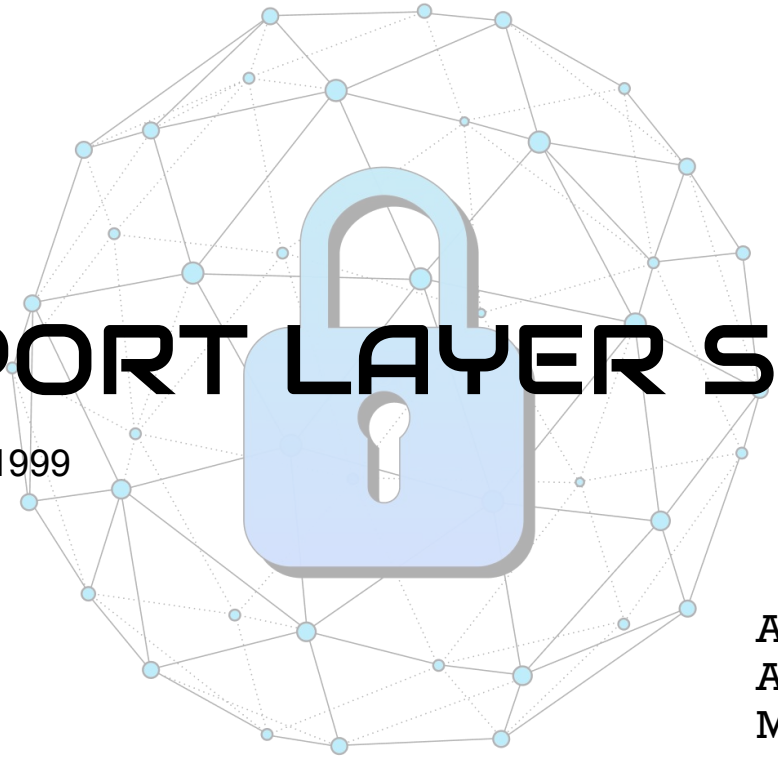
TRANSPORT LAYER SECURITY



Making hackers cry since 1999

Abinaya B - 22Z202
Abinaya Suresh - 22Z203
M S Padmavathi - 22Z234
Snesha B - 22Z260
Thakshin Kumar T - 22Z266
Dharshini V - 23Z438

Introduction to TLS and Importance

Thakshin Kumar
22Z266

What is TLS and Why it is used in cryptography?

TLS (Transport Layer Security) is a cryptographic protocol that secures communication over a network by encrypting data and ensuring its integrity and authenticity. It's the successor to SSL.

TLS is used in cryptography to **secure data transmission** by:

1. **Encrypting** data so attackers can't read it.
2. **Authenticating** the communicating parties (e.g., verifying a website's identity).
3. **Ensuring integrity**, so data isn't altered during transfer.

Brief history: From SSL to TLS

1. **SSL (Secure Sockets Layer)** is an older security protocol developed by Netscape in mid of 1990s to encrypt data between a client and a server. **SSL 2.0 (1995)** was the first public version but had security flaws. **SSL 3.0 (1996)** fixed many issues and became widely used.

2. **TLS (Transport Layer Security)** was found in 1999 is its improved successor, offering stronger encryption, better performance, and enhanced security against modern attacks. (**TLS 1.1 in 2006, TLS 1.2 in 2008, TLS 1.3 in 2018**)

3. **TLS is preferred over SSL** because SSL has known vulnerabilities and is no longer considered secure, while TLS provides updated cryptographic algorithms and safer key exchange methods.

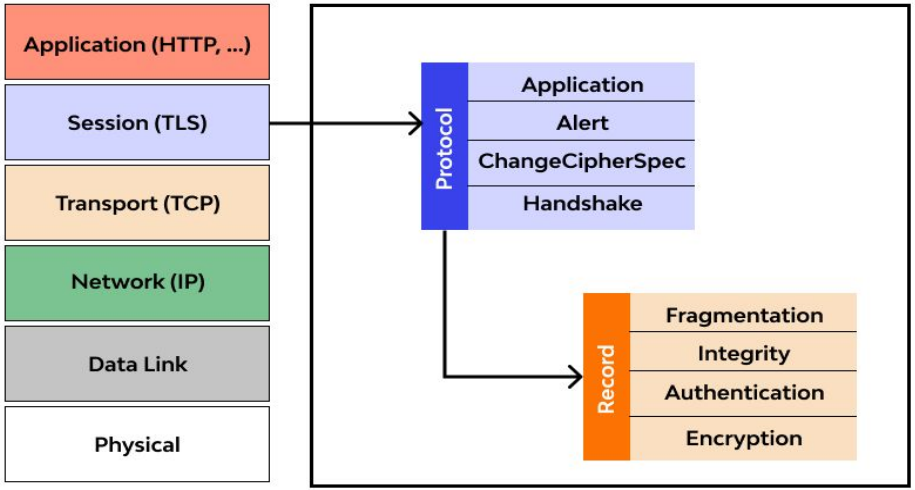
Real World Applications of TLS

1. **Web browsing:** Secures HTTPS connections between browsers and websites.
2. **Email:** Protects communication in protocols like SMTP, IMAP, and POP3.
3. **Online banking and shopping:** Encrypts financial transactions and personal data.
4. **VPNs:** Secures data transmission over virtual private networks.
5. **Instant messaging and VoIP:** Ensures privacy in apps like WhatsApp, Signal, and Skype.
6. **APIs and web services:** Protects data exchanged between servers and applications.

Architecture and Working

Abinaya B - 22Z202

Architecture

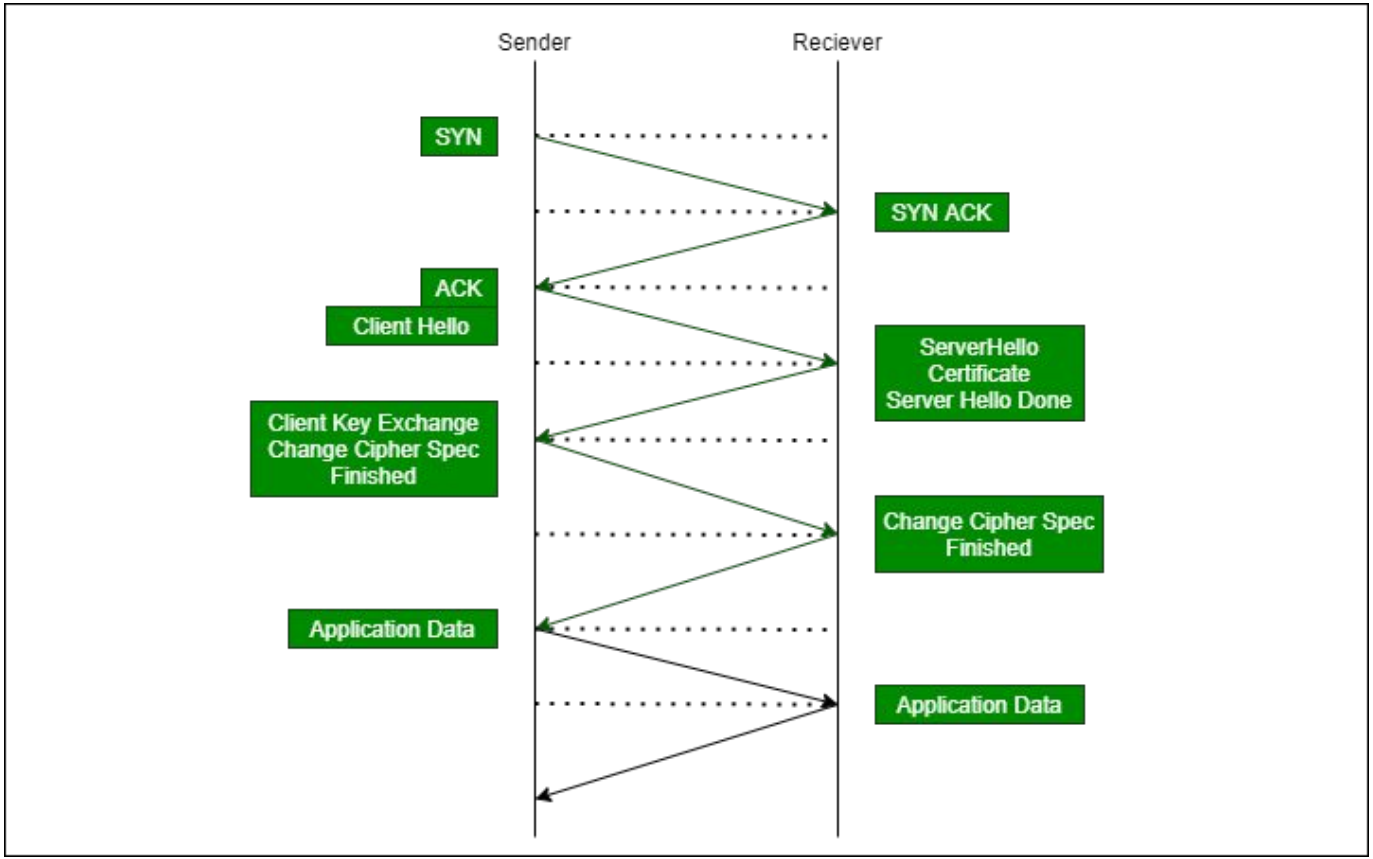


1. **Protocol Layer** – manages handshake, alerts, and cipher changes

2. **Record Layer** – handles encryption, integrity, and authentication

Link in the speaker notes

Working


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

Working

Server-End

1. Fragmentation
2. Compression
3. Authentication and Integrity(MAC)
4. Encryption and transmission

Receiver-End

1. Decryption
2. MAC verification
3. Integrity check

TLS Handshake Protocol

How a secure connection is established between client and server

M.S.Padmavathi
22Z234

Introduction

Transport Layer Security, or TLS, is a cryptographic protocol that provides **secure communication over the internet**.

It ensures that the data exchanged between a client and a server remains **private, authenticated, and tamper-proof**.

TLS evolved from the older SSL protocol, and today it’s used in applications like **web browsing (HTTPS)**, **emails**, and **VPN connections**.

So basically, TLS is what keeps our online interactions safe.

What is TLS Handshake?

Definition:
The **TLS Handshake Protocol** is the process that initiates communication between a client (like a web browser) and a server (like a website), allowing them to:

- Authenticate each other's identity.
- Agree on encryption methods.
- Generate **session keys** for secure data transfer.

Purpose:

1. To ensure **confidentiality** (data is encrypted).
2. To ensure **integrity** (data isn't altered).
3. To ensure **authentication** (the server is genuine).

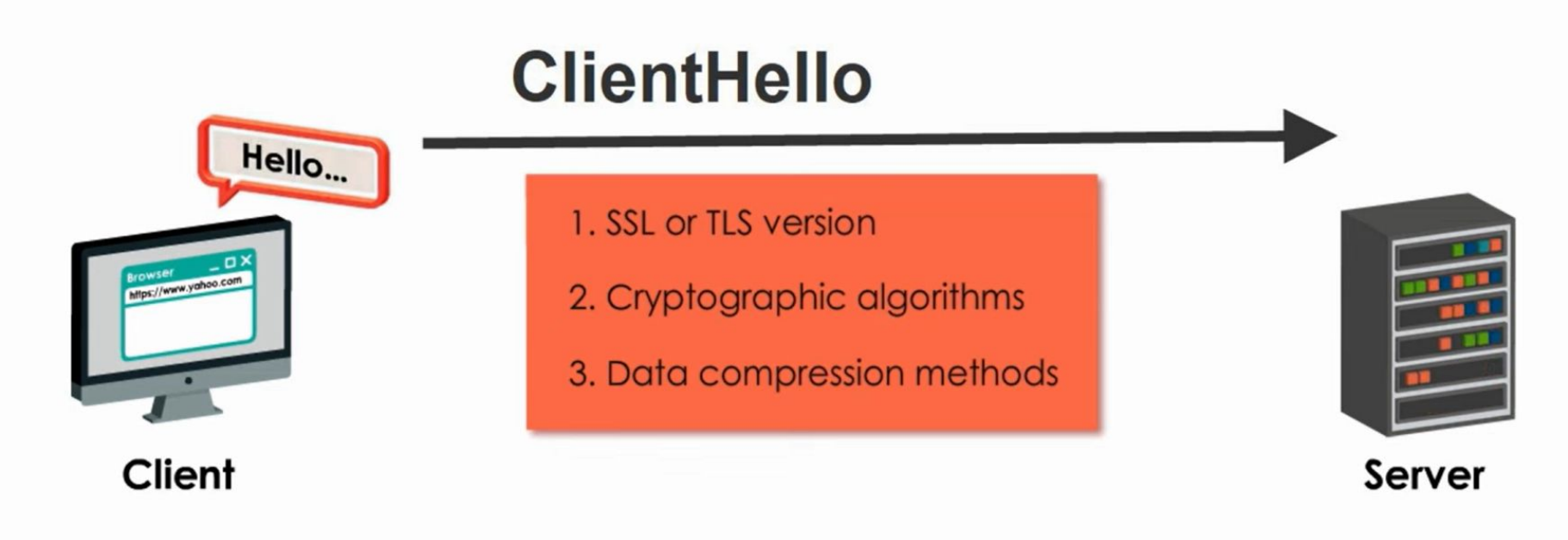
TLS in the OSI Model

TLS — despite its name — actually operates **between** two OSI layers:

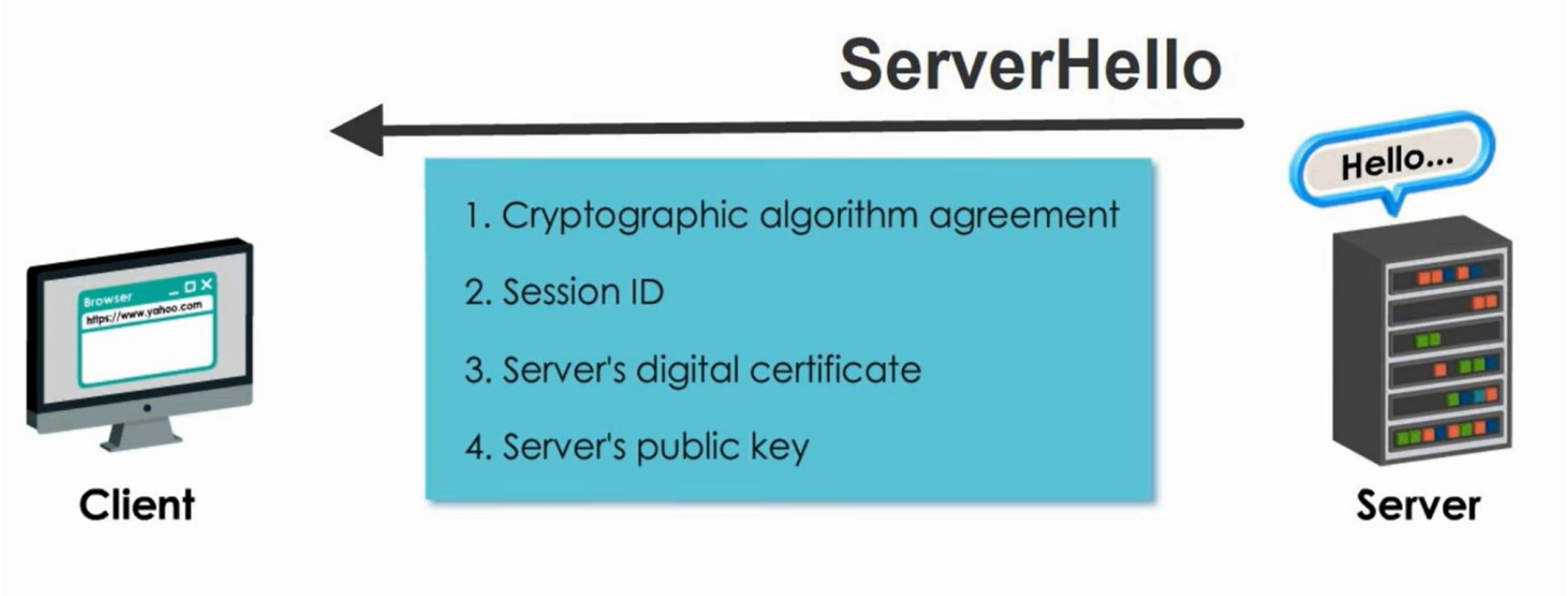
OSI Layer	Layer Name	TLS Relation
Layer 5	**Session Layer**	TLS helps establish, manage, and terminate secure sessions.
Layer 4	**Transport Layer**	TLS runs *on top of* TCP, providing encryption and authentication for the data transmitted.

Steps involved in Handshake

Step 1



Step 2


ServerHello
Hello...
1. Cryptographic algorithm agreement
2. Session ID
3. Server's digital certificate
4. Server's public key
Browser
https://www.yahoo.com
Client
Server

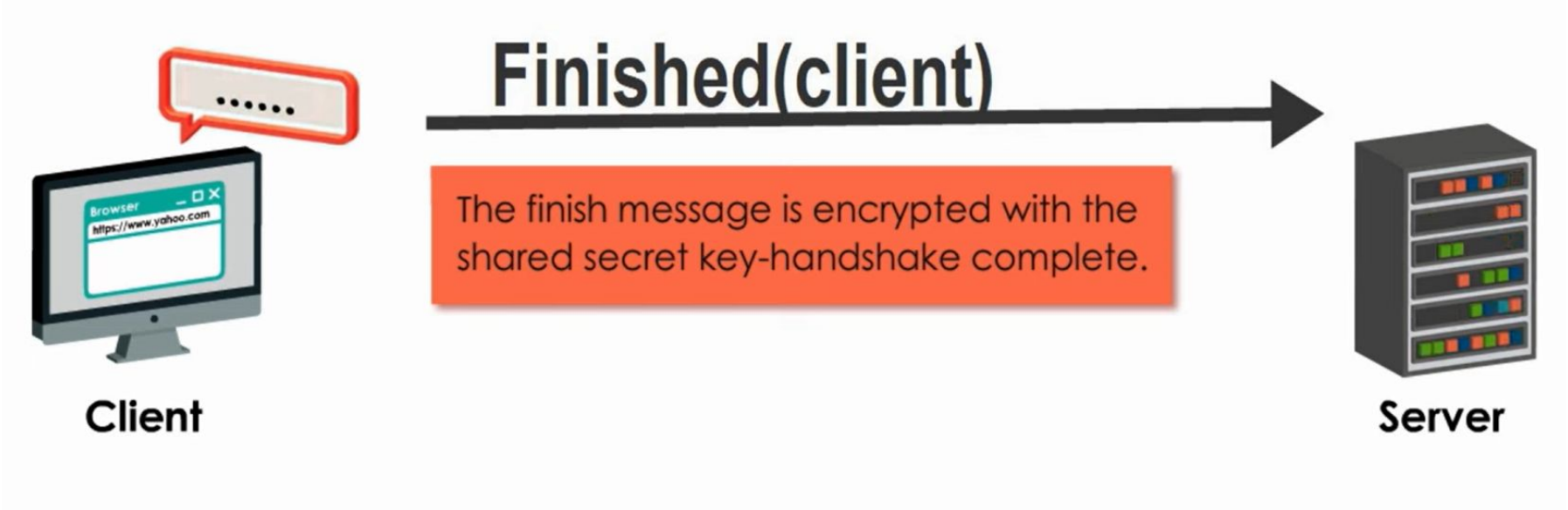
Step 3


Certificate Authority(CA)
CERTIFICATE
Browser
https://www.yahoo.com
Client
Server

Step 4


ClientKeyExchange
A shared secret key encrypted with the server's public key.
Browser
https://www.yahoo.com
Client
Server

Step 5


Finished(client)
The finish message is encrypted with the shared secret key-handshake complete.
Browser
https://www.yahoo.com
Client
Server

Step 6


Finished(server)
The finish message is encrypted with the shared secret key-handshake complete.
Browser
https://www.yahoo.com
Client
Server


Browser
https://www.yahoo.com
Exchange Messages
Client
Server

Key Elements Used in Handshake

1. **Certificates** – Provided by Certificate Authorities (CA) to prove the server's authenticity.

2. **Public and Private Keys** – Used in asymmetric cryptography for key exchange.

3. **Session Keys** – A temporary symmetric key used for fast and secure encryption after the handshake.

TLS Encryption & Authentication

Abinaya Suresh
22Z203

The Core Components

TLS ensures a secure connection through three key pillars:

1. **Cipher Suites**
    - The cryptographic "toolkits."
2. **Certificates & PKI**
    - The digital identity system.
3. **Digital Signatures & Authentication**
    - The method for verifying trust.

Cipher Suites

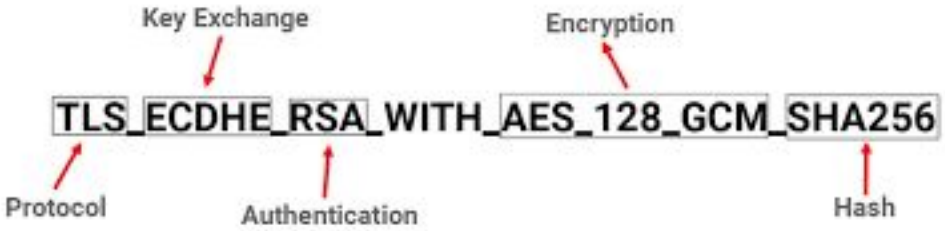
What is a Cipher Suite?

- A set of algorithms that the client and server agree to use.
- The client sends a list of suites it supports, and the server picks one.

What does it define?

- **Key Exchange Algorithm** (e.g., RSA, ECC)
- **Bulk Encryption Algorithm** (e.g., AES, ChaCha20)
- **Message Authentication Code (MAC)** for data integrity.

Insides Cipher Suites



ECDHE: The Key Exchange algorithm. Defines how the session key is created.

RSA: The Authentication algorithm. Used to prove the server's identity with its certificate.

AES_128_GCM: The Bulk Encryption algorithm used for the actual data.

SHA256: The Message Authentication Code (MAC) algorithm, which ensures data integrity.

Certificates & PKI

- **Digital Certificates**
  - The server sends its certificate to the client during the handshake.
  - Acts as a digital "ID card" to prove the server's identity.
- **Public Key Infrastructure (PKI)**
  - The system of trust that makes certificates work.
  - It relies on trusted **Certificate Authorities (CAs)** to issue and sign certificates.
  - Your browser trusts the CA, and therefore can trust the server's certificate that the CA has signed.

Inside a Digital Certificate

Subject: Who the certificate belongs to (e.g., www.google.com).

Issuer: Which Certificate Authority (CA) issued it (e.g., Google Trust Services).

Validity Period: The "Valid From" and "Valid To" dates.

Public Key: The subject's public key, used for the initial key exchange.

Issuer's Digital Signature: The cryptographic signature from the CA to prove it's authentic.

Digital Signatures & Authentication

- **What is a Digital Signature?**
  - A cryptographic stamp of approval from a Certificate Authority on the server's certificate.
  - It verifies the certificate's authenticity and ensures it hasn't been altered.
- **The Authentication Process**
  - Your browser verifies this digital signature using the CA's public key.
  - Successful validation confirms that the server is genuine.
  - This is the primary defense against man-in-the-middle attacks.

TLS Versions & Improvements

Dharshini V (23z438)

SSL vs TLS

- **SSL (Secure Sockets Layer)** was the **original protocol** for securing data over the internet.
- Developed in the 1990s, but it had **many vulnerabilities** (e.g., POODLE attack).
- To address these issues, **TLS (Transport Layer Security)** was introduced as an **improved and more secure version** of SSL.
- TLS is now the **standard** for encrypting communications in HTTPS, emails, and VPNs.

Evolution of TLS

- TLS evolved from SSL (Secure Sockets Layer)
  - **SSL 2.0 (1995)** – insecure, deprecated
  - **SSL 3.0 (1996)** – improvements but still flawed (POODLE attack)
  - **TLS 1.0 (1999)** – replaced SSL
  - **TLS 1.1 (2006)** – better protection
  - **TLS 1.2 (2008)** – widely used and secure (until recently)
  - **TLS 1.3 (2018)** – modern, faster, more secure

1. **Security Algorithms**
   - TLS 1.0 & 1.1 supported weak ciphers (like RC4, MD5).
   - TLS 1.2 introduced stronger options (AES, SHA-256).
   - TLS 1.3 removed all weak algorithms entirely.

2. **Handshake Process**
   - TLS 1.2 and earlier needed **two round trips** to establish a connection.
   - TLS 1.3 reduced it to **one**, making it **faster**.

3. **Forward Secrecy**
   - Optional in TLS 1.2.
   - **Mandatory in TLS 1.3**, ensuring past data remains secure even if keys are compromised.

4. **Simplified Protocol**
   - TLS 1.3 removed outdated features (RSA key exchange, static keys, compression) to reduce attack surfaces.

Improvements in TLS 1.3

- **Fewer Handshakes → Faster Performance**
  Establishes a secure connection in a single round trip, improving website loading times.

- **Forward Secrecy by Default**
  Each session uses unique keys, so even if one key is stolen, past sessions remain protected.

- **Stronger Encryption**
  Supports only modern and secure algorithms (AES-GCM, ChaCha20-Poly1305).

- **Simplified and More Secure Design**
  Removes legacy options that caused confusion and security risks in older versions.

TLS ATTACKS, CHALLENGES AND FUTURE

SNESHA B - 22Z260

TLS ATTACKS

1. Heartbleed (2014)

What: Vulnerability in OpenSSL's Heartbeat extension.
Impact: Leaked server memory — private keys, passwords, session data could be exposed.
Cause: Buffer over-read (implementation bug in OpenSSL).
Mitigation: Update OpenSSL to patched versions, rotate/revoke affected certificates, and adopt secure coding & testing.

2. POODLE (2014)

What: Padding oracle attack that targets SSL 3.0 (and TLS via forced downgrade).
Impact: Allows attackers to decrypt parts of HTTPS traffic after forcing a downgrade.
Cause: Flawed padding handling in SSL 3.0 and protocol fallback behavior.
Mitigation: Disable SSL 3.0 and protocol fallback; enforce TLS 1.2/1.3 only.

3. BEAST (2011)

What: Browser Exploit Against SSL/TLS targeting TLS 1.0 using CBC mode.
Impact: Can decrypt HTTP cookies/session tokens under certain conditions.
Cause: Weakness in TLS 1.0 CBC implementation and block cipher handling.
Mitigation: Disable TLS 1.0, prefer AEAD ciphers (e.g., AES-GCM, ChaCha20) and use TLS 1.2/1.3.

TLS CHALLENGES

1. Legacy Support

- Older systems, devices, and browsers still rely on **TLS 1.0 or 1.1**, which are insecure.
- Organizations often maintain compatibility with outdated versions, leaving systems **vulnerable to attacks**.

2. Certificate Management

- **Expired, self-signed, or misconfigured certificates** break trust chains.
- Manual renewal and deployment increase human error.
- Large-scale infrastructures need **automated certificate lifecycle management**.

3. Implementation Bugs

- TLS libraries like **OpenSSL** and **GnuTLS** are complex and prone to vulnerabilities.
- Even small coding errors can lead to severe security breaches (e.g., Heartbleed).
- Continuous testing, code audits, and security reviews are essential.

4. Performance Overhead

- Handshakes and encryption operations increase **latency** and **CPU usage**, especially in high-traffic web services.
- TLS 1.3 has improved performance, but older versions still cause slowdowns.

TLS FUTURE TRENDS

1. Post-Quantum Cryptography (PQC)

- Quantum computers could break traditional RSA and ECC algorithms.
- Future TLS versions will include **quantum-safe key exchange** and **digital signatures**.
- NIST has already selected PQC algorithms (e.g., Kyber, Dilithium) for standardization.

2. Encrypted ClientHello (ECH)

- In TLS 1.3+, **ECH** encrypts the SNI (Server Name Indication) field, hiding which website a user is connecting to.
- Improves privacy against network surveillance and censorship.

3. Automation & Zero-Touch Management

- Large-scale deployments will rely on **automated certificate management** and **security policy enforcement**.
- Integration with DevOps tools ensures continuous compliance.

4. Lightweight TLS for IoT Devices

- TLS adaptations for **resource-constrained IoT systems** (e.g., TLS-PSK and DTLS).
- Focus on balancing **security** with **low power and memory consumption**.

Questions!!

1. Why is TLS preferred over SSL for securing network communication?
   a. TLS is faster than SSL
   b. TLS provides better security and is an upgraded version of SSL
   c. TLS is older and more stable than SSL
   d. SSL supports more encryption algorithms than TLS
2. TLS operates between which two layers in the network model?
   a. Network and Data Link
   b. Transport and Application
   c. Application and Physical
   d. Session and Network

Questions!!

3. If an attacker manages to trick a user into connecting to a fake website that uses HTTPS but a self-signed certificate, what's the real risk?

a. The data is still safe because it's encrypted
b. The attacker could decrypt or alter information because the connection isn't truly authenticated
c. Nothing — HTTPS always means secure
d. The browser will automatically detect and fix it

4. Which of the following statements best describes the main improvement in TLS 1.3 compared to TLS 1.2?

a. It uses weaker encryption algorithms for compatibility
b. It requires two round trips for the handshake process
c. It introduces faster connection setup with fewer handshake.
d. It reintroduces SSL for backward compatibility

Questions!!

5. Why is Post-Quantum Cryptography important for the future of TLS?

a. It makes TLS compatible with IoT devices.
b. It increases certificate expiry time.
c. It protects encrypted data from quantum computer attacks.
d. It reduces encryption overhead for web servers.

Answers!!

1. B) TLS provides better security and is an upgraded version of SSL
2. B) Transport and Application
3. B) The attacker could decrypt or alter information because the connection isn’t truly authenticated
4. C) It introduces faster connection setup with fewer handshakes
5. C) It protects encrypted data from quantum computer attacks

THANK YOU!!