Network security and cyber law

Unit 2

Syllabus

- **Application Layer Security:** Security threats and countermeasures SET protocol, Electronic Mail Security, Pretty Good Privacy (PGP), S / Mime.
- **Transport Layer Security:** Secure Socket Layer & Transport Layer Security, Wireless Transport layer security.

OSI (Open Systems Interconnection)

The **OSI (Open Systems Interconnection) Model** is a conceptual framework that standardizes network communication into **seven layers**. Each layer has specific functions and interacts with adjacent layers to ensure smooth data transmission.

7 Layers of the OSI Model:

1. **Physical Layer** – The Physical Layer is the first and lowest layer of the OSI model. It is responsible for the actual transmission of raw binary data (0s and 1s) over a physical medium, such as cables or wireless signals.

2. **Data Link Layer** – The Data Link Layer is the second layer of the OSI model, responsible for node-to-node communication and error detection within a local network (LAN). It ensures that data is transferred correctly between two devices connected on the same network.

3. **Network Layer** – The Network Layer is the third layer of the OSI model, responsible for routing, addressing, and data delivery between different networks. It ensures that data packets travel from the source to the destination, even across multiple networks.

1. **Transport Layer** – The Transport Layer is the fourth layer of the OSI model, responsible for end-to-end communication, data flow control, error handling, and reliability. It ensures that data is delivered completely and correctly between devices. Use port numbers from end to end points

2. **Session Layer** – The Session Layer is the fifth layer of the OSI model, responsible for establishing, maintaining, and terminating communication sessions between applications. It ensures that data exchanges remain organized and synchronized between devices.

3. **Presentation Layer** – The Presentation Layer is the sixth layer of the OSI model, responsible for data translation, encryption, and compression. It ensures that data from the application layer is in a format that the receiving system can understand.

4. **Application Layer** – The Application Layer is the seventh and topmost layer of the OSI model. It provides direct interaction between users and network services, enabling communication between applications over a network.

- The **Application Layer** is highly vulnerable to **cyber security threats** due to its direct interaction with users and network services. Below are some common threats and their countermeasures:

Secure Electronic Transaction (SET)

Secure Electronic Transaction (SET) is a protocol designed to secure online credit card transactions. It ensures confidentiality, integrity, and authentication during e-commerce transactions. SET was developed by Visa, Mastercard, IBM, Microsoft, and other companies, but it was eventually replaced by more efficient protocols like 3D Secure (e.g., Verified by Visa, Mastercard SecureCode) due to its complexity.

Key Components of SET

1. **Cardholder** – The customer who initiates the online purchase.
2. **Merchant** – The online store selling products or services.
3. **Issuer Bank** – The bank that issued the credit card to the cardholder.
4. **Acquirer Bank** – The merchant's bank that processes the payment.
5. **Payment Gateway** – A service that securely processes transactions between the merchant and the acquirer bank.
6. **Digital Certificates** – Used to authenticate all parties and ensure secure communication.

1. Cardholder Registration

The cardholder registers with their issuing bank and gets a **digital certificate** (like an electronic signature).

The digital certificate includes the cardholder's public key and is issued by a **Certificate Authority (CA)**.

2: Merchant Registration

The merchant registers with their acquiring bank and also gets a **digital certificate** for authentication.

3: Purchase Initiation

The cardholder selects a product and proceeds to checkout.

The merchant sends their digital certificate to the customer to prove they are a valid business.

4: Payment Authorization (Dual Signature)

The cardholder sends two encrypted messages:

- **Order Information (OI)** → Contains purchase details but hides payment details from the merchant.
- **Payment Information (PI)** → Contains credit card details but hides order details from the bank.

A **Dual Signature** ensures integrity, using hashing (SHA) and encryption (RSA).

5: Payment Processing

The merchant forwards the encrypted PI to the payment gateway.

The payment gateway verifies the cardholder's certificate and decrypts the PI to check the payment details.

The payment gateway contacts the issuing bank for authorization.

If approved, the gateway sends a payment approval to the merchant.

6: Transaction Confirmation

The merchant sends an order confirmation to the cardholder.

The cardholder receives proof of purchase with the merchant's digital signature.

Limitations of SET

Complexity – Requires extensive infrastructure and software for merchants and banks.
Slow adoption – Due to high costs and technical difficulties.
Replaced by modern protocols – 3D Secure (e.g., Visa Secure, Mastercard Identity Check) is now preferred.

Electronic Mail (Email) Security

- **Electronic Mail (Email) Security** involves multiple layers of authentication, encryption, filtering, and monitoring to prevent unauthorized access, phishing, malware, and data breaches. It ensures that emails are sent, received, and stored securely without exposing sensitive information to attackers.

Key components

1. **Authentication Protocols**

- **SPF (Sender Policy Framework)**: Prevents spammers from sending emails that appear to come from your domain. SPF allows the domain owner to specify which mail servers are authorized to send email on their behalf.
- **DKIM (DomainKeys Identified Mail)**: Uses encryption to authenticate the sender's domain and ensure that the email content is not altered during transmission.
- **DMARC (Domain-based Message Authentication, Reporting & Conformance)**: Helps to protect against phishing attacks by setting up policies on how emails should be handled if they fail SPF or DKIM checks.

2. Encryption

- **Transport Layer Security (TLS)**: Ensures that emails are encrypted during transit, so even if intercepted, the content is unreadable.
- **End-to-End Encryption**: Services like **PGP** (Pretty Good Privacy) or **S/MIME** (Secure/Multipurpose Internet Mail Extensions) encrypt the content of the email itself, ensuring that only the intended recipient can read the message.

PGP (Pretty Good Privacy)

- **PGP (Pretty Good Privacy) is an encryption program used for securing email communication and file storage by providing privacy, authentication, and data integrity.** It combines both data encryption and digital signatures to ensure that the email content or files you send remain confidential and unaltered. PGP is widely used for securing email exchanges, preventing unauthorized access, and confirming the identity of senders.

Advantages of PGP:

- **Confidentiality**: The message can only be decrypted by the recipient who possesses the corresponding private key.
- **Integrity**: The digital signature ensures that the message has not been altered.
- **Authentication**: The sender's digital signature proves that the message came from them.

Use Cases of PGP:

- **Email Encryption**: Ensuring that emails are secure and only readable by the intended recipient.
- **File Encryption**: Encrypting sensitive files before sending or storing them.
- **Digital Signatures**: Verifying the authenticity of documents or software packages.

3. Phishing Protection

- **Phishing** is one of the most common cyber threats in email security. It involves fraudulent emails that attempt to trick users into revealing personal information or downloading malicious content. Protection strategies include:
  - **Educating users** to recognize suspicious emails and not to click on untrusted links.
  - **Antivirus software** that scans emails for malicious attachments or links.
  - **Anti-phishing filters** that detect and block phishing emails.

4. Spam Filtering

- **Spam filters** are essential to detect and block unwanted or unsolicited emails, which can often contain malicious links or attachments. They use various techniques like keyword analysis, blacklists, and machine learning to detect spam.

5. Malware Protection

- **Antivirus scanning** is used to detect and block malicious attachments (like Trojans, worms, or ransomware) that may be included in emails.
- **Behavioral analysis** tools can identify abnormal email behaviors or attachments that could indicate the presence of malware.

1. **Sender Authentication**:The email server verifies SPF, DKIM, and DMARC records to ensure the sender is legitimate.
2. **Encryption & Secure Transmission**: The message is encrypted using **TLS** or **end-to-end encryption** before being sent.
3. **Spam & Threat Filtering**: The recipient's email security system scans for **phishing, malware, and spam**.
4. **User Authentication & Access Control**: The recipient must enter a **password** or use **MFA** to access their email securely.
5. **Data Protection & Compliance Checks**:If the email contains sensitive information, **DLP policies** apply encryption or redaction.

Threat	Description
Phishing	Fraudulent emails trick users into revealing credentials or personal data.
Spear Phishing	Targeted phishing attacks against specific individuals or organizations.
Business Email Compromise (BEC)	Attackers impersonate executives or partners to manipulate employees into transferring money or data.
Spoofing	Attackers forge email headers to make emails appear from trusted sources.
Malware & Ransomware	Malicious attachments or links infect devices and encrypt files for ransom.
Man-in-the-Middle (MITM) Attacks	Attackers intercept and alter email communications.
Spam	Unsolicited emails that may contain malicious links or scams.
Data Loss	Accidental or intentional leaking of sensitive information via email.

Pretty Good Privacy (PGP)

- **Pretty Good Privacy (PGP)** is an encryption program used for secure email communication, file encryption, and digital signatures. It ensures:
- **Confidentiality –** Encrypts messages so only the intended recipient can read them.
- **Integrity –** Ensures messages are not altered in transit.

- **Authentication –** Verifies the sender's identity using digital signatures.

- PGP uses asymmetric encryption (public/private key pairs) and symmetric encryption (session keys) for high security.

PGP encryption involves **two main processes**:

A. Encrypting a Message

1. **Generate a session key**

PGP creates a one-time **random symmetric key** to encrypt the message.

This key is encrypted with the recipient's **public key**.

2. **Encrypt the message**

The session key encrypts the message using a **symmetric cipher (AES, 3DES, etc.)**.

3. **Send the encrypted message**

The recipient receives the **encrypted session key** and the **ciphertext message**.

B. Decrypting a Message

1. **Decrypt the session key**

The recipient's **private key** decrypts the session key.

2. **Decrypt the message**

The decrypted session key decrypts the message.

- A session key is a randomly generated, temporary key used for encrypting a message in symmetric encryption. In PGP, the session key is encrypted using the recipient’s public key and sent along with the encrypted message.

Why use both asymmetric & symmetric encryption?

- Asymmetric encryption (public/private keys) is slow for large messages.
- PGP encrypts messages with fast symmetric encryption and uses public key cryptography only for securing the session key.

3. Digital Signatures in PGP

- PGP also provides digital signatures for authentication & integrity:

Signing a Message

- The sender hashes the message (SHA-256, SHA-512).
- The hash is encrypted using the sender's private key (digital signature).

Verifying a Message

- The recipient decrypts the signature using the sender's public key.
- The hash is compared to the calculated hash to check for tampering.

- PGP also provides **digital signatures** for authentication & integrity:

Signing a Message

- The sender **hashes** the message (SHA-256, SHA-512).
- The hash is **encrypted** using the sender's **private key** (digital signature).

Verifying a Message

- The recipient **decrypts the signature** using the sender's **public key**.
- The hash is compared to the calculated hash to check for tampering.

4. PGP Key Management

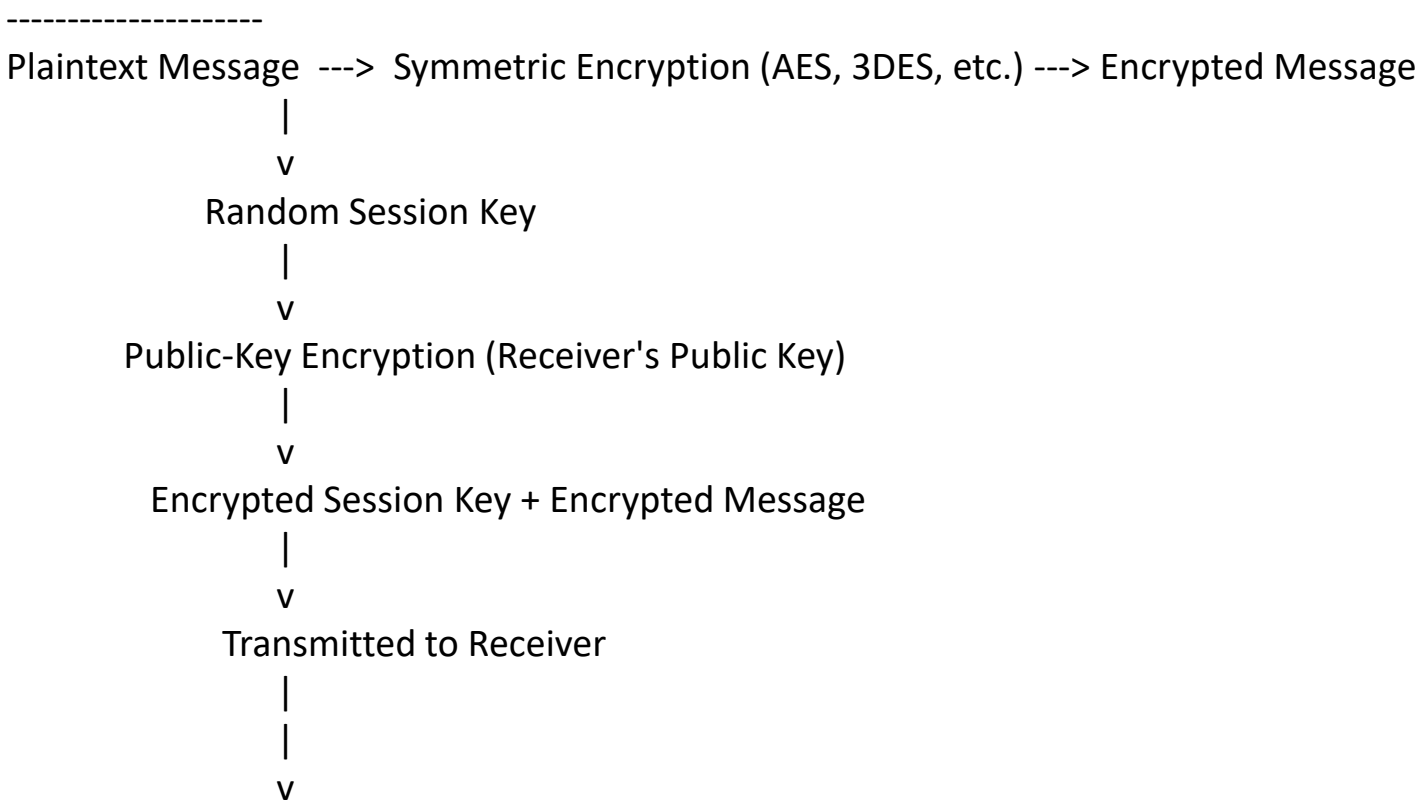
PGP uses a **Web of Trust** model instead of a centralized authority:
Users manually **sign each other's keys** to establish trust.
Public keys are shared via **key servers** (e.g., MIT PGP Keyserver).
Private keys are kept secret for decryption and signing.

SSL (Secure Sockets Layer)

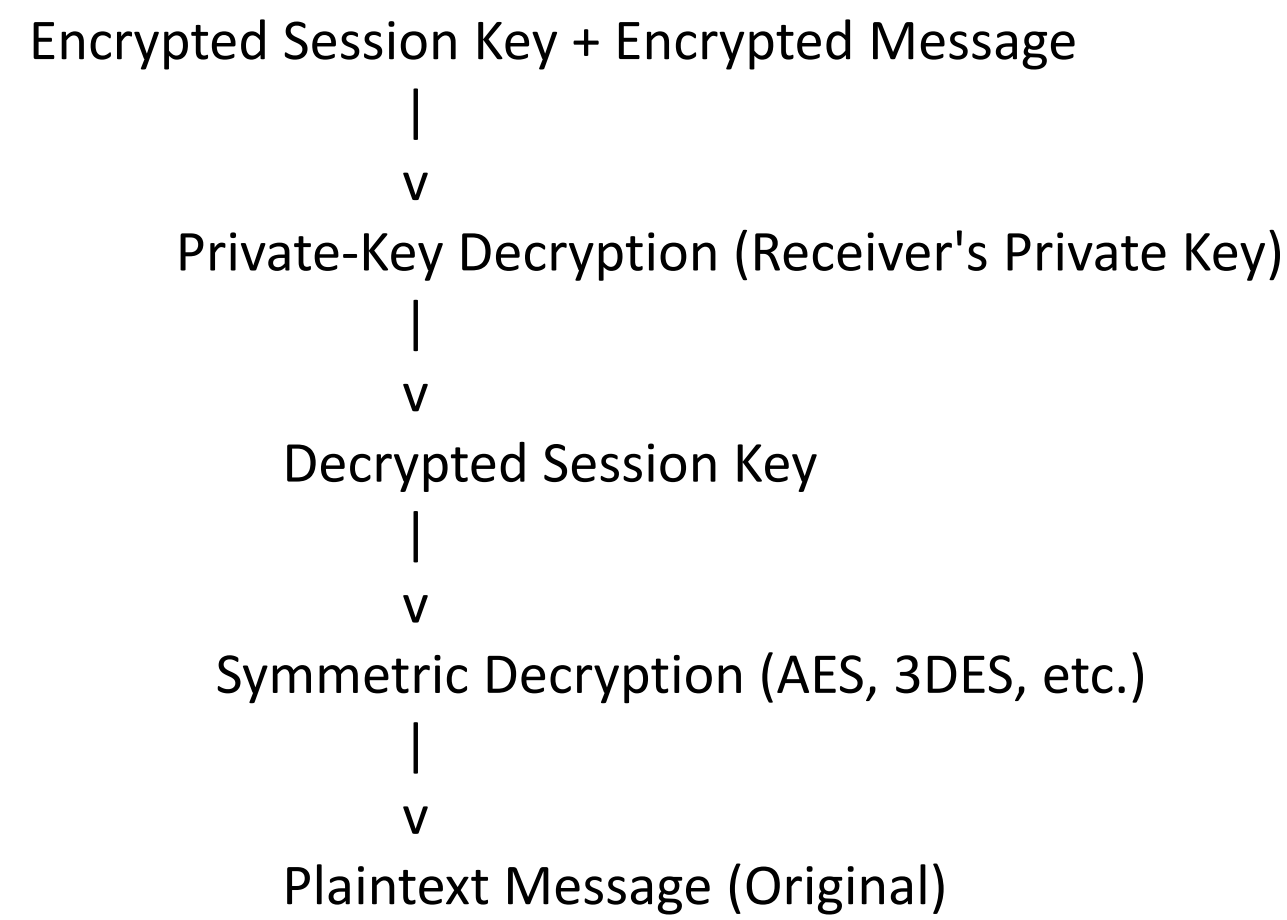
- SSL (Secure Sockets Layer) is an encryption protocol that establishes a secure connection between a client (browser, email client) and a server (website, mail server). It prevents eavesdropping, data tampering, and man-in-the-middle attacks.

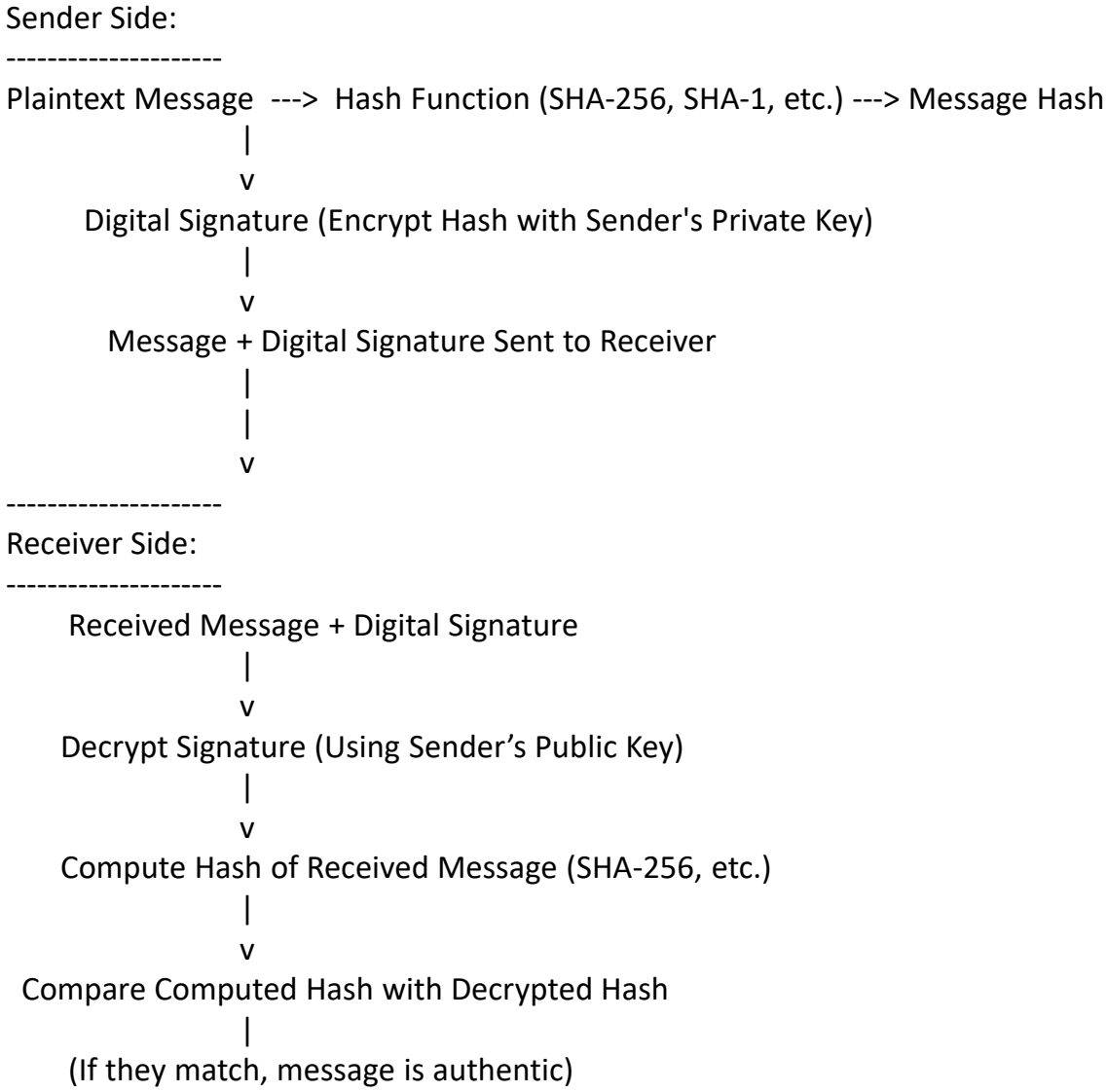
PGP Digital Signature Process

Sender Side:



Receiver Side:




Sender Side:
Plaintext Message ---> Hash Function (SHA-256, SHA-1, etc.) ---> Message Hash
Digital Signature (Encrypt Hash with Sender's Private Key)
Message + Digital Signature Sent to Receiver
Receiver Side:
Received Message + Digital Signature
Decrypt Signature (Using Sender’s Public Key)
Compute Hash of Received Message (SHA-256, etc.)
Compare Computed Hash with Decrypted Hash
(If they match, message is authentic)

What is MIME?

- MIME is an internet standard that extends email capabilities by allowing the transmission of multimedia content, attachments, and non-ASCII text. It enables emails to include images, videos, PDFs, and formatted text (HTML emails) by encoding non-text data into a format suitable for email transmission.

Security Risks of MIME

1. **Lack of Encryption** – MIME emails are sent in plain text, making them vulnerable to eavesdropping.
2. **No Sender Authentication** – Email senders can be spoofed, leading to phishing attacks.
3. **Malware via Attachments** – MIME allows file attachments, which attackers exploit to send viruses or malicious links.
4. **Email Content Modification** – Since MIME lacks integrity checks, attackers can alter email content in transit.

S/MIME (Secure/Multipurpose Internet Mail Extensions)

S/MIME (Secure/Multipurpose Internet Mail Extensions) is a widely used protocol for securing email communications through encryption and digital signatures. It enhances email security by ensuring **confidentiality, integrity, authentication, and non-repudiation** of messages.

Key Features of S/MIME

- **Encryption** – Protects email contents from unauthorized access by encrypting messages using public key cryptography (typically RSA or ECC).
- **Digital Signatures** – Ensures message integrity and sender authentication using certificates issued by a trusted Certificate Authority (CA).
- **Authentication** – Verifies that the email sender is legitimate and prevents email spoofing.
- **Data Integrity** – Detects any tampering of the email content during transit.
- **Non-repudiation** – Prevents the sender from denying that they sent the email, as digital signatures provide cryptographic proof.

How S/MIME works?

a) Sending an Encrypted and Signed Email

- **Message Creation** – The sender composes an email in an S/MIME-supported email client (e.g., Outlook, Apple Mail).
- **Digital Signature (Authentication & Integrity)**
  - The sender signs the email using their **private key**.
  - The recipient verifies it using the sender's **public key (certificate)**.
- **Encryption (Confidentiality)**
  - The sender encrypts the email using the recipient's **public key**.
  - Only the recipient can decrypt it using their **private key**.
- **Email Transmission** – The encrypted and signed email is sent via an SMTP server.

b) Receiving an Encrypted and Signed Email

- The recipient's email client retrieves the email from the mail server like POP3 (Post Office Protocol v3)/ IMAP (Internet Message Access Protocol).
- **Decryption** – The recipient uses their **private key** to decrypt the message.
- **Signature Verification** – The recipient verifies the sender's identity using their **public key (certificate)**.XC
- **Message Display** – If the verification and decryption are successful, the email is displayed as intended.

- **For businesses and enterprises** → S/MIME is more secure because it uses trusted Certificate Authorities (CAs) and integrates well with corporate email systems.

- **For privacy-conscious individuals and activists** → PGP is better because it provides end-to-end encryption without relying on a central authority, making it more resistant to government oversight.

Feature	S/MIME	PGP
Encryption Type	Uses **asymmetric encryption** (public-private key pair) for encryption & digital signing	Uses a **hybrid encryption approach** (symmetric encryption for speed + asymmetric encryption for security)
Key Management	**Centralized** (Relies on **Certificate Authority (CA)** for issuing and managing digital certificates)	**Decentralized** (Uses **Web of Trust**, where users manually verify and sign each other's keys)
Trust Model	Based on **trusted CAs** that issue and validate certificates	Users **manually verify public keys** or rely on a **Web of Trust**
Ease of Use	Easier for enterprises since certificates are automatically issued and managed by Cas	More complex, as users must manually exchange and verify keys
Encryption Scope	**End-to-end encryption** for email content, ensuring confidentiality and authenticity	**End-to-end encryption** using a hybrid method (symmetric key + asymmetric key)
Signature Verification	Uses **X.509 certificates** issued by a CA	Uses **Web of Trust** where users sign and verify public keys
Revocation & Expiry	Certificates can be revoked via a CA and have an expiry date	Users must manually revoke and distribute new keys if compromised
Phishing & Spoofing Resistance	**More secure** against impersonation, as CAs verify sender identity	**Less secure** if users do not manually verify keys properly
Best Used For	**Enterprise environments** (corporate email security, compliance, regulated industries)	**Privacy-focused individuals, activists, and journalists**

TLS (Transport layer security)

- SSL or Secure Sock ets Layer, is an Internet security protocol that encrypts data to keep it safe. It was created by Netscape in 1995 to ensure privacy, authentication, and data integrity in online communications. SSL is the older version of what we now call TLS(Transport Layer Security).

- Websites using SSL/TLS have “HTTPS” in their URL instead of “HTTP.”

Working of SSL(secure socket layer)

- **Encryption**: SSL encrypts data transmitted over the web, ensuring privacy. If someone intercepts the data, they will see only a jumble of characters that is nearly impossible to decode.

- **Authentication**: SSL starts an authentication process called a handshake between two devices to confirm their identities, making sure both parties are who they claim to be.

- **Data Integrity**: SSL digitally signs data to ensure it hasn’t been tampered with, verifying that the data received is exactly what was sent by the sender.

Importance of SSL

- Originally, data on the web was transmitted in plaintext, making it easy for anyone who intercepted the message to read it. For example, if someone logged into their email account, their username and password would travel across the Internet unprotected.

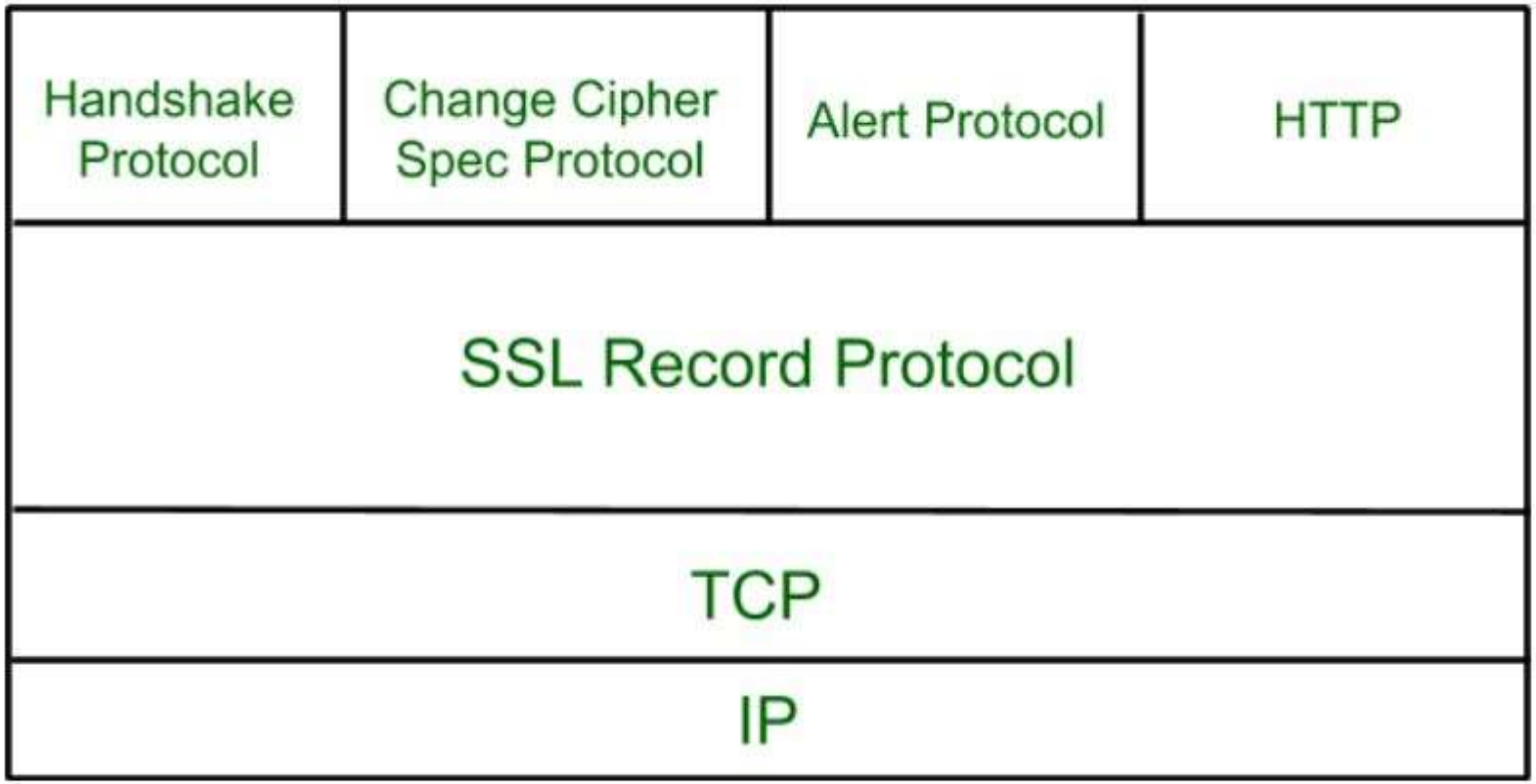
- SSL was created to solve this problem and protect user privacy. By encrypting data between a user and a web server, SSL ensures that anyone who intercepts the data sees only a scrambled mess of characters. This keeps the user's login credentials safe, visible only to the email service.

SSL helps prevent cyber attacks by:

- **Authenticating Web Servers**: Ensuring that users are connecting to the legitimate website, not a fake one set up by attackers.
- **Preventing Data Tampering**: Acting like a tamper-proof seal, SSL ensures that the data sent and received hasn't been altered during transit.

Secure Socket Layer Protocols

- SSL Record Protocol
- Handshake Protocol
- Change-Cipher Spec Protocol
- Alert Protocol



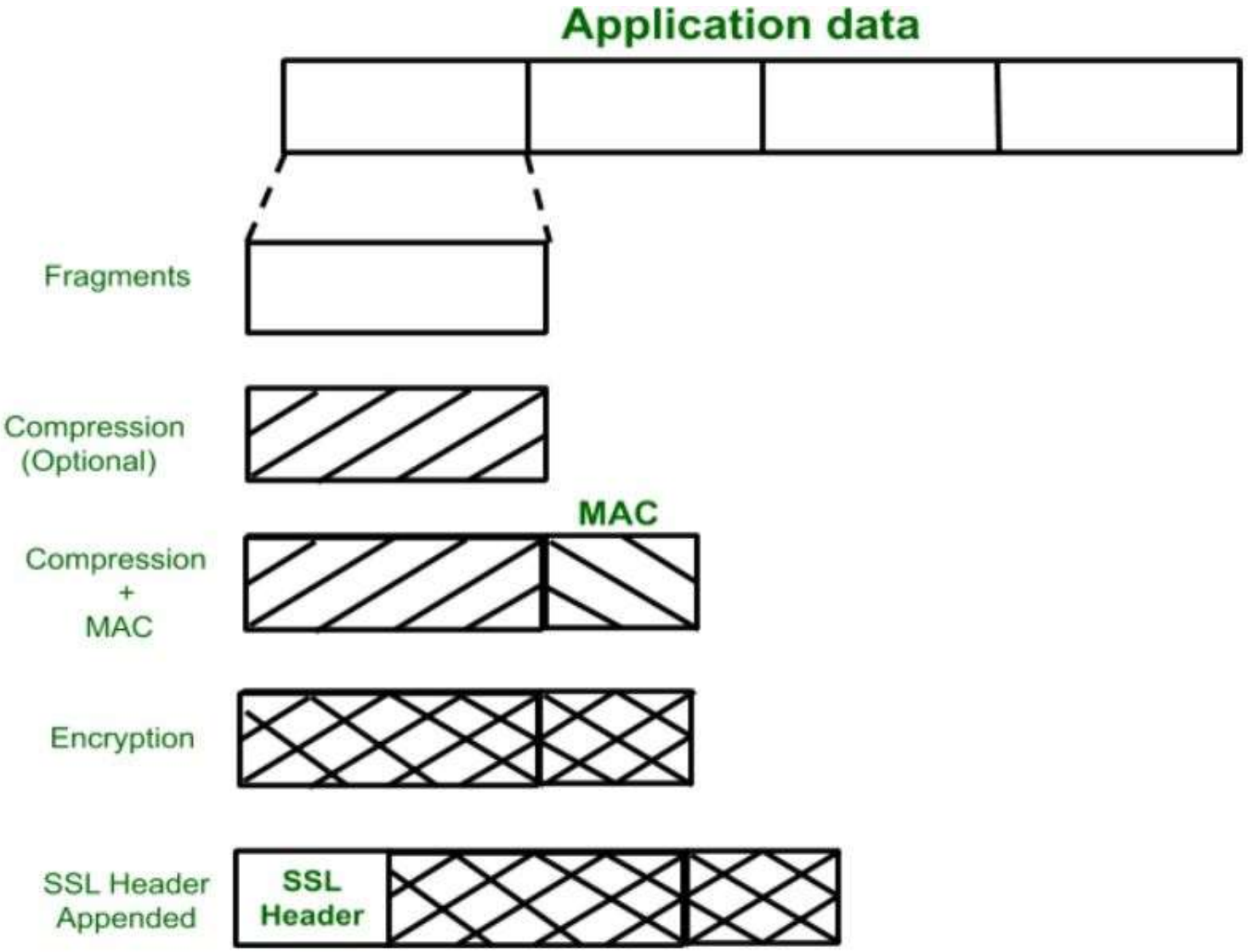
SSL Record Protocol

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.

•


Application data
Fragments
Compression
(Optional)
MAC
Compression
+
MAC
Encryption
SSL Header
Appended
SSL
Header

Handshake Protocol

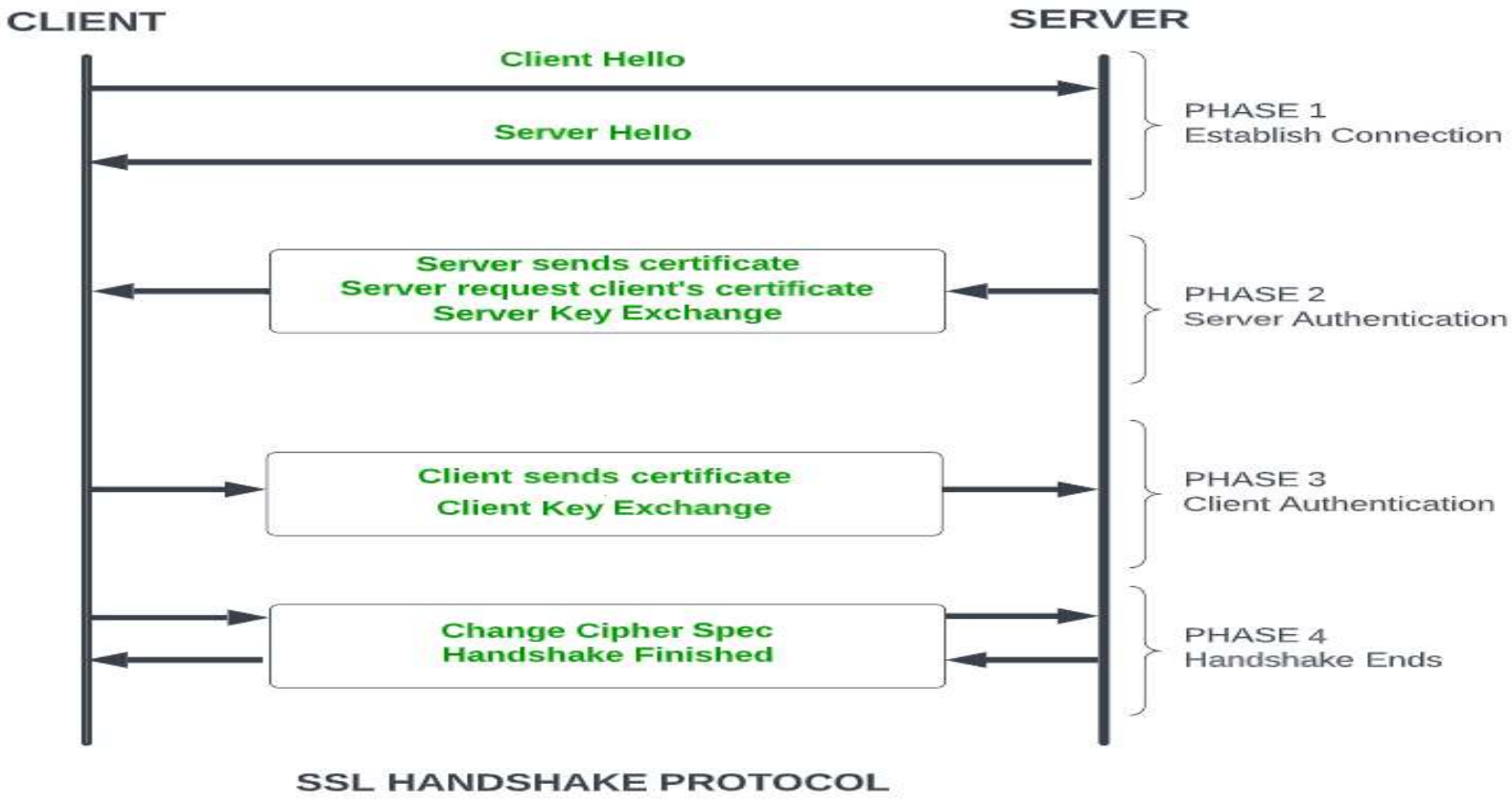
- Handshake Protocol is used to establish sessions. This protocol allows the client and server to authenticate each other by sending a series of messages to each other. Handshake protocol uses four phases to complete its cycle.

- **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 it 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 it certificate and Client-exchange-key.

- **Phase-4:** In Phase-4 Change Cipher Spec occurs and after this the Handshake Protocol ends.



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:

LEVEL	ALERT
1 BYTE	1BYTE

Warning (level = 1):This Alert has no impact on the connection between sender and receiver.

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

1. The advantage of this approach is that the service can be tailored to the specific needs of the given application.
2. Secure Socket Layer was originated by Netscape.
3. SSL is designed to make use of TCP to provide reliable end-to-end secure service.
4. This is a two-layered protocol.

SSL (Secure Sockets Layer) was not secure

SSL (Secure Sockets Layer) was not secure due to multiple **vulnerabilities and weaknesses** in its design, making it susceptible to various cyberattacks. Here are the key reasons why SSL was replaced by TLS:

1. Outdated Cryptographic Algorithms

SSL used weak encryption ciphers such as **RC4** and hashing functions like **MD5 and SHA-1**, which were later found to be vulnerable to cryptographic attacks.

These algorithms could be broken using **brute force** or collision attacks, compromising data security.

2. Vulnerability to Man-in-the-Middle (MITM) Attacks

SSL was vulnerable to **MITM attacks**, where attackers could intercept and manipulate data exchanged between a client and a server.

Examples:

BEAST (Browser Exploit Against SSL/TLS) – Exploited weaknesses in SSL 3.0's cipher block chaining (CBC) mode.

POODLE (Padding Oracle On Downgraded Legacy Encryption) – Allowed attackers to force a downgrade to SSL 3.0 and decrypt encrypted traffic.

3. Lack of Proper Authentication Mechanisms

SSL 2.0 did not verify whether a message had been tampered with, allowing attackers to modify encrypted messages.

SSL 3.0 improved authentication, but it was still vulnerable to downgrade attacks.

4. Susceptibility to Downgrade Attacks

Attackers could force a **protocol downgrade** from a secure version (TLS) to an older, vulnerable version (SSL 3.0), making it easier to exploit known weaknesses.

This was exploited in the **POODLE attack**, allowing attackers to decrypt HTTPS traffic.

5. No Forward Secrecy

SSL did not support **Perfect Forward Secrecy (PFS)**, meaning if an attacker obtained a session key, they could decrypt all past and future communications.

TLS introduced PFS to protect past sessions even if a key was compromised.

Due to these weaknesses, SSL was replaced by **TLS (Transport Layer Security)**, which provides **stronger encryption, better authentication, improved performance, and resistance to modern cyberattacks**.

Transport Layer Securities (TLS)

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 tampers 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.

TLS handshake

- TLS is a data privacy and security protocol implemented for secure communication over internet. It usually encrypts communication between server and clients. TLS is a successor to **Secure Socket Layer (SSL)** protocol. SSL v3.0 and TLS v1.0 were very similar but it was replaced with TLS. You can also refer to **Transport Layer Security (TLS).**

A. TLS has a Record Protocol similar to SSL, but with stronger encryption, better authentication, and improved security. The **TLS 1.3 record protocol** is significantly more secure, as it removes weak cryptographic algorithms and integrates authentication with encryption.

1. TCP Handshake (Three-Way Handshake)

Before the TLS handshake begins, a standard **TCP three-way handshake** is performed to establish a reliable connection:

SYN: The client initiates the connection by sending a SYN (synchronize) message to the server.

SYN-ACK: The server responds with a SYN-ACK (synchronize-acknowledge) message.

ACK: The client acknowledges the server's response with an ACK message.

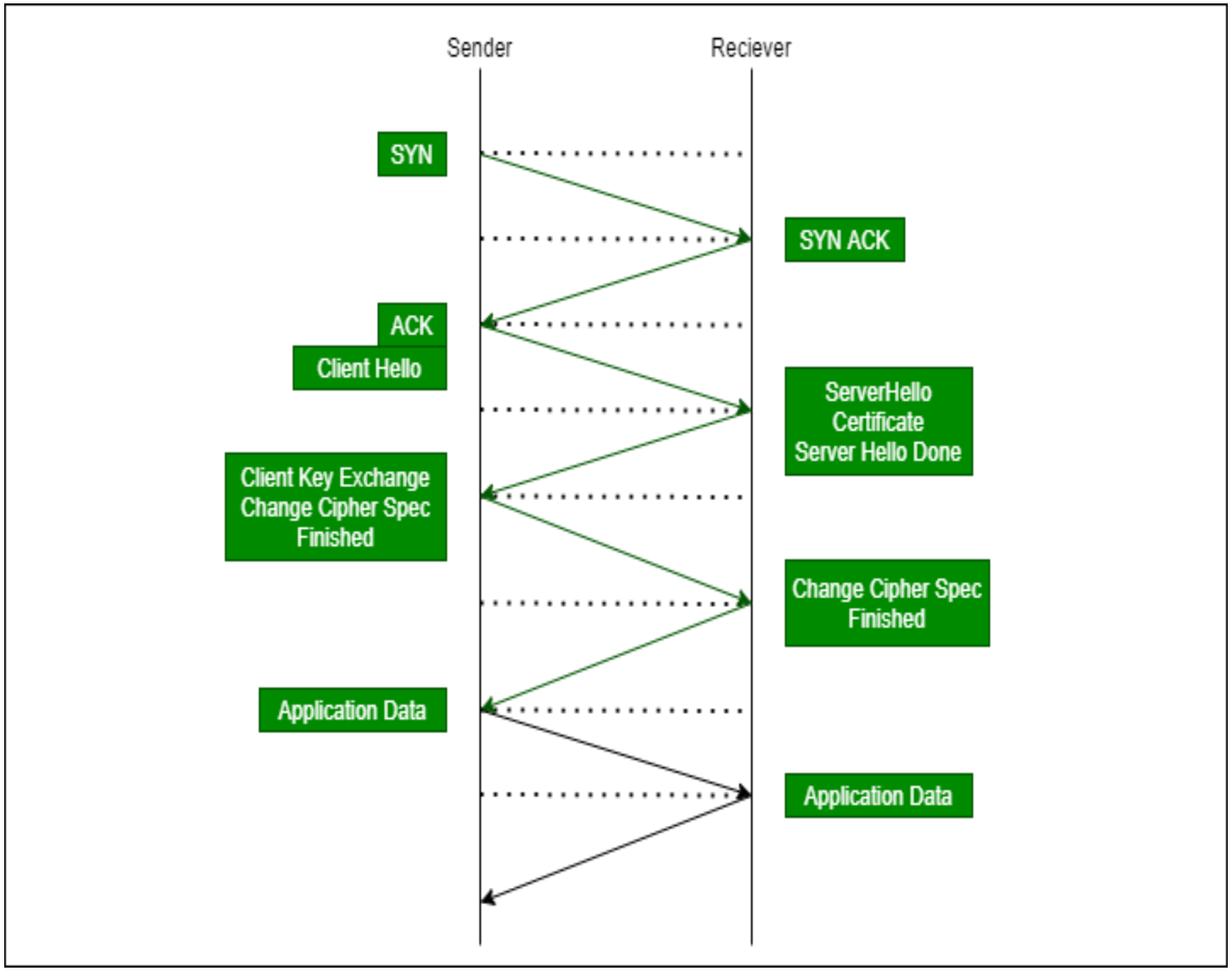
Once the TCP connection is established, the TLS handshake begins.

2. TLS Handshake Process

- **ClientHello**:
  - The client sends a "Client Hello" message to the server.
  - It includes supported TLS versions, cipher suites, and a randomly generated number.
- **ServerHello + Certificate**:
  - The server responds with a "Server Hello" message.
  - It selects the TLS version and cipher suite from the client's options.
  - It provides its digital certificate (X.509) to prove its identity.
  - The server may also send a "Server Hello Done" message to indicate it has finished its part.
- **Client Key Exchange + Change Cipher Spec + Finished**:
  - The client verifies the server's certificate.
  - It generates a "pre-master secret" (for symmetric encryption) and encrypts it using the server's public key.
  - It sends a "Change Cipher Spec" message to indicate that further communication will be encrypted.
  - A "Finished" message is sent to confirm that encryption keys are set.
- **Server Change Cipher Spec + Finished**:
  - The server decrypts the pre-master secret and derives the session key.
  - It sends a "Change Cipher Spec" message to indicate that it will now use encryption.
  - A "Finished" message is sent to confirm that encryption is enabled.

3. Secure Communication Begins

- After a successful TLS handshake:
- The client and server can securely exchange **encrypted application data** (e.g., HTTP requests and responses in HTTPS).
- This handshake process ensures **confidentiality, integrity, and authentication** for secure communication over the internet.


Sender
Reciever
SYN
SYN ACK
ACK
Client Hello
ServerHello
Certificate
Server Hello Done
Client Key Exchange
Change Cipher Spec
Finished
Change Cipher Spec
Finished
Application Data
Application Data

Alert protocol in TLS

Field	Description
Alert Level	Indicates whether the alert is **Warning (1) or Fatal (2)**.
Alert Description	Specifies the exact reason for the alert (e.g., bad_record_mac, handshake_failure).
Encryption	If the TLS handshake is completed, alerts are **sent encrypted**. Otherwise, they are sent in plaintext.

TLS Alert Protocol in TLS 1.3 vs TLS 1.2

Feature	TLS 1.2	TLS 1.3
Alert Messages	Many alerts exist	Fewer alerts (simplified protocol)
Handshake Failure	Sent in plaintext	Sent encrypted after ServerHello
Certificate Errors	Many certificate-related alerts	Uses bad_certificate for most cases
Session Termination	close_notify required	close_notify required

Responsibilities of TLS

TLS is essential for secure communication over the internet, ensuring:

1. **Encryption (Confidentiality)** → Prevents eavesdropping.
2. **Authentication** → Verifies identity using certificates.
3. **Data Integrity** → Prevents tampering.
4. **Secure Key Exchange** → Ensures strong encryption keys.
5. **Efficient Session Management** → Reduces handshake delays.
6. **Defense Against Attacks** → Protects against MITM, replay, and downgrade attacks.

Wireless Transport Layer Security (WTLS)

- **Wireless Transport Layer Security (WTLS)** is a security protocol designed specifically for **wireless networks and mobile devices**. It is based on **TLS (Transport Layer Security)** but optimized for the **Wireless Application Protocol (WAP)** used in mobile communication.
- WTLS ensures **secure communication** between **mobile devices** (such as early WAP-enabled phones) and **wireless servers** by providing:
    1. **Confidentiality** (Encryption)
    2. **Integrity** (Data protection)
    3. **Authentication** (Identity verification)
- Since WAP is used over **wireless networks (GSM, GPRS, CDMA, etc.)**, it faces security risks:
  Data interception → Wireless signals can be intercepted by hackers.
  Man-in-the-Middle (MITM) attacks → Attackers can alter data in transit.
  Limited mobile device resources → Standard TLS is too heavy for early mobile devices.

WTLS solves these issues by offering **lightweight encryption, authentication, and data integrity** specifically optimized for mobile networks.

WAP (Wireless Application Protocol)

- **WAP (Wireless Application Protocol)** is a **communication protocol** designed to enable **mobile devices** (such as early cell phones and PDAs) to access the internet and web services.
- Developed in the **late 1990s**, WAP allowed basic web browsing and online services over **2G and early 3G networks**.
- It was mainly used when mobile devices had **small screens, low bandwidth, and limited processing power**.
- **Think of WAP as an early version of mobile internet before smartphones and full-fledged browsers like Chrome and Safari.**

- WAP was used for:

1. **Mobile Web Browsing** → Allowed access to simplified web pages (WML instead of HTML).
2. **Mobile Banking** → Users could check balances and transfer money on WAP-enabled sites.
3. **Email and Messaging** → Provided access to mobile email and early chat services.
4. **Online News and Weather Updates** → Enabled real-time updates for mobile users.
5. **E-commerce and Online Services** → Allowed basic transactions via mobile phones.

- **Example**: In the early 2000s, Nokia and Motorola phones had a **WAP browser** to access web-based services like Yahoo! or mobile banking.

WTLS Security Features

Feature	Function in WAP
Encryption (Confidentiality)	Ensures that data exchanged between the mobile device and the WAP server is encrypted. Prevents eavesdropping.
Authentication	Verifies the identity of the server (and optionally the client) using digital certificates or shared keys. Prevents MITM attacks.
Data Integrity	Uses cryptographic hashes (e.g., SHA-1, MD5) to ensure that messages are not altered in transit.
Optimized for Wireless Networks	Designed for **low bandwidth, high latency, and mobile devices** with limited power.

WTLS Handshake Process in WAP

When a WAP device connects to a secure service (e.g., mobile banking), WTLS follows these steps:

1. **ClientHello** → The mobile device sends a handshake request with supported encryption methods.
2. **ServerHello + Certificate** → The WAP server responds with a chosen encryption method and a certificate.
3. **Key Exchange** → A shared encryption key is securely exchanged between the client and server.
4. **Change Cipher Spec** → Encryption starts for secure communication.
5. **Finished** → Secure WAP session is established, and data can be exchanged securely.

WAP itself became obsolete →

1. Modern smartphones use **full HTML browsers (Chrome, Safari)** instead of WAP.
2. **Faster Mobile Networks (3G, 4G, 5G)** → Higher speeds made lightweight protocols like WTLS unnecessary.
3. **Better Security (HTTPS + TLS 1.3)** → Stronger encryption replaced WTLS for mobile security.
4. **WTLS had security flaws** → Vulnerabilities in its key exchange made it less secure than modern TLS.

Today, HTTPS with TLS 1.3 is used instead of WTLS for mobile security.

- Although WTLS was useful for early mobile security, it had limitations:
  1. **Weak Encryption** → Vulnerabilities in older algorithms (e.g., RC5, SHA-1).
  2. **"WAP Gap" Security Flaw** → Data was decrypted at the WAP Gateway, exposing it to attacks.
  3. **Slower Networks Replaced by 3G/4G** → Faster mobile networks made WTLS unnecessary.
  4. **Better Alternatives (TLS 1.3, HTTPS)** → Modern TLS replaced WTLS for mobile security.

Common Security Threats & countermeasures

Malware (Viruses, Worms, Trojans, Ransomware)

- **Threat:** Malicious software designed to disrupt, damage, or gain unauthorized access to systems.
- **Countermeasures:**
  - Use up-to-date antivirus and anti-malware software.
  - Implement endpoint detection and response (EDR).
  - Educate users about phishing and suspicious downloads.

Phishing Attacks

- **Threat:** Cybercriminals trick users into revealing sensitive information (passwords, credit card details) via fake emails or websites.
- **Countermeasures:**
  - Enable email filtering and anti-phishing tools.
  - Conduct user awareness training.
  - Use multi-factor authentication (MFA).

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

- **Threat:** Attackers overwhelm a network or system with excessive traffic, rendering it inaccessible.
- **Countermeasures:**
  - Use firewalls and intrusion prevention systems (IPS).
  - Deploy traffic monitoring and rate-limiting solutions.
  - Implement content delivery networks (CDNs) and DDoS protection services.

Man-in-the-Middle (MitM) Attacks

- **Threat:** Attackers intercept and manipulate communication between two parties without their knowledge.
- **Countermeasures:**
  - Use encrypted communication protocols (SSL/TLS, VPNs).
  - Enable strong authentication mechanisms.
  - Regularly update and patch network devices.

SQL Injection & Cross-Site Scripting (XSS)

- **Threat:** Attackers exploit vulnerabilities in web applications to gain unauthorized access or manipulate data.
- **Countermeasures:**
  - Implement input validation and sanitization.
  - Use web application firewalls (WAFs).
  - Conduct regular security testing and code reviews.

Zero-Day Exploits

- **Threat:** Attackers exploit unknown software vulnerabilities before a patch is available.
- **Countermeasures:**
  - Maintain a robust vulnerability management program.
  - Apply security patches and updates as soon as they are available.
  - Use behavioral-based threat detection tools.

Insider Threats

- **Threat:** Employees or trusted individuals misuse their access to steal data.
- **Countermeasures:**
  - Implement role-based access control (RBAC) and least privilege principles.
  - Monitor user activities and enforce logging/auditing policies.
  - Conduct regular security awareness training.

Password Attacks (Brute Force, Credential Stuffing)

- **Threat:** Attackers attempt to guess or use stolen credentials to gain unauthorized access.
- **Countermeasures:**
  - Enforce strong password policies and use MFA.
  - Implement account lockout policies after repeated failed attempts.
  - Use password managers and avoid credential reuse.

General Network Security Countermeasures

Firewalls: Filter network traffic based on predefined rules.

Intrusion Detection and Prevention Systems (IDS/IPS): Monitor and respond to suspicious activities.

Encryption: Protect data at rest and in transit using strong encryption methods.

Network Segmentation: Limit access to critical systems by isolating them from other networks.

Security Information and Event Management (SIEM): Aggregate logs and analyze threats in real-time.

Regular Security Audits and Penetration Testing: Identify vulnerabilities and address them proactively.

By implementing these countermeasures, organizations can significantly reduce the risk of security breaches and maintain a secure network environment.

Timeline of SSL to TLS Transition

SSL 2.0 (1995) – Deprecated due to major security flaws.

SSL 3.0 (1996) – Vulnerable to POODLE attack; deprecated in 2015.

TLS 1.0 (1999) – First replacement for SSL, but now deprecated.

TLS 1.1 (2006) – Improved security but deprecated in 2020.

TLS 1.2 (2008) – Secure and widely used today.

TLS 1.3 (2018) – More secure and faster, removing outdated cryptographic features.