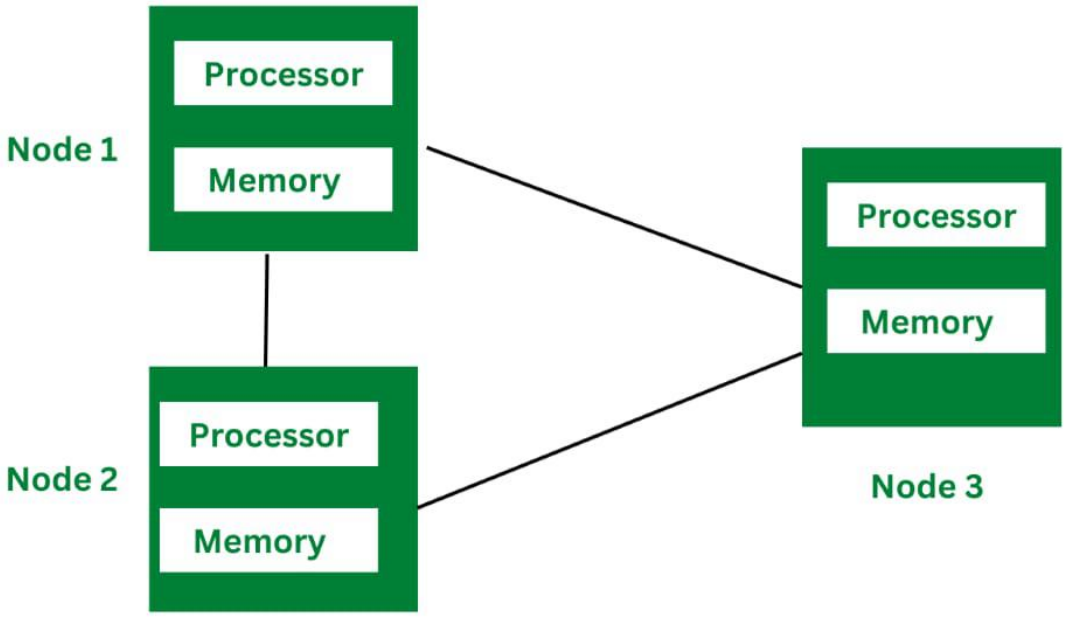
Fault tolerance

Distributed systems are defined as a collection of multiple independent systems connected together as a single system. Every independent system has its own memory and resources and some common resources and peripheral devices that are common to devices connected together. The design of Distributed systems is a complex process where all the nodes or devices need to be connected together even if they are located at long distances. Challenges faced by distributed systems are Fault Tolerance, transparency, and communication primitives. Fault Tolerance is one of the major challenges faced by distributed systems.



In distributed systems, there are three types of problems that occur. All these three types of problems are related.

- **Fault:** Fault is defined as a weakness or shortcoming in the system or any hardware and software component. The presence of fault can lead to error and failure.
- **Errors:** Errors are incorrect results due to the presence of faults.
- **Failure:** Failure is the final outcome where the assigned goal is not achieved.

Fault Tolerance

Fault Tolerance is defined as the ability of the system to function properly even in the presence of any failure. Distributed systems consist of multiple components due to which there is a high risk of faults occurring. Due to the presence of faults, the overall performance may degrade.

Types of Faults

1. **Transient Faults:** Transient Faults are the type of faults that occur once and then disappear. These types of faults do not harm the system to a great extent but are very difficult to find or locate. Processor fault is an example of transient fault.
2. **Intermittent Faults:** Intermittent Faults are the type of faults that comes again and again. Such as once the fault occurs it vanishes upon itself and then reappears again. An example of intermittent fault is when the working computer hangs up.
3. **Permanent Faults:** Permanent Faults are the type of faults that remains in the system until the component is replaced by another. These types of faults can cause very severe damage to the system but are easy to identify. A burnt-out chip is an example of a permanent Fault.

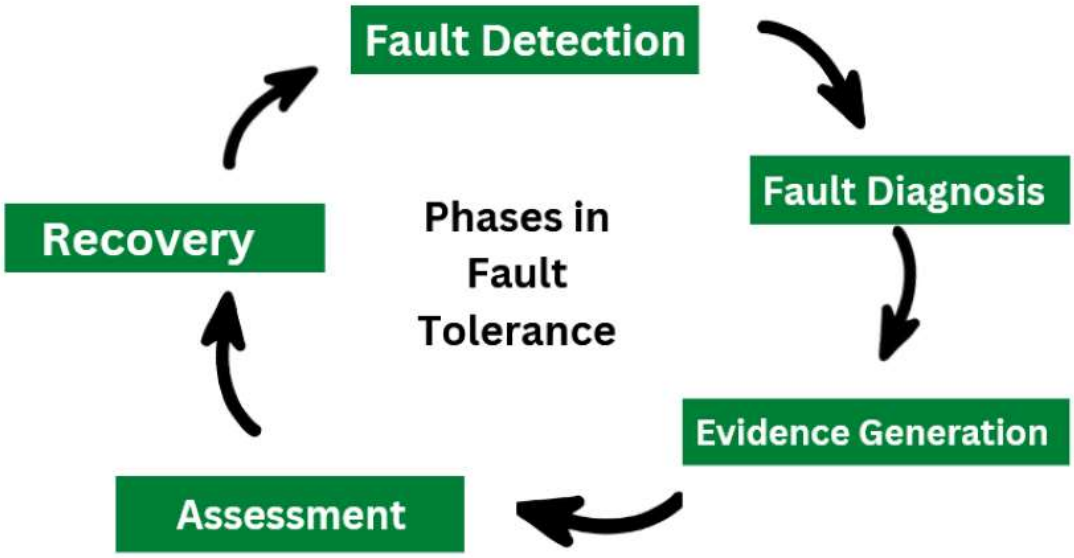
Need for Fault Tolerance in Distributed Systems

Fault Tolerance is required in order to provide below four features.

1. **Availability:** Availability is defined as the property where the system is readily available for its use at any time.
2. **Reliability:** Reliability is defined as the property where the system can work continuously without any failure.
3. **Safety:** Safety is defined as the property where the system can remain safe from unauthorized access even if any failure occurs.
4. **Maintainability:** Maintainability is defined as the property states that how easily and fastly the failed node or system can be repaired.

Fault Tolerance in Distributed Systems

In order to implement the techniques for fault tolerance in distributed systems, the design, configuration and relevant applications need to be considered. Below are the phases carried out for fault tolerance in any distributed systems.



1. Fault Detection

Fault Detection is the first phase where the system is monitored continuously. The outcomes are being compared with the expected output. During monitoring if any faults are identified they are being notified. These faults can occur due to various reasons such as hardware failure, network failure, and software issues. The main aim of the first phase is to detect these faults as soon as they occur so that the work being assigned will not be delayed.

2. Fault Diagnosis

Fault diagnosis is the process where the fault that is identified in the first phase will be diagnosed properly in order to get the root cause and possible nature of the faults. Fault diagnosis can be done manually by the administrator or by using automated Techniques in order to solve the fault and perform the given task.

3. Evidence Generation

Evidence generation is defined as the process where the report of the fault is prepared based on the diagnosis done in an earlier phase. This report involves the details of the causes of the fault, the nature of faults, the solutions that can be used for fixing, and other alternatives and preventions that need to be considered.

4. Assessment

Assessment is the process where the damages caused by the faults are analyzed. It can be determined with the help of messages that are being passed from the component that has encountered the fault. Based on the assessment further decisions are made.

5. Recovery

Recovery is the process where the aim is to make the system fault free. It is the step to make the system fault free and restore it to state forward recovery and backup recovery. Some of the common recovery techniques such as reconfiguration and resynchronization can be used.

Types of Fault Tolerance in Distributed Systems

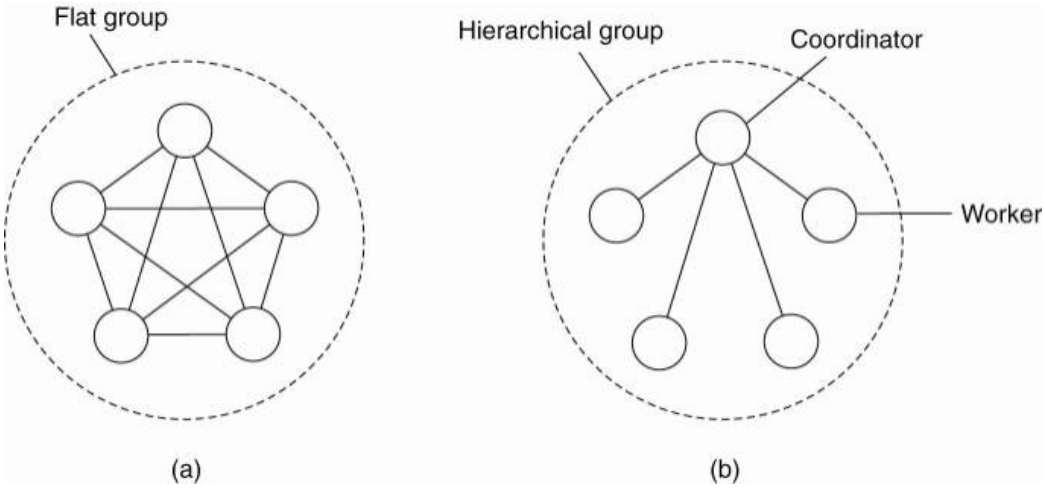
1. **Hardware Fault Tolerance:** Hardware Fault Tolerance involves keeping a backup plan for hardware devices such as memory, hard disk, CPU, and other hardware peripheral devices. Hardware Fault Tolerance is a type of fault tolerance that does not examine faults and runtime errors but can only provide hardware backup. The two different approaches that are used in Hardware Fault Tolerance are fault-masking and dynamic recovery.

2. **Software Fault Tolerance:** Software Fault Tolerance is a type of fault tolerance where dedicated software is used in order to detect invalid output, runtime, and programming errors. Software Fault Tolerance makes use of static and dynamic methods for detecting and providing the solution. Software Fault Tolerance also consists of additional data points such as recovery rollback and checkpoints.

3. **System Fault Tolerance:** System Fault Tolerance is a type of fault tolerance that consists of a whole system. It has the advantage that it not only stores the checkpoints but also the memory block, and program checkpoints and detects the errors in applications automatically. If the system encounters any type of fault or error it does provide the required mechanism for the solution. Thus system fault tolerance is reliable and efficient.

Process Resilience

- Processes can be made fault tolerant by arranging to have a group of processes, with each member of the group being identical.
- A message sent to the group is delivered to all of the "copies" of the process (the group members), and then only one of them performs the required service.
- If one of the processes fail, it is assumed that one of the others will still be able to function (and service any pending request or operation
- Flat Groups versus Hierarchical Groups
- (a) Communication in a flat group. (b) Communication in a simple hierarchical group.



- **Communication in a flat group –** all the processes are equal, decisions are made collectively.

  Note: no single point-of-failure, however: decision making is complicated as consensus is required.

- **Communication in a simple hierarchical group –** one of the processes is elected to be the coordinator, which selects another process (a worker) to perform the operation.

Recovery in Distributed Systems:

Recovery from an error is essential to fault tolerance, and error is a component of a system that could result in failure. The whole idea of error recovery is to replace an erroneous state with an error-free state. Error recovery can be broadly divided into two categories.

1. Backward Recovery:

Moving the system from its current state back into a formerly accurate condition from an incorrect one is the main challenge in backward recovery. It will be required to accomplish this by periodically recording the system's state and restoring it when something goes wrong. A checkpoint is deemed to have been reached each time (part of) the system's current state is noted.

2. Forward Recovery:

Instead of returning the system to a previous, checkpointed state in this instance when it has entered an incorrect state, an effort is made to place the system in a correct new state from which it can continue to operate. The fundamental issue with forward error recovery techniques is that potential errors must be anticipated in advance. Only then is it feasible to change those mistakes and transfer to a new state.

These two types of possible recoveries are done in fault tolerance in distributed system.

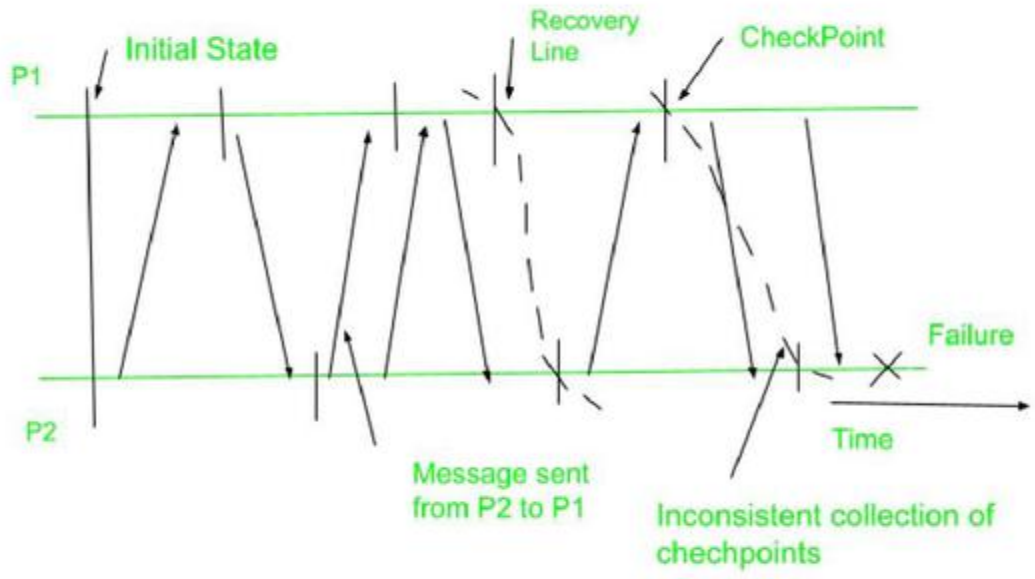
Stable Storage:

Stable storage, which can resist anything but major disasters like floods and earthquakes, is another option. A pair of regular discs can be used to implement stable storage. Each block on drive 2 is a duplicate of the corresponding block on drive 1, with no differences. The block on drive 1 is updated and confirmed first whenever a block is updated. then the identical block on drive 2 is finished.

Suppose that the system crashes after drive 1 is updated but before the update on drive 2. Upon recovery, the disk can be compared with blocks. Since drive 1 is always updated before drive 2, the new block is copied from drive 1 to drive 2 whenever two comparable blocks differ, it is safe to believe that drive 1 is the correct one. Both drives will be identical once the recovery process is finished.

Check pointing:

Backward error recovery calls for the system to routinely save its state onto stable storage in a fault-tolerant distributed system. We need to take a distributed snapshot, often known as a consistent global state, in particular. If a process P has recorded the receipt of a message in a distributed snapshot, then there should also be a process Q that has recorded the sending of that message. It has to originate somewhere, after all.



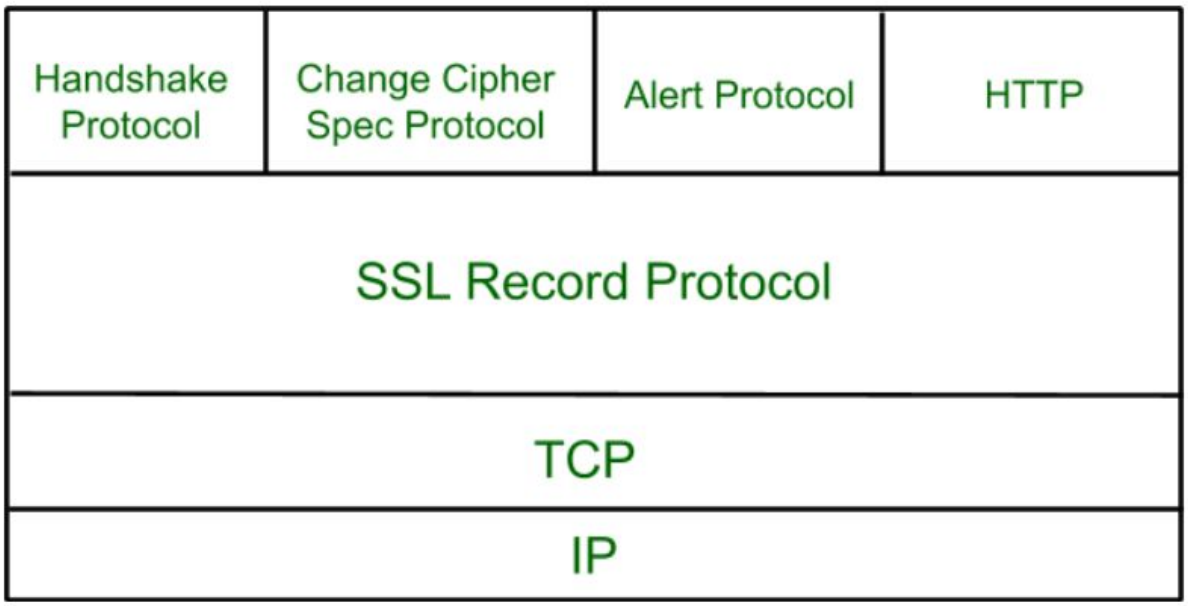
Each process periodically saves its state to a locally accessible stable storage in backward error recovery techniques. We must create a stable global state from these local states in order to recover from a process or system failure. Recovery to the most current distributed snapshot, also known as a recovery line, is recommended in particular. In other words, as depicted in Fig., a recovery line represents the most recent stable cluster of checkpoints.

Secure Socket Layer (SSL)

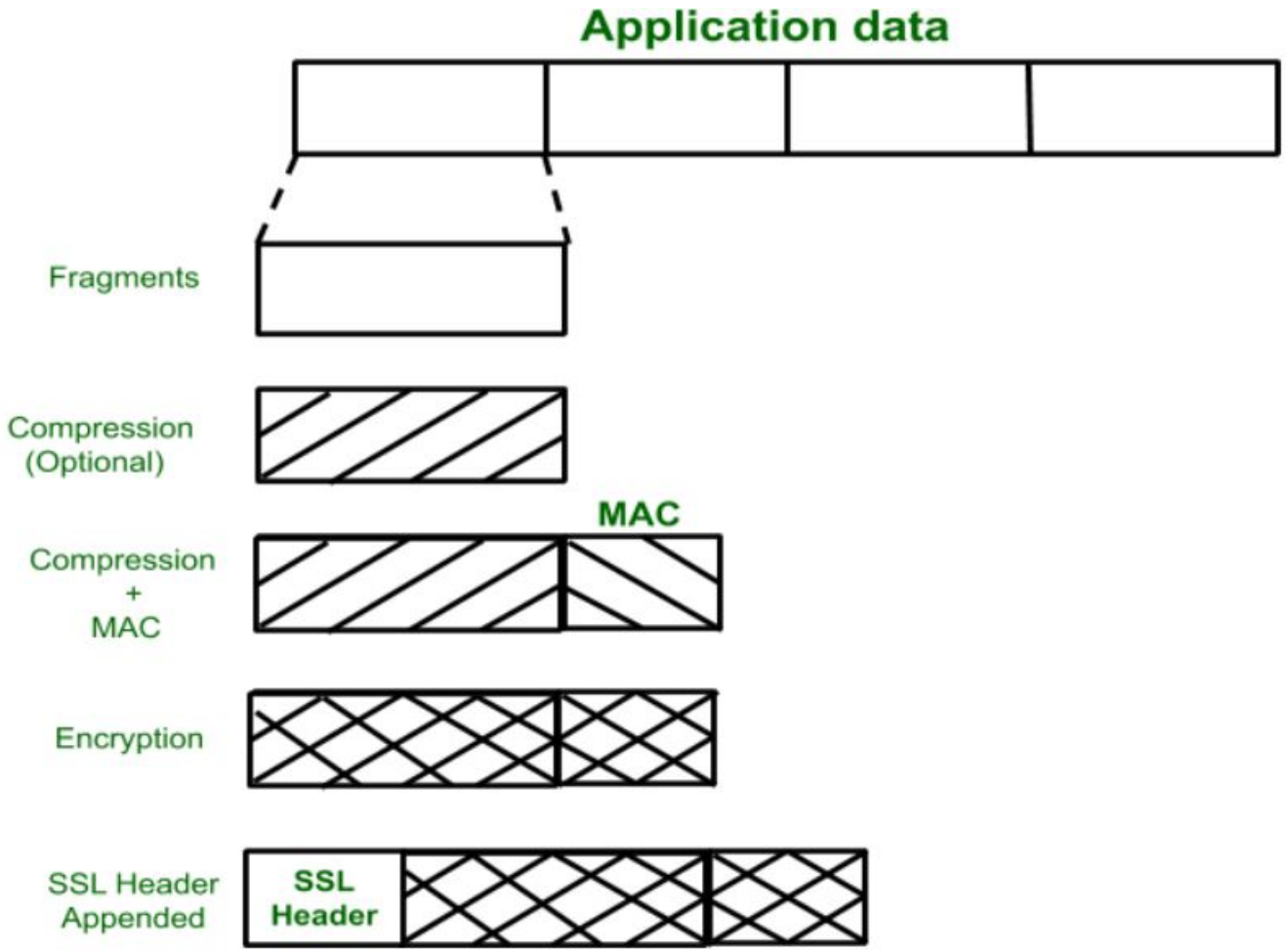
- Secure Socket Layer (SSL) provides security to the data that is transferred between web browser and server. SSL encrypts the link between a web server and a browser which ensures that all data passed between them remain private and free from attack.

- Secure Socket Layer Protocols:
- SSL record protocol
- Handshake protocol
- Change-cipher spec protocol
- Alert protocol

SSL Protocol Stack:

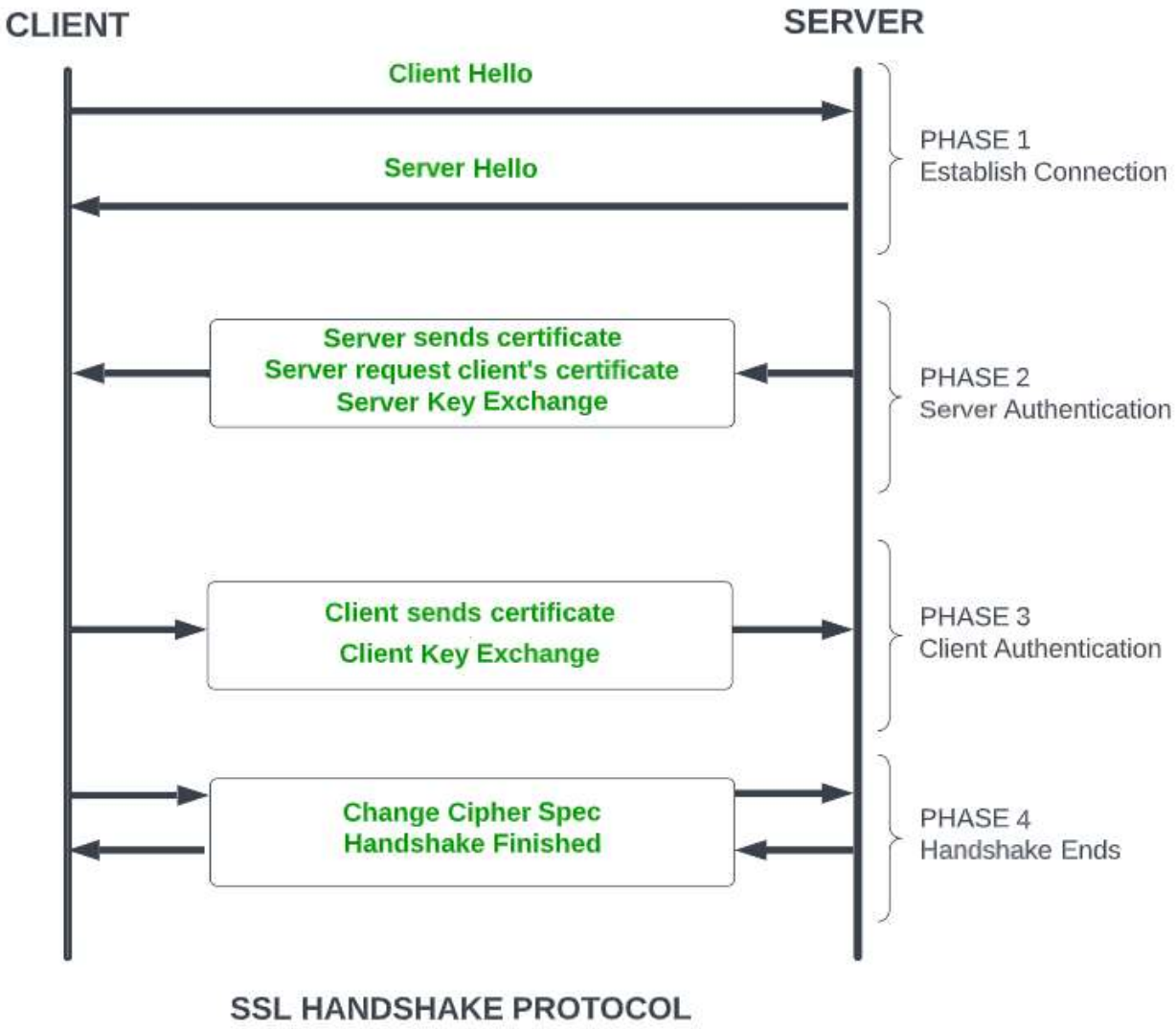


- SSL Record Protocol:
- SSL Record provides two services to SSL connection.
- Confidentiality
- Message Integrity
- In the SSL Record Protocol application data is divided into fragments. The fragment is compressed and then encrypted MAC (Message Authentication Code) generated by algorithms like SHA (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.



- **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 end 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

- 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.

Kerberos

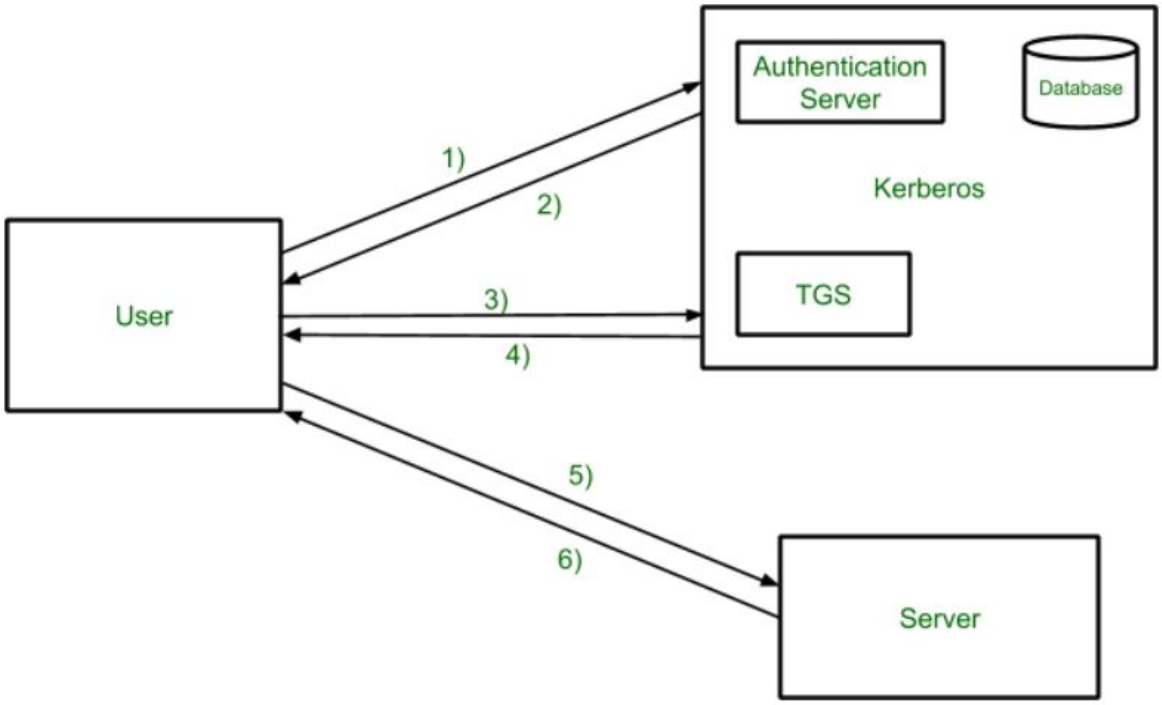
Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. In Kerberos Authentication server and database is used for client authentication. Kerberos runs as a third-party trusted server known as the Key Distribution Center (KDC). Each 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:



- **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.

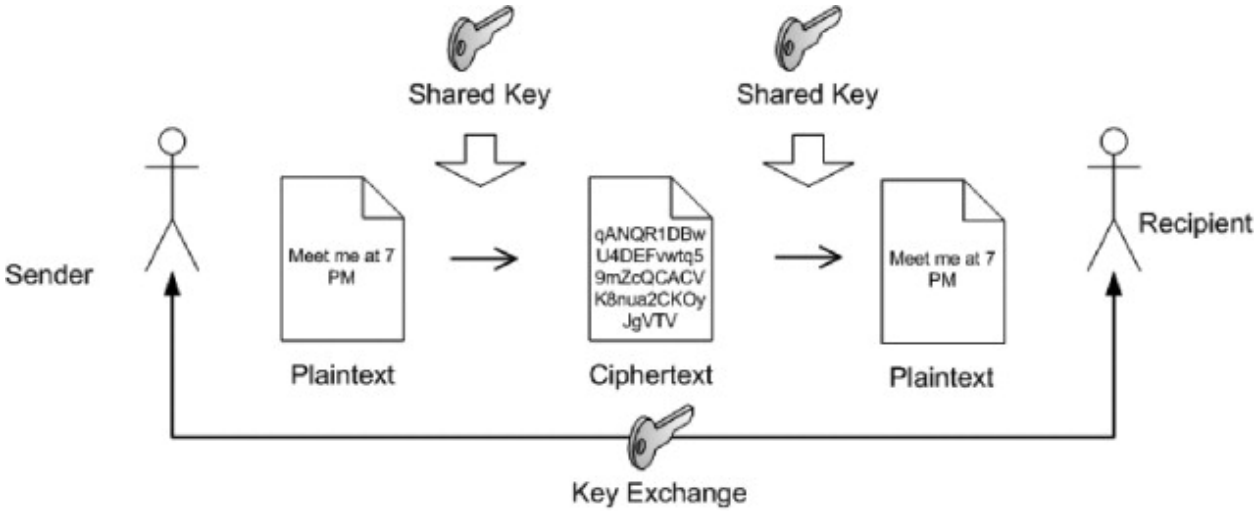
- **Authorization**: Kerberos also offers a system for authorization in addition to authentication. After being authenticated, a user can submit service tickets for certain network resources. Users can access just the resources they have been given permission to use thanks to information about their 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.

Cryptography

The term **cryptography** is a Greek word which means “secret writing”.• It is an art and science of transforming messages so as to make them secure and immune to attacks.

Cryptography involves the process of encryption and decryption. This process is depicted.



• The terminology used in cryptography is given below:

1. **Plaintext:** The original message or data that is fed into the algorithm as input is called plaintext.

2. **Encryption algorithm:** The encryption algorithm is the algorithm that performs various substitutions and transformations on the plaintext. Encryption is the process of changing plaintext into cipher text.

3. **Ciphertext:** Ciphertext is the encrypted form the message. It is the scrambled message produced as output. It depends upon the plaintext and the key.

4. **Decryption algorithm:** The process of changing Ciphertext into plain text is known as decryption. Decryption algorithm is essentially the encryption algorithm run in reverse. It takes the Ciphertext and the key and produces the original plaintext.

5. **Key**: It also acts as input to the encryption algorithm. The exact substitutions and transformations performed by the algorithm depend on the key. Thus a key is a number or a set of number that the algorithm uses to perform encryption and decryption.

Here are the main goals of cryptography:

1. **Confidentiality:** Cryptography helps keep information private and secure from unauthorized access. Encryption is a common technique used to achieve confidentiality by converting plaintext into ciphertext, making it unreadable without the proper decryption key.

2. **Integrity:** Cryptography ensures the integrity of information, meaning that the data has not been altered or tampered with during transmission. Hash functions are commonly employed to generate fixed-size hash values that act as a digital fingerprint for the data.

3. **Authentication:** Cryptography helps verify the identity of parties involved in communication. Digital signatures and authentication protocols are used to confirm the origin and authenticity of messages or transactions.

4. **Non-repudiation:** This ensures that a sender cannot deny the authenticity of a message or transaction. Digital signatures play a crucial role in providing non-repudiation.

There are two type of cryptography:

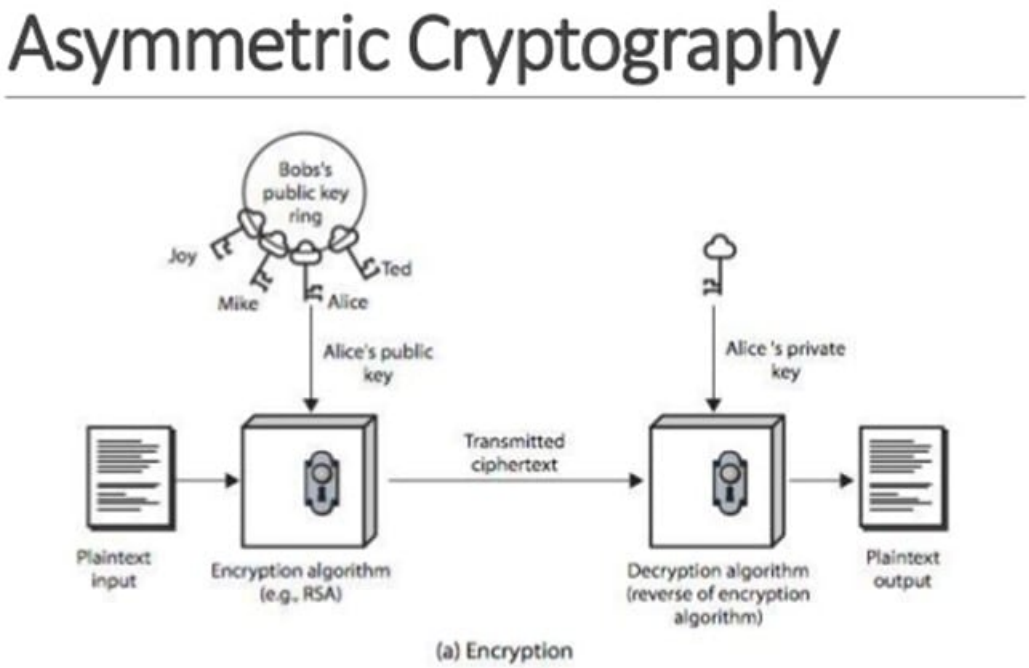
Asymmetric key cryptography

Asymmetric cryptography, also known as public-key cryptography, is a process that uses a pair of related keys -- one public key and one private key -- to encrypt and decrypt a message and protect it from unauthorized access or use.

A public key is a cryptographic key that can be used by any person to encrypt a message so that it can only be decrypted by the intended recipient with their private key. A private key -- also known as a secret key -- is shared only with key's initiator.

When someone wants to send an encrypted message, they can pull the intended recipient's public key from a public directory and use it to encrypt the message before sending it. The recipient of the message can then decrypt the message using their related private key.

If the sender encrypts the message using their private key, the message can be decrypted only using that sender's public key, thus authenticating the sender. These encryption and decryption processes happen automatically; users do not need to physically lock and unlock the message.



Symmetric key cryptography

A symmetric key is one that is used both to encrypt and decrypt information. This means that to decrypt information, one must have the same key that was used to encrypt it. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. This requirement that both parties have access to the secret key is one of the main drawbacks of symmetric key encryption, in comparison to public-key encryption.