial for managing databases effectively. The [GATE CS Self-Paced Course](#) covers recovery strategies in DBMS, providing practical insights into maintaining data consistency

Rollback/Undo Recovery Technique

The rollback/undo recovery technique is based on the principle of backing out or undoing the effects of a transaction that has not been completed successfully due to a system failure or error. This technique is accomplished by undoing the changes made by the transaction using the log records stored in the transaction log. The transaction log contains a record of all the transactions that have been performed on the database. The system uses the log records to undo the changes made by the failed transaction and restore the database to its previous state.

Commit/Redo Recovery Technique

The commit/redo recovery technique is based on the principle of reapplying the changes made by a transaction that has been completed successfully to the database. This technique is accomplished by using the log records stored in the transaction log to redo the changes made by the transaction that was in progress at the time of the failure or error. The system uses the log records to reapply the changes made by the transaction and restore the database to its most recent consistent state.

Checkpoint Recovery Technique

Checkpoint Recovery is a technique used to improve data integrity and system stability, especially in databases and distributed systems. It entails preserving the system's state at regular intervals, known as checkpoints, at which all ongoing transactions are either completed or not initiated. This saved state, which includes memory and CPU registers, is kept in stable, non-volatile storage so that it can withstand system crashes. In the event of a breakdown, the system can be restored to the most recent checkpoint, which reduces data loss and downtime. The frequency of checkpoint formation is carefully regulated to decrease system overhead while ensuring that recent data may be restored quickly.

Overall, recovery techniques are essential to ensure data consistency and availability in Database Management System, and each technique has its own advantages and limitations that must be considered in the design of a recovery system.

Database Systems

There are both automatic and non-automatic ways for both, backing up data and recovery from any failure situations. The techniques used to recover lost data due to system crashes, transaction errors, viruses, catastrophic failure, incorrect command execution, etc. are database recovery techniques. So to prevent data loss recovery techniques based on deferred updates and immediate updates or backing up data can be used. Recovery techniques are heavily dependent upon the existence of a special file known as a **system log**. It contains information about the start and end of each transaction and any updates which occur during the **transaction**. The log keeps track of all transaction operations that affect the values of database items. This information is needed to recover from transaction failure.

- **The log is kept on disk start_transaction(T):** This log entry records that transaction T starts the execution.
- **read_item(T, X):** This log entry records that transaction T reads the value of database item X.
- **write_item(T, X, old_value, new_value):** This log entry records that transaction T changes the value of the database item X from old_value to new_value. The old value is sometimes known as a before an image of X, and the new value is known as an afterimage of X.
- **commit(T):** This log entry records that transaction T has completed all accesses to the database successfully and its effect can be committed (recorded permanently) to the database.
- **abort(T):** This records that transaction T has been aborted.
- **checkpoint:** A checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in a consistent state, and all the transactions were committed.

A transaction T reaches its **commit** point when all its operations that access the database have been executed successfully i.e. the transaction has reached the point at which it will not **abort** (terminate without completing). Once committed, the transaction is permanently

recorded in the database. Commitment always involves writing a commit entry to the log and writing the log to disk. At the time of a system crash, the item is searched back in the log for all transactions T that have written a start_transaction(T) entry into the log but have not written a commit(T) entry yet; these transactions may have to be rolled back to undo their effect on the database during the recovery process.

- **Undoing:** If a transaction crashes, then the recovery manager may undo transactions i.e. reverse the operations of a transaction. This involves examining a transaction for the log entry write_item(T, x, old_value, new_value) and setting the value of item x in the database to old-value. There are two major techniques for recovery from non-catastrophic transaction failures: [deferred updates and immediate updates](#).
- **Deferred Update:** This technique does not physically update the database on disk until a transaction has reached its commit point. Before reaching commit, all transaction updates are recorded in the local transaction workspace. If a transaction fails before reaching its commit point, it will not have changed the database in any way so UNDO is not needed. It may be necessary to REDO the effect of the operations that are recorded in the local transaction workspace, because their effect may not yet have been written in the database. Hence, a deferred update is also known as the **No-undo/redo algorithm**.
- **Immediate Update:** In the immediate update, the database may be updated by some operations of a transaction before the transaction reaches its commit point. However, these operations are recorded in a log on disk before they are applied to the database, making recovery still possible. If a transaction fails to reach its commit point, the effect of its operation must be undone i.e. the transaction must be rolled back hence we require both undo and redo. This technique is known as **undo/redo algorithm**.
- **Caching/Buffering:** In this one or more disk pages that include data items to be updated are cached into main memory buffers and then updated in memory before being written back to disk. A collection of in-memory buffers called the DBMS cache is kept under the control of DBMS for holding these buffers. A directory is used to keep track of which database items are in the buffer. A dirty bit is associated with each buffer, which is 0 if the buffer is not modified else 1 if modified.
- **Shadow Paging:** It provides atomicity and durability. A directory with n entries is constructed, where the ith entry points to the ith database page on the link. When a transaction began executing the current directory is copied into a shadow directory. When a page is to be modified, a shadow page is allocated in which changes are made and when it is ready to become durable, all pages that refer to the original are updated to refer new replacement page.
- **Backward Recovery:** The term "Rollback" and "UNDO" can also refer to backward recovery. When a backup of the data is not available and previous modifications need to be undone, this technique can be helpful. With the backward recovery method, unused modifications are removed and the database is returned to its prior condition. All adjustments made during the previous transaction are reversed during the backward recovery. In other words, it reprocesses valid transactions and undoes the erroneous database updates.
- **Forward Recovery:** "Roll forward" and "REDO" refers to forwarding recovery. When a database needs to be updated with all changes verified, this forward recovery technique is helpful. Some failed transactions in this database are applied to the database to roll those modifications forward. In other words, the database is restored using preserved data and valid transactions counted by their past saves.

Backup Techniques

There are different types of Backup Techniques. Some of them are listed below.

- **Full database Backup:** In this full database including data and database, Meta information needed to restore the whole database, including full-text catalogs are backed up in a predefined time series.
- **Differential Backup:** It stores only the data changes that have occurred since the last full database backup. When some data has changed many times since the last full database backup, a differential backup stores the most recent version of the changed data. For this first, we need to restore a full database backup.
- **Transaction Log Backup:** In this, all events that have occurred in the database, like a record of every single statement executed is backed up. It is the backup of transaction log entries and contains all transactions that had happened to the database. Through this, the database can be recovered to a specific point in time. It is even possible to perform a backup from a transaction log if the data files are destroyed and not even a single committed transaction is lost.

Unit 1 EM

Different environmental problems - Energy and the environment

Energy and the environment are closely intertwined, and various environmental problems arise from the production, distribution, and consumption of energy. Here are some key environmental issues related to energy:

1. **Climate Change:** The burning of fossil fuels for energy releases greenhouse gases, such as carbon dioxide, into the atmosphere. This contributes to global warming and climate change, leading to rising temperatures, sea level rise, extreme weather events, and disruptions to ecosystems.
2. **Air Pollution:** The combustion of fossil fuels and certain industrial processes release pollutants into the air, contributing to air pollution. Common pollutants include sulfur dioxide, nitrogen oxides, particulate matter, and volatile organic compounds, which can have adverse effects on human health and the environment.
3. **Water Pollution:** Energy production and extraction processes can contaminate water sources. For example, oil and gas drilling, as well as coal mining, may result in the release of pollutants into water bodies, affecting aquatic ecosystems and potentially harming human health.
4. **Land Use and Habitat Destruction:** Large-scale energy infrastructure projects, such as dams, nuclear power plants, and solar farms, can lead to habitat destruction and alteration. This can result in the displacement of wildlife and fragmentation of ecosystems, impacting biodiversity.
5. **Resource Depletion:** The extraction of natural resources for energy production, such as coal, oil, and gas, can lead to resource depletion and ecosystem degradation. Non renewable resources, in particular, are finite and can contribute to environmental degradation.
6. **Waste Generation:** The production and use of energy often generate significant amounts of waste, including hazardous materials and radioactive waste from nuclear power plants. Improper disposal and management of these wastes can pose serious environmental and health risks.
7. **Deforestation:** Biomass energy production, such as the burning of wood for fuel, can contribute to deforestation if not managed sustainably. Deforestation has wide-ranging impacts on ecosystems, biodiversity, and carbon sequestration.
8. **Nuclear Accidents:** The use of nuclear energy carries the risk of accidents, such as meltdowns or radioactive leaks, which can have severe and long-lasting environmental consequences. Chernobyl and Fukushima are examples of major nuclear incidents with significant environmental impacts.
9. **Renewable Energy Challenges:** While renewable energy sources like solar and wind are

considered cleaner, their widespread adoption also poses environmental challenges. These include the environmental impact of manufacturing and disposing of solar panels and wind turbines, as well as the potential alteration of landscapes for large-scale renewable energy projects.

Agriculture and the environment, the atmosphere and human activities, etc. Need for environmental management:-

Agriculture, human activities, and the atmosphere are interconnected elements of the environment, and their interactions can have significant environmental implications. The need for environmental management becomes crucial to address various challenges and ensure the sustainable coexistence of human activities with the environment. Here are some key aspects:

1. Agriculture and Environmental Impact:

- **Land Use Change:** Expansion of agricultural land often leads to deforestation and habitat loss, impacting biodiversity and ecosystem services.
- **Water Usage and Pollution:** Agricultural practices, including irrigation and the use of fertilizers and pesticides, can lead to water scarcity and contamination, affecting aquatic ecosystems and human health.
- **Soil Degradation:** Intensive farming practices may lead to soil erosion, depletion of nutrients, and loss of soil fertility, contributing to desertification and degradation.

2. Atmosphere and Human Activities:

- **Greenhouse Gas Emissions:** Human activities, such as the burning of fossil fuels for energy and industrial processes, contribute to the release of greenhouse gases, leading to climate change and global warming.
- **Air Pollution:** Industrial emissions, transportation, and other human activities release pollutants into the air, impacting air quality and human health.

3. Need for Environmental Management:

- **Biodiversity Conservation:** Environmental management is essential for protecting and conserving biodiversity, ensuring the stability and resilience of ecosystems and the services they provide.
- **Climate Change Mitigation:** Efforts to reduce greenhouse gas emissions and adapt to climate change are crucial components of environmental management to mitigate the impacts of climate change.
- **Resource Conservation:** Sustainable management practices are necessary to prevent overexploitation of natural resources, promoting responsible use and conservation for future generations.
- **Pollution Control:** Environmental management strategies aim to control and

minimize pollution through regulations, monitoring, and the promotion of cleaner technologies.

- **Land Use Planning:** Proper land use planning helps prevent habitat destruction, deforestation, and urban sprawl, ensuring sustainable development that balances human needs with environmental preservation.
- **Water Resource Management:** Effective management of water resources involves sustainable practices in agriculture, industry, and domestic use to prevent water scarcity, contamination, and depletion.
- **Waste Management:** Proper waste management, including recycling, reuse, and proper disposal, is crucial to minimize the environmental impact of human activities and reduce pollution.
- **Environmental Education and Awareness:** Promoting environmental education and awareness is essential for fostering a sense of responsibility and encouraging sustainable behaviors among individuals and communities.

Environmental impact analysis

Environmental Impact Analysis (EIA) is a process used to evaluate the potential environmental consequences of a proposed project, plan, or program. It involves a systematic and comprehensive examination of the potential impacts on the environment, both positive and negative. Here are the various steps involved in environmental impact analysis:

1. Project Identification:

- Define the purpose and need for the project.
- Clearly outline the project scope, objectives, and components.

2. Screening:

- Evaluate the project to determine if it requires a full EIA.
- Identify and exclude projects with minimal or no significant environmental impact.

3. Scoping:

- Define the boundaries of the study area.
- Identify potential environmental impacts and stakeholders.
- Establish the scope of the analysis and the key issues to be addressed.

4. Baseline Data Collection:

- Gather information on the existing environmental conditions in the study area. • Collect data on air quality, water quality, soil conditions, biodiversity, socio economic factors, and other relevant parameters.

5. Identification and Prediction of Impacts:

- Assess the potential impacts of the project on the environment.
- Consider direct and indirect effects, cumulative impacts, and long-term consequences.
- Use models, simulations, or expert judgment to predict impacts.

6. Assessment of Alternatives:

- Evaluate alternative project designs, locations, or technologies.

- Compare the environmental impacts of different options.
7. **Mitigation Measures:**
 - Propose measures to avoid, minimize, or compensate for adverse impacts.
 - Develop strategies for enhancing positive impacts.
 8. **Drafting the Environmental Impact Statement (EIS):**
 - Prepare a comprehensive document summarizing the analysis and findings.
 - Include information on the project, alternatives, environmental impacts, and proposed mitigation measures.
 9. **Public Consultation:**
 - Share the EIS with the public and stakeholders for feedback.
 - Address concerns and incorporate relevant suggestions.
 10. **Review and Approval:**
 - Regulatory authorities review the EIS to ensure compliance with environmental regulations.
 - Decision-makers assess the information and approve or deny the project.
 11. **Monitoring and Enforcement:**
 - Establish a monitoring plan to track actual impacts during project implementation.
 - Enforce mitigation measures and take corrective actions if necessary.
 12. **Post-Implementation Evaluation:**
 - Assess the actual environmental impacts against the predictions.
 - Evaluate the effectiveness of mitigation measures.

The nature, scope and components of environmental management:-

Environmental management is a systematic process that involves planning, implementing, monitoring, and regulating activities to ensure the sustainable use of natural resources, protect ecosystems, and minimize adverse impacts on the environment. The nature, scope, and components of environmental management encompass a broad range of activities and considerations. Here's an overview:

1. Nature of Environmental Management:

- **Holistic Approach:** Environmental management takes a holistic approach, recognizing the interconnectedness of various environmental components. It considers ecosystems, biodiversity, air and water quality, and human activities as interrelated elements.
- **Sustainability:** The core principle of environmental management is to promote sustainability, ensuring that present and future generations can meet their needs without compromising the ability of the environment to support life.

2. Scope of Environmental Management:

- **Natural Resource Management:** Environmental management addresses the sustainable use and conservation of natural resources, including land, water, forests, and biodiversity.
- **Pollution Control:** It involves strategies to control and reduce pollution, covering air, water, and soil pollution from industrial, agricultural, and urban sources.
- **Climate Change Mitigation and Adaptation:** Environmental management addresses efforts to mitigate greenhouse gas emissions, as well as strategies to adapt

to the impacts of climate change.

- **Waste Management:** Proper handling, recycling, and disposal of solid and hazardous waste are integral parts of environmental management.
- **Land Use Planning:** Planning and regulating land use to prevent habitat destruction, preserve ecosystems, and promote sustainable development are key aspects.
- **Environmental Impact Assessment (EIA):** Conducting EIAs for proposed projects helps evaluate potential environmental impacts before implementation, allowing for informed decision-making.

3. Components of Environmental Management:

- **Policy and Legislation:** Development and implementation of environmental policies and regulations at local, national, and international levels provide the legal framework for environmental management.
- **Institutional Framework:** Establishing governmental and non-governmental institutions responsible for environmental management, including environmental protection agencies and regulatory bodies.
- **Planning and Strategy:** Developing comprehensive environmental plans and strategies that outline goals, objectives, and action plans to address specific environmental challenges.
- **Monitoring and Assessment:** Regular monitoring of environmental parameters, conducting environmental impact assessments, and assessing the effectiveness of management measures.
- **Stakeholder Engagement:** Involving various stakeholders, including communities, industries, and non-profit organizations, in decision-making processes and fostering a sense of shared responsibility.
- **Education and Awareness:** Promoting environmental education and awareness to inform and engage the public in sustainable practices and behaviors.
- **Technology and Innovation:** Encouraging the development and adoption of environmentally friendly technologies and innovative solutions to address environmental challenges.

Scope and components of Environmental Management

Environmental Management refers to the systematic approach and practices employed by individuals, organizations, or governments to address environmental concerns and achieve sustainability goals (aim to transform our world. They are a call to action to end poverty and inequality, protect the planet, and ensure that all people enjoy health, justice and prosperity.). The scope of environmental management is broad and encompasses (hold) various components to ensure effective and responsible use of natural resources while minimizing negative impacts on the environment. Here's an explanation of the scope and components of Environmental Management:

Scope of Environmental Management:

1. Natural Resource Management:

- **Land Management:** Focuses on sustainable (able to be maintained at a certain rate or level) land use, soil conservation (prevention of [wasteful](#) use of a resource), and preventing land degradation.
 - **Water Management:** Involves efficient water use, conservation, and protection of water quality.
 - **Forest Management:** Aims at sustainable forestry practices, conservation of biodiversity, and prevention of deforestation.
2. **Waste Management:**
 - **Waste Reduction:** Strategies to minimize the generation of waste at its source.
 - **Waste Treatment and Disposal:** Proper handling, treatment, and disposal of various types of waste to prevent environmental pollution.
 3. **Air Quality Management:**
 - **Emission Control:** Implementing measures to reduce air pollutants from industrial processes, vehicles, and other sources.
 - **Air Quality Monitoring:** Regular monitoring of air quality to identify and address potential sources of pollution.
 4. **Biodiversity Conservation:**
 - **Habitat Protection:** Preserving natural habitats (A natural habitat is an ecological or environmental area where a specific species lives.) to support diverse ecosystems.
 - **Species Conservation:** Measures to protect endangered species and promote biodiversity.
 5. **Environmental Impact Assessment (EIA):**
 - **Project Evaluation:** Assessing the potential environmental impacts of proposed projects before implementation.
 - **Mitigation (Preventive measure) Planning:** Identifying and planning measures to minimize or offset negative environmental impacts.
 6. **Policy and Compliance:**
 - **Environmental Legislation (Court rules-):** Ensuring compliance with local and international environmental laws and regulations.
 - **Policy Development:** Developing and implementing internal policies to guide environmental performance.

Components of Environmental Management:

1. **Environmental Planning:**
 - Developing comprehensive plans to integrate environmental considerations into decision-making processes.
2. **Environmental Monitoring:**
 - Regularly assessing and analyzing environmental parameters to track changes and identify potential issues.
3. **Environmental Auditing:**
 - Systematic examination of an organization's environmental performance to ensure compliance with regulations and identify areas for improvement.
4. **Environmental Education and Awareness:**

- Promoting understanding and awareness of environmental issues among stakeholders, employees, and the public.
- 5. Green Practices and Technologies:**
 - Adoption of environmentally friendly practices and technologies to reduce resource consumption and pollution.
- 6. Corporate Social Responsibility (CSR):**
 - Integrating environmental responsibility into corporate strategies and practices. • (Companies that practice corporate social responsibility measure their success not only by the profits they generate, but also by their impact on people, communities, and the environment.)
- 7. Emergency Response Planning:**
 - Developing plans to address and mitigate the environmental impact of emergencies, such as spills or accidents.
- 8. Stakeholder Engagement:**
 - Involving and communicating with various stakeholders, including communities, NGOs, and government agencies.
- 9. Sustainable Development:**
 - Balancing economic, social, and environmental considerations to ensure long term sustainability.
- 10. Energy Management:**
 - Implementing energy-efficient practices and exploring renewable energy sources to reduce the environmental footprint.

Need of public participation in EIA

Public participation in Environmental Impact Assessment (EIA) is crucial for several reasons, as it enhances the overall effectiveness and legitimacy of the environmental decision-making process. Here are some key reasons why public participation is considered essential in EIA:

1. Informed Decision-Making:

- Public participation ensures that affected communities and stakeholders have access to information about proposed projects and their potential environmental impacts.
- It allows individuals to voice their concerns, share local knowledge, and contribute valuable insights that may not be captured through expert assessments alone.

2. Transparency and Accountability:

- Public participation promotes transparency in decision-making processes. When the public is involved, decision-makers are more accountable for their choices.
- It helps build trust between the authorities, project proponents, and the public by demonstrating a commitment to openness and accountability.

3. Democratic Values:

- Involving the public aligns with democratic principles, giving people the right to participate in decisions that affect their environment and quality of life.
- EIA processes that incorporate public input are more likely to reflect a broad range of perspectives and interests, avoiding the concentration of power in the hands of a few.

4. Local Knowledge and Expertise:

- Local communities often possess valuable knowledge about their environments that may not be

fully captured in technical assessments.

- Public participation allows for the inclusion of this local knowledge, providing a more comprehensive understanding of the potential impacts and risks associated with a project. 5.

Enhanced Decision Quality:

- Public input can lead to improved decision quality by identifying potential issues or alternatives that may have been overlooked in the initial assessment.
- A diverse range of perspectives helps decision-makers make more informed and balanced choices, leading to more sustainable outcomes.

6. Conflict Prevention and Resolution:

- Early and meaningful public participation can help identify and address concerns before conflicts arise.
- Involving the public in the decision-making process provides a platform for constructive dialogue and negotiation, helping to prevent or manage conflicts that may arise during project implementation.

7. Social Acceptance:

- Engaging the public in EIA processes increases the likelihood of project acceptance and social approval.

Environmental management

Environmental management is a comprehensive approach to address the interactions between human activities and the environment in order to achieve sustainable and responsible outcomes. It involves the planning, implementation, monitoring, and improvement of policies, practices, and processes to minimize negative impacts on the environment while promoting ecological integrity, social equity, and economic viability. Here is an in-depth explanation of key components of environmental management:

1. Policy Development:

- **Strategic Planning:** Establishing long-term environmental goals and objectives, often informed by legal requirements, scientific knowledge, and societal values.
- **Regulatory Framework:** Creating laws and regulations to govern environmental activities, such as pollution control, land use planning, and natural resource management.

2. Environmental Impact Assessment (EIA):

- **Identification of Impacts:** Evaluating potential environmental, social, and economic impacts of proposed projects, policies, or activities before they are undertaken.
- **Mitigation and Enhancement Measures:** Developing strategies to minimize adverse impacts and enhance positive contributions to the environment.
- **Public Consultation:** Involving stakeholders and the public in the decision-making process to ensure transparency and address concerns.

3. Compliance and Enforcement:

- **Monitoring and Inspection:** Regularly checking compliance with environmental regulations through monitoring programs and site inspections.

- **Enforcement Actions:** Taking legal actions against individuals or entities that violate environmental laws, which may include fines, penalties, or corrective measures.

4. Sustainable Resource Management:

- **Biodiversity Conservation:** Implementing measures to protect and conserve biodiversity, including the establishment of protected areas, habitat restoration, and species conservation programs.

- **Water Resource Management:** Ensuring sustainable use and protection of water resources through efficient water management practices, pollution control, and watershed management.

- **Land Use Planning:** Balancing development with environmental conservation by regulating land use and zoning, promoting sustainable agriculture, and preventing urban sprawl.

5. Waste Management:

- **Waste Reduction and Recycling:** Implementing strategies to reduce waste generation, promote recycling, and encourage the responsible disposal of waste materials.

- **Hazardous Waste Management:** Safely handling and disposing of hazardous materials to prevent environmental contamination and protect human health.

6. Air Quality Management :

- **Emission Control:** Regulating and monitoring air emissions from industrial processes, transportation, and other sources to minimize air pollution.

- **Alternative Energy Sources:** Encouraging the use of cleaner and renewable energy sources to reduce reliance on fossil fuels and decrease greenhouse gas emissions.

7. Environmental Education and Awareness:

- **Public Outreach:** Educating the public and stakeholders about environmental issues, sustainable practices, and the importance of conservation.

- **Capacity Building:** Providing training and building the capacity of individuals, communities, and organizations to actively participate in environmental management efforts.

8. Continuous Improvement:

- **Adaptive Management:** Using feedback mechanisms and ongoing monitoring to adjust environmental management strategies as new information becomes available.

- **Research and Innovation:** Investing in research and development to advance environmentally friendly technologies, methodologies, and solutions.

9. International Cooperation:

- **Global Collaboration:** Engaging in international partnerships and agreements to address Trans boundary environmental issues, such as climate change, biodiversity loss, and marine pollution.
- **Information Sharing:** Collaborating with other countries to share knowledge, expertise, and best practices for effective environmental management.

Environmental management is a dynamic and interdisciplinary field that involves the active participation of governments, businesses, communities, and individuals. The ultimate goal is to achieve a balance between human development and environmental protection, ensuring that present and future generations can enjoy a healthy and sustainable planet.

UNIT 02

Environmental policy analysis can be conducted at both micro (individual projects or specific cases) and macro (regional, national, or global) levels. Analyzing environmental policies at these different scales provides a comprehensive understanding of their impacts, effectiveness, and implications. Here's a breakdown of environmental policy analysis at the micro and macro levels:

Micro-Level Environmental Policy Analysis:

1. Project-Specific Impact Assessment:

- Evaluate the environmental impact of specific projects or activities.
- Assess potential harm to local ecosystems, biodiversity, and natural resources.

Consider the immediate and localized effects on air and water quality. 2. **Community Engagement:**

- Analyze the involvement of local communities in policy development and implementation.
- Assess the social and economic implications of environmental policies on specific communities.
- Evaluate the effectiveness of community-based initiatives and local conservation efforts.

3. Technology and Innovation:

- Assess the adoption and impact of specific technologies within a project or industry.
- Evaluate the efficiency and environmental performance of innovative solutions.

Consider the feasibility of implementing cleaner technologies at a smaller scale. 4.

Regulatory Compliance:

- Analyze how well individual entities adhere to environmental regulations and standards.
- Evaluate the enforcement and effectiveness of regulatory measures at the project level.
- Identify potential gaps in compliance and enforcement mechanisms.

5. Economic Analysis:

- Assess the economic implications of environmental policies on individual businesses or projects.
- Consider the cost-effectiveness of specific pollution control measures.
- Analyze the economic benefits of sustainable practices at the micro level.

Macro-Level Environmental Policy Analysis:

1. National and Global Trends:

- Evaluate the broader environmental policy landscape at national and global levels.
- Assess trends in policy development, implementation, and effectiveness.
- Consider the alignment of policies with international agreements and commitments.

2. Ecosystem Services and Biodiversity:

- Analyze the impact of policies on larger ecosystems and biodiversity.
- Assess the effectiveness of conservation measures on a regional or national scale.
- Consider the interconnectedness of ecosystems and the cumulative impact of policies.

3. Climate Change and Global Agreements:

- Evaluate policies addressing climate change on a global scale.
- Assess the effectiveness of international agreements and commitments.
- Analyze contributions to greenhouse gas emissions and mitigation efforts at the macro level.

4. Energy and Resource Management:

- Assess policies related to energy production, consumption, and resource management.
- Analyze the impact on national resource reserves and depletion rates.
- Evaluate the transition to renewable energy sources on a larger scale.

5. Policy Integration and Coordination:

- Analyze the integration of environmental policies with other sectors (e.g., economic, social, health).
- Assess the coordination among different levels of government in policy implementation.
- Consider the synergies and conflicts between environmental policies and other policy domains.

6. Long-Term Sustainability:

- Evaluate the long-term sustainability of environmental policies.
- Assess the resilience of ecosystems and the ability of policies to adapt to changing environmental conditions.
- Consider intergenerational equity and the impact of policies on future generations.

Policy Analysis in Environmental Management

Policy analysis in environmental management involves a systematic approach to evaluating and assessing the effectiveness, efficiency, and impacts of policies. Various methods, including operational methods, quantitative methods, statistical analysis, resource allocation analysis, and environmental economics, can be employed. Here are the general steps involved in policy analysis, with a focus on these methods:

1. Problem Definition:

- Clearly define the environmental issue or problem that the policy aims to address.
- Identify the key stakeholders and their interests.
- Determine the scope and scale of the problem at hand.

2. Policy Context:

- Examine the historical context and evolution of existing policies related to the environmental issue.
- Analyze the legal and institutional framework within which the policy operates.

Identify any gaps or inconsistencies in the current policy context.

3. Policy Goals and Objectives:

- Evaluate the stated goals and objectives of the environmental policy.
- Determine whether the goals are specific, measurable, achievable, relevant, and time bound (SMART).
- Assess the alignment of policy objectives with the identified environmental problem.

4. Operational Methods:

- Evaluate how well the policy is operationalized in practice.
- Assess the implementation mechanisms, including monitoring, enforcement, and compliance.
- Consider the role of various stakeholders in implementing the policy.

Quantitative Methods:

- Employ quantitative tools such as cost-benefit analysis (CBA) to assess the economic impacts of the policy.
- Use mathematical models to simulate the potential outcomes and effectiveness of the policy.
- Quantify environmental indicators to measure the success of the policy in achieving its goals.

6. Statistical Analysis:

- Utilize statistical methods to analyze data related to environmental parameters and policy outcomes.
- Conduct regression analysis or time-series analysis to identify correlations or trends.
- Evaluate statistical significance to draw meaningful conclusions.

7. Public Policy Analysis:

- Assess the political feasibility of the policy by analyzing stakeholder interests, power dynamics, and policy acceptance.
- Conduct a public opinion analysis to gauge public support or opposition.

Examine the policy's alignment with political ideologies and public values.

Resource Allocation Analysis:

- Evaluate how resources (financial, human, and technological) are allocated to implement the policy.
- Assess the efficiency of resource use in achieving the policy objectives.

Consider alternative resource allocation scenarios to optimize outcomes.

Environmental Economics:

- Apply economic principles to analyze the market and non-market values of environmental resources.
- Conduct a cost-effectiveness analysis to evaluate the efficiency of policy measures.
- Assess the potential for market-based instruments, such as cap-and-trade systems or environmental taxes.

10. Policy Recommendations:

- Based on the analysis, provide recommendations for policy improvement or revision.
- Consider alternative policy options and their potential impacts.
- Propose strategies for enhancing the policy's effectiveness, efficiency, and sustainability.

11. Communication of Findings:

- Present the findings of the policy analysis in a clear and accessible manner.
- Communicate

the implications of the analysis to policymakers, stakeholders, and the public.

- Ensure transparency and openness in sharing the methodology and data used in the analysis.

12. Feedback Loop and Iterative Process:

- Establish mechanisms for continuous monitoring and evaluation of the policy's implementation.
- Implement a feedback loop to incorporate lessons learned and adjust the policy as needed.
- Recognize that policy analysis is often an iterative process, and revisions may be necessary over time.

By following these steps and employing a combination of operational methods, quantitative analysis, statistical techniques, public policy analysis, resource allocation analysis, and environmental economics, policymakers can gain valuable insights into the effectiveness and implications of environmental policies.

Environmental Management

Unit Three

Notes

Environmental Management plan:-

An Environmental Management Plan (EMP) is a comprehensive document that outlines the strategies, actions, and procedures to effectively manage and mitigate environmental impacts associated with a project, activity, or operation. It is typically developed during the planning phase of a project and serves as a guiding framework throughout its lifecycle. The primary goal of an EMP is to ensure that environmental considerations are integrated into decision-making processes and that environmental sustainability is maintained.

Key components of an Environmental Management Plan may include:

1. **Introduction and Project Description:** Provides an overview of the project, including its objectives, scope, and location.
2. **Legal and Regulatory Requirements:** Identifies relevant environmental laws, regulations, permits, and standards that the project must comply with.
3. **Environmental Impact Assessment (EIA):** Summarizes the findings of the environmental impact assessment conducted for the project, including identification of potential environmental impacts and proposed mitigation measures.
4. **Objectives and Targets:** Sets out specific environmental objectives and targets that the project aims to achieve, such as reducing emissions, minimizing waste generation, or preserving biodiversity.
5. **Roles and Responsibilities:** Defines the roles and responsibilities of project stakeholders

involved in environmental management, including project managers, environmental officers, contractors, and regulatory authorities.

6. **Environmental Management Framework:** Outlines the overarching principles, policies, and procedures for managing environmental risks and impacts throughout the project lifecycle.
7. **Mitigation Measures:** Specifies the measures to be implemented to prevent, minimize, or offset adverse environmental impacts, such as pollution control measures, habitat restoration, or waste management practices.
8. **Monitoring and Reporting:** Describes the monitoring protocols and methodologies to assess the effectiveness of mitigation measures and compliance with environmental standards. It also outlines reporting requirements and frequency.
9. **Emergency Response and Contingency Planning:** Establishes procedures and protocols for responding to environmental emergencies, such as spills, leaks, or natural disasters, to minimize environmental damage and risks to human health.
10. **Training and Capacity Building:** Identifies the training needs of personnel involved in environmental management and provides guidelines for capacity building activities to enhance environmental awareness and competency.
11. **Stakeholder Engagement and Consultation:** Describes the mechanisms for engaging with stakeholders, including local communities, indigenous groups, NGOs, and government agencies, to solicit feedback, address concerns, and foster collaboration.
12. **Review and Revision:** Specifies the process for periodic review and revision of the EMP to incorporate lessons learned, changes in regulations, or emerging environmental issues.

An effective Environmental Management Plan not only helps to minimize adverse environmental impacts but also promotes sustainable development by integrating environmental considerations into decision-making processes and fostering a culture of environmental stewardship and responsibility.

Factor affecting Preparation of Environment Management plan :-

The preparation of an Environmental Management Plan (EMP) is influenced by various factors, including regulatory requirements, project characteristics, environmental considerations, stakeholder interests, and resource availability. Here are some of the key factors affecting the preparation of an EMP:

1. **Regulatory Framework:** Compliance with local, national, and international environmental regulations and standards is a primary driver for developing an EMP. Requirements may include environmental impact assessments, permits, and reporting obligations.
2. **Nature of the Project:** The scale, complexity, and type of project significantly influence the content and scope of the EMP. Large infrastructure projects may require more detailed and comprehensive plans compared to smaller-scale activities.
3. **Environmental Impact Assessment (EIA):** The findings of the EIA, including identification of potential environmental impacts and mitigation measures, provide important input for developing the EMP.
4. **Project Location:** Environmental characteristics of the project site, such as sensitive ecosystems, biodiversity hotspots, and proximity to water bodies or residential areas, influence the environmental management strategies and measures included in the EMP.
5. **Stakeholder Concerns and Interests:** Input from stakeholders, including local communities, indigenous groups, environmental NGOs, regulatory agencies, and project proponents, shapes the priorities, objectives, and mitigation measures outlined in the EMP.
6. **Resource Availability:** The availability of financial, human, and technical resources impacts the feasibility and implementation of environmental management measures. Adequate resources are essential for monitoring, compliance, and capacity building activities.
7. **Technology and Best Practices:** Advances in environmental technology and best practices influence the selection of mitigation measures and management strategies included in the EMP. Incorporating innovative solutions can enhance environmental performance and sustainability.
8. **Project Phases and Lifecycle:** The different phases of the project lifecycle

Preparation of Environmental Management Plan

Preparing an Environmental Management Plan (EMP) involves a systematic process to assess environmental risks, identify mitigation measures, and establish management strategies to minimize adverse impacts and promote environmental sustainability. Here's a step-by-step guide on how to propose an EMP:

1. **Understand Regulatory Requirements:** Familiarize yourself with relevant environmental regulations, permits, and standards applicable to your project. Determine the legal framework and compliance obligations that will shape the EMP.
2. **Conduct Environmental Impact Assessment (EIA):** Conduct a thorough assessment of the project's potential environmental impacts. Identify key environmental aspects and sensitivities of the project site. Engage qualified environmental consultants if necessary to perform detailed assessments.
3. **Identify Stakeholders:** Identify and engage with stakeholders, including local communities, indigenous groups, regulatory agencies, NGOs, and project proponents. Understand their concerns, interests, and expectations regarding environmental management.
4. **Set Objectives and Targets:** Establish clear environmental objectives and targets that align with project goals and regulatory requirements. Define measurable indicators to assess progress and performance.
5. **Risk Assessment and Mitigation:** Identify potential environmental risks associated with project activities. Prioritize risks based on severity and likelihood of occurrence. Develop mitigation measures and contingency plans to address identified risks effectively.
6. **Develop Management Strategies:** Based on the findings of the EIA and risk assessment, develop management strategies to minimize adverse impacts and promote environmental sustainability. Consider measures such as pollution prevention, habitat restoration, waste management, and energy efficiency.
7. **Outline Monitoring and Reporting Protocols:** Define monitoring protocols and methodologies to track environmental performance indicators and assess compliance with environmental standards. Establish reporting requirements and frequency to communicate environmental performance to stakeholders.
8. **Incorporate Adaptive Management:** Incorporate adaptive management principles to allow for flexibility and adjustment of environmental management measures based on changing conditions, new information, or lessons learned during project implementation.
9. **Allocate Resources:** Determine the financial, human, and technical resources required to implement the EMP effectively. Ensure adequate resources are allocated for monitoring, training, and capacity building activities.
10. **Integrate Stakeholder Feedback:** Seek input and feedback from stakeholders throughout the EMP development process. Consider incorporating suggestions and

addressing concerns raised

Various Components of Environmental Management plan:-

An Environmental Management Plan (EMP) comprises several components aimed at effectively managing and mitigating environmental impacts associated with a project, activity, or operation. While the specific components may vary depending on the nature and scale of the project, typical elements of an EMP include:

1. **Introduction and Project Description:** Provides an overview of the project, including its objectives, scope, location, and duration. It sets the context for the EMP and outlines the need for environmental management.
2. **Legal and Regulatory Requirements:** Summarizes the relevant environmental laws, regulations, permits, and standards that govern the project. It ensures compliance with applicable legal and regulatory frameworks.
3. **Environmental Impact Assessment (EIA):** Presents the findings of the EIA conducted for the project, including identification of potential environmental impacts and proposed mitigation measures. It serves as a basis for developing the EMP.
4. **Objectives and Targets:** Defines specific environmental objectives and targets that the project aims to achieve. These may include goals related to pollution prevention, resource conservation, biodiversity conservation, and community engagement.
5. **Management Framework:** Outlines the overarching principles, policies, and procedures for managing environmental risks and impacts throughout the project lifecycle. It provides guidance on implementing environmental management practices.
6. **Mitigation Measures:** Specifies the measures to be implemented to prevent, minimize, or offset adverse environmental impacts. These measures may include pollution control measures, habitat restoration, waste management practices, and erosion control measures.
7. **Monitoring and Reporting Protocols:** Describes the monitoring protocols, methodologies, and indicators to assess the effectiveness of mitigation measures and ensure compliance with environmental standards. It also outlines reporting requirements, frequency, and responsible parties.
8. **Roles and Responsibilities:** Defines the roles and responsibilities of project stakeholders involved in environmental management, including project managers, environmental officers, contractors, and regulatory authorities. It clarifies accountability and

communication channels.

9. **Training and Capacity Building:** Identifies the training needs of personnel involved in environmental management and provides guidelines for capacity building activities to enhance environmental awareness, compliance, and competency.
10. **Emergency Response and Contingency Planning:** Establishes procedures and protocols for responding to environmental emergencies, such as spills, leaks, or natural disasters. It outlines contingency plans to minimize environmental damage and risks to human health.
11. **Stakeholder Engagement and Consultation:** Describes the mechanisms for engaging with stakeholders, including local communities, indigenous groups, environmental NGOs, and government agencies. It solicits feedback, addresses concerns, and fosters collaboration.
12. **Review and Revision:** Specifies the process for periodic review and revision of the EMP to incorporate lessons learned, changes in regulations, or emerging environmental issues. It ensures that the EMP remains relevant and effective over time.
13. These components collectively form a comprehensive Environmental Management Plan that guides the integration of environmental considerations into project planning, implementation, and monitoring, ultimately promoting sustainable development and environmental stewardship.

Environmental Management

Unit Four

Water Prevention and control Act 1974:-

The Water (Prevention and Control of Pollution) Act, 1974 is a key environmental legislation in India aimed at preventing and controlling water pollution. Here's an overview of the act:

1. **Objective:** The primary objective of the Water (Prevention and Control of Pollution) Act, 1974 is to prevent and control water pollution and maintain or restore the wholesomeness of water resources in India.
2. **Enactment:** The act was enacted by the Parliament of India in 1974 and came into force on 23rd March 1974.
3. **Applicability:** The act applies to the whole of India, including the Union territories. It covers all water sources such as rivers, lakes, streams, wells, and underground water.
4. **Key Provisions:**

- **Establishment of Pollution Control Boards:** The act provides for the establishment of Central Pollution Control Board (CPCB) at the national level and State Pollution Control Boards (SPCBs) at the state level. These boards are responsible for implementing the provisions of the act, issuing consent to establish and operate industries, conducting inspections, and monitoring water quality.
 - **Restrictions on Discharge of Pollutants:** The act prohibits the discharge of pollutants into water bodies beyond prescribed standards. Industries and other establishments are required to obtain consent from the pollution control boards for discharging effluents into water bodies.
 - **Water Quality Standards:** The act empowers the Central Government to prescribe standards for the quality of water to be maintained in various water bodies. These standards include parameters such as pH, dissolved oxygen, biochemical oxygen demand (BOD), chemical oxygen demand (COD), and specific pollutants.
 - **Powers of Pollution Control Boards:** The act grants powers to the pollution control boards to issue directions, inspect premises, take samples, analyze water quality, and levy penalties for non-compliance with the provisions of the act.
 - **Penalties and Offences:** The act specifies penalties for contravention of its provisions, including fines and imprisonment. It also prescribes penalties for obstructing the pollution control authorities in the discharge of their duties.
5. **Amendments:** The Water (Prevention and Control of Pollution) Act, 1974 has undergone amendments to strengthen its provisions and address emerging environmental challenges. Some significant amendments include the Water (Prevention and Control of Pollution) Amendment Act, 1988 and the Water (Prevention and Control of Pollution) Amendment Act, 2003.
6. **Interplay with Other Laws:** The act works in conjunction with other environmental laws and regulations, such as the Environment (Protection) Act, 1986, and the Water Cess Act, 1977, to ensure comprehensive protection and management of water resources in India.

Air Prevention and Control Pollution Act 1981:-

The Air (Prevention and Control of Pollution) Act, 1981 is a significant environmental legislation in India aimed at preventing and controlling air pollution. Here's an overview of the act:

1. **Objective:** The primary objective of the Air (Prevention and Control of Pollution) Act, 1981 is to prevent and control air pollution and maintain or restore the quality of air in India to a level consistent with the preservation of public health and the environment.
2. **Enactment:** The act was enacted by the Parliament of India in 1981 and came into force on 29th March 1981.
3. **Applicability:** The act applies to the whole of India, including the Union territories. It covers all sources of air pollution, including industries, vehicles, thermal power plants, and other activities.

4. Key Provisions:

- **Establishment of Pollution Control Boards:** The act provides for the establishment of Central Pollution Control Board (CPCB) at the national level and State Pollution Control Boards (SPCBs) at the state level. These boards are responsible for implementing the provisions of the act, issuing consent to establish and operate industries, conducting inspections, and monitoring air quality.
- **Restrictions on Emissions:** The act prohibits the emission of pollutants into the atmosphere beyond prescribed standards. It requires industries and other sources of air pollution to obtain consent from the pollution control boards for emitting pollutants into the air.
- **Air Quality Standards:** The act empowers the Central Government to prescribe standards for the quality of air to be maintained in different areas. These standards include permissible limits for pollutants such as sulfur dioxide (SO₂), nitrogen dioxide (NO₂), particulate matter (PM₁₀ and PM_{2.5}), carbon monoxide (CO), and ozone (O₃).
- **Powers of Pollution Control Boards:** The act grants powers to the pollution control boards to issue directions, inspect premises, take samples, analyze air quality, and levy penalties for non-compliance with the provisions of the act.
- **Monitoring and Surveillance:** The act mandates regular monitoring and surveillance of air quality in designated areas. It requires the pollution control boards to establish monitoring stations and collect data on air pollutant levels.
- **Penalties and Offences:** The act specifies penalties for contravention of its provisions, including fines and imprisonment. It also prescribes penalties for obstructing the pollution control authorities in the discharge of their duties.

5. **Amendments:** The Air (Prevention and Control of Pollution) Act, 1981 has undergone amendments to strengthen its provisions and address emerging environmental challenges. Some significant amendments include the Air (Prevention and Control of Pollution) Amendment Act, 1987 and the Air (Prevention and Control of Pollution) Amendment Act, 1988.
6. **Interplay with Other Laws:** The act works in conjunction with other environmental laws and regulations, such as the Environment (Protection) Act, 1986, and the Motor Vehicles Act, 1988, to ensure comprehensive protection and management of air quality in India.

Overall, the Air (Prevention and Control of Pollution) Act, 1981 plays a crucial role in safeguarding public health, preserving the environment, and promoting sustainable development by controlling air pollution from various sources.

Hazardous waste Rules 1989

The Hazardous Waste (Management, Handling, and Transboundary Movement) Rules, 1989 is a significant environmental regulation in India aimed at managing the generation, handling, storage, transportation, treatment, and disposal of hazardous wastes. Here's an overview of the rules:

1. **Objective:** The primary objective of the Hazardous Waste Rules, 1989 is to prevent and control environmental pollution caused by hazardous wastes and ensure their safe management throughout their lifecycle. The rules aim to protect human health and the environment from the adverse effects of hazardous waste.
2. **Enactment:** The rules were formulated and enacted by the Ministry of Environment, Forest, and Climate Change (MoEFCC), Government of India, under the Environment (Protection) Act, 1986.
3. **Applicability:** The Hazardous Waste Rules, 1989 apply to the generation, handling, storage, transportation, treatment, and disposal of hazardous wastes generated by industries, processes, and activities listed in Schedule I of the rules. These include industries such as chemical manufacturing, pharmaceuticals, electronics, pesticides, and others.
4. **Key Provisions:**
 - **Classification of Hazardous Wastes:** The rules classify hazardous wastes based on their characteristics (such as toxicity, reactivity, flammability, and corrosivity) and

specific sources. The classification helps in identifying and managing hazardous wastes effectively.

- **Authorization and Registration:** The rules require industries and entities generating, handling, or treating hazardous wastes to obtain authorization from the State Pollution Control Boards (SPCBs) or Pollution Control Committees (PCCs) before commencing operations. Small quantity generators may need to register instead of obtaining authorization.
 - **Storage and Transportation:** The rules prescribe standards and procedures for the safe storage and transportation of hazardous wastes. It includes requirements for labeling, packaging, handling, and storage of hazardous wastes to prevent spills, leaks, and accidents.
 - **Treatment and Disposal:** The rules specify the methods and technologies for the treatment, recycling, recovery, and disposal of hazardous wastes. Treatment facilities must comply with the prescribed standards and obtain necessary approvals from regulatory authorities.
 - **Manifest System:** The rules establish a manifest system for the tracking and documentation of hazardous waste movement from the point of generation to its final disposal or treatment facility. Manifests provide information on the type, quantity, origin, and destination of hazardous wastes.
 - **Liability and Penalties:** The rules impose liability on generators, transporters, and operators of hazardous waste facilities for any environmental damage or accidents resulting from improper handling or management of hazardous wastes. Penalties may be imposed for non-compliance with the rules.
5. **Amendments:** The Hazardous Waste Rules, 1989 have undergone several amendments over the years to strengthen their provisions, align with international conventions, and address emerging environmental challenges. Amendments have been made in 2000, 2003, 2008, 2016, and 2019.
6. **Interplay with Other Laws:** The rules work in conjunction with other environmental laws and regulations, such as the Environment (Protection) Act, 1986, and the Biomedical Waste Management Rules, 2016, to ensure comprehensive management of hazardous wastes and environmental protection.

Overall, the Hazardous Waste (Management, Handling, and Trans boundary Movement) Rules, 1989 play a crucial role in regulating the generation, handling, and disposal of hazardous

wastes in India, thereby minimizing environmental pollution and protecting public health and safety.

Factory act 1984 and its amendments in 1987

The Factory Act, 1984, also known as the Factories Act, is a key legislation in India aimed at regulating and ensuring the safety, health, and welfare of workers employed in factories. It provides comprehensive guidelines for the establishment, operation, and maintenance of factories, with a focus on preventing occupational hazards and promoting a safe working environment. Here's an overview of the Factory Act, 1984, and its amendments in 1987:

1. **Objective:** The primary objective of the Factory Act, 1984, is to safeguard the health, safety, and welfare of workers employed in factories by setting forth provisions related to factory premises, working conditions, health and safety measures, employment of young persons and women, and other related aspects.
2. **Enactment:** The Factory Act, 1984, was enacted by the Parliament of India to replace the earlier Factories Act, 1948, with updated provisions to address contemporary industrial challenges and ensure better protection for workers.
3. **Applicability:** The Factory Act, 1984, applies to all factories engaged in manufacturing processes and specified hazardous activities listed under the act. It covers various aspects of factory operations, including health, safety, welfare, working hours, employment of young persons, and annual leave with wages.

4. Key Provisions:

- **Factory Registration and Licensing:** The act mandates the registration and licensing of factories with the appropriate government authority before commencement of operations. It specifies the criteria for registration, renewal, and revocation of factory licenses.
- **Working Hours and Overtime:** The act prescribes the maximum number of working hours per week, daily and weekly rest intervals, and provisions for overtime work. It also regulates the employment of young persons and restricts their working hours.
- **Health and Safety Measures:** The act lays down provisions for ensuring the health and safety of workers within factory premises. It requires factories to maintain adequate ventilation, lighting, cleanliness, and sanitation, as well as provide safety

equipment, protective gear, and first aid facilities.

- **Welfare Facilities:** The act mandates the provision of welfare amenities such as canteens, restrooms, drinking water facilities, and washing facilities for workers. It also includes provisions for the disposal of waste and effluents generated within the factory premises.
- **Inspectors and Enforcement:** The act empowers government-appointed inspectors to conduct inspections of factory premises, examine records, investigate accidents, and enforce compliance with the provisions of the act. It also specifies penalties for violations and contraventions.
- **Amendments in 1987:** In 1987, the Factory Act, 1947, underwent amendments to introduce certain changes and updates to its provisions. While specific details of the amendments in 1987 are not readily available, amendments to the act typically aim to address emerging industrial concerns, enhance worker protection, and improve regulatory effectiveness.

5. **Impact:** The Factory Act, 1947, and its subsequent amendments have significantly contributed to improving working conditions, enhancing occupational safety and health standards, and protecting the rights of workers employed in factories across India. The act has played a crucial role in reducing workplace accidents, occupational diseases, and ensuring the overall well-being of factory workers.

Overall, the Factory Act, 1947, remains a cornerstone legislation in India's industrial framework, providing a legal framework for the regulation of factory operations and the protection of workers' rights and interests.