TLS Architecture

TLS Handshake Protocol

SRIHARI K K
22Z263

TLS Handshake Protocol

- The TLS handshake is the process that establishes a secure connection between a client and a server.

Detailed Steps:

1. Client sends a 'Client Hello' message proposing TLS version and supported cipher suites.

2. Server replies with a 'Server Hello' confirming the TLS version and cipher suite.

3. Server sends its digital certificate for authentication.

4. (Optional) Client sends its certificate for mutual authentication.

5. Both sides exchange keys to establish a shared secret.

6. Secure communication begins once the handshake is complete.

Negotiation, Authentication & Key Exchange:

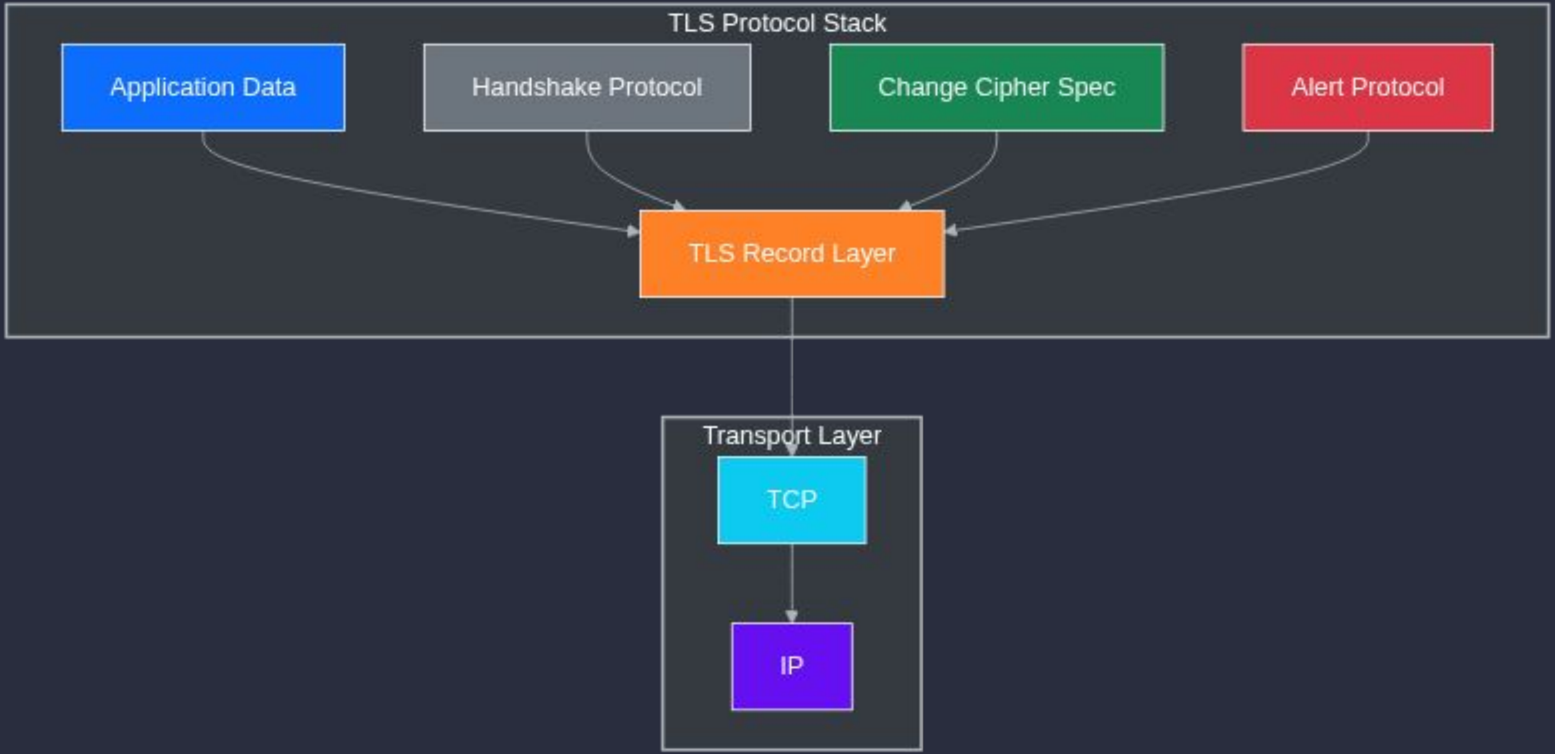
- During the handshake, both the client and server negotiate the TLS version and cipher suite to use.
- The server proves its identity using a digital certificate issued by a trusted Certificate Authority (CA).
- Optionally, the client can also be authenticated with its own certificate.
- Key exchange mechanisms like RSA, Diffie-Hellman, or Elliptic Curve Diffie-Hellman (ECDHE) are used to create shared secret keys.
- These keys are then used to encrypt data and ensure integrity throughout the session.
- The handshake is crucial for establishing trust, confidentiality, and secure communication.

TLS Architecture and Components

Introduction to TLS

- ❖ Transport Layer Security (TLS) standard protocol for securing data in transit over networks
- ❖ Purpose: Privacy, integrity, authentication
- ❖ Use cases: HTTPS, email, VoIP, VPN

TLS Architecture Overview



Role of the TLS Record Layer

- Foundation of all TLS operations
- Responsibilities:
  - Fragmentation and reassembly
  - Compression (optional)
  - Encryption
  - Message Authentication Code (MAC)

Confidentiality in the Record Layer

- **Confidentiality**: symmetric encryption (AES, ChaCha20)
- **Integrity**: MAC (HMAC-SHA256, Poly1305)
- Encryption + MAC ≠ Encrypt-then-MAC in TLS 1.3
- Protection against eavesdropping and modification

TLS Protocols: Change Cipher Spec and Alert Mechanisms

PRANEETH M
22z279

Change Cipher Spec – What and Why?

What i s it?

- A small message in TLS that tells the other side:
  —-->" Start using the new encryption now!"

Why is it important?

- Happens after both sides agree on how to secure the connection.

- Makes sure all future messages are encrypted and secure.
- It's a signal to switch from plain communication to secure communication.

How Do They Start Using the New Keys?

Steps:

1. Client sends Change Cipher Spec

2. Client sends an encrypted "Finished" message

3. Server sends Change Cipher Spec

4. Server sends its encrypted "Finished" message

After this:
—----->All communication is encrypted using the agreed settings.

What is the Alert Protocol?

What does it do?

- Sends error or warning messages between client and server.

Two Types of Alerts:

- Warning – Not serious (e.g., connection closing normally)
- Fatal – Serious problem (e.g., wrong certificate) → Connection stops

Why it matters:

- Helps both sides know if something went wrong.
- Makes the connection safer and more reliable.

TLS Certificates and Public Key Infrastructure

PRATISH Joi
22Z247

Certificates and Public Key Infrastructure

Authentication:

- Verifying that the party you're communicating with is genuine and not an imposter.
- TLS uses digital certificates to prove the identity of websites and prevent attacks like man-in-the-middle (MITM).

Digital Certificate:

- Electronic document that proves ownership of a public key.
- It links a public key with information about its owner (organization name, domain name, issuer, expiry date, etc.).
- Certificates follow the X.509 standard.

Role of Certificate Authorities (CAs):

- A Certificate Authority (CA) is a trusted third party that issues and validates certificates.
- Examples: DigiCert, GlobalSign, Let's Encrypt, GoDaddy.

Process:

1. **The website owner requests a certificate from a CA.**
2. **The CA verifies ownership and identity.**
3. **The CA digitally signs the certificate and issues it.**
4. **Browsers trust certificates issued by known CAs.**

Certificates and Public Key Infrastructure

Public Key Infrastructure (PKI):

PKI is the system that manages digital certificates and public-key encryption. It provides a framework for trust between users, servers, and CAs.

PKI handles: Certificate issuance, Certificate verification, Certificate revocation, Key management

PKI Components: Certificate Authorities (CAs), Registration Authorities (RAs), Certificate Repositories, Revocation Lists (CRLs)

Why It Matters:

- Certificates and PKI ensure that:
- You are communicating with the right server.
- Data exchanged is encrypted.
- Attackers can’t impersonate legitimate websites.

Certificate Chain of Trust:TLS certificates are trusted through a **hierarchical structure**:

Root CA – The ultimate trusted authority (built into browsers).
Intermediate CA – Issues certificates on behalf of the Root CA.
End-Entity/Server Certificate – The website’s own certificate.

How application data is transmitted over TLS

RITHVIK K
22z253

How application data is transmitted over TLS

1. Handshake Completion:
A shared session key is established between client and server for secure communication.

2. Data Fragmentation:
Application data is split into small chunks called TLS records.

3. Encryption and Authentication:
Each record is encrypted and authenticated to prevent tampering or eavesdropping.

4. Secure Transmission:
Encrypted records are sent over TCP as unreadable binary data.

5. Decryption at Receiver:
The receiver uses the same session key to decrypt and verify the data.

6. Data Delivery:
Decrypted data is passed to the application layer for normal use.

Modern TLS security features

Modern TLS security features

1. Stronger Encryption
TLS 1.3 uses algorithms like AES-GCM and ChaCha20-Poly1305 to protect data efficiently.

2. Perfect Forward Secrecy
Each session has a unique key, so past data stays safe even if one key is compromised.

3. Simplified Handshake
Reduces connection steps, making TLS faster and less vulnerable.

4. 0-RTT Resumption
Enables quick reconnection without repeating the full handshake.

5. Removal of Weak Algorithms
Outdated methods like RSA key exchange and SHA-1 are eliminated for stronger security.

6. Authenticated Encryption (AEAD)
Combines encryption and integrity checks to ensure data can’t be read or altered.

Common attacks against TLS and best practice defenses

Common TLS Attacks and Defenses

1. Man-in-the-Middle (MITM)
Attack: Intercepts and alters communication between client and server.
Defense: Use valid certificates, enable HTTPS, and apply certificate pinning.

2. Downgrade Attacks
Attack: Forces the connection to use weaker, older TLS versions.
Defense: Disable SSL/TLS 1.0 & 1.1; enforce TLS 1.2 or 1.3 only.

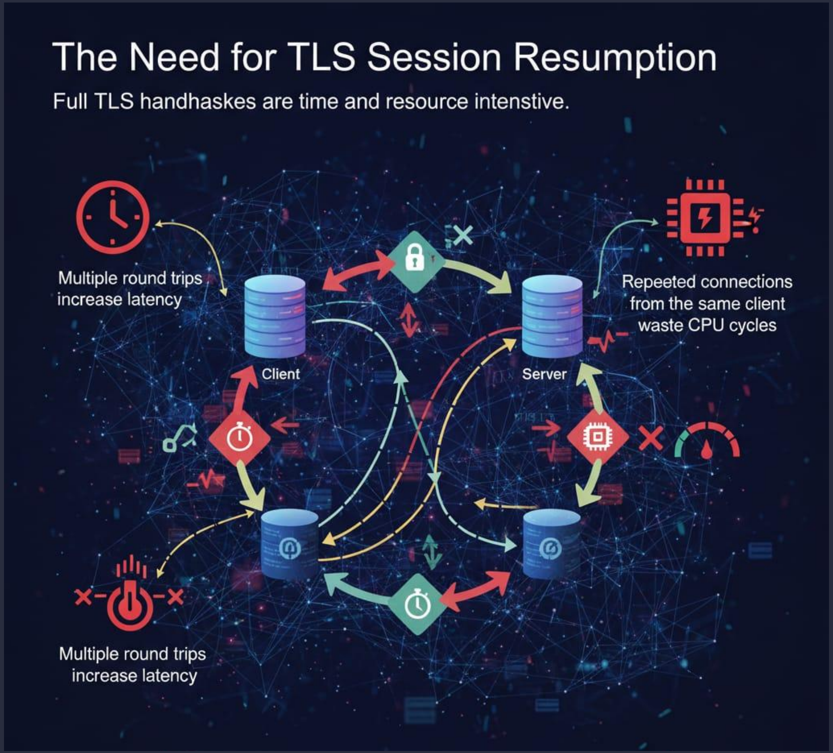
3. Heartbleed
Attack: Exploits a flaw in OpenSSL to leak server memory and sensitive data.
Defense: Update OpenSSL immediately and reissue compromised certificates.

TLS Session Resumption and Performance Optimization

Kapil arif K - 23z432

Need for TLS Session Resumption

- Full TLS handshakes are time and resource intensive.
- Multiple round trips increase latency.
- Repeated connections from the same client waste CPU cycles.
- Goal: Optimize performance while keeping security intact.


The Need for TLS Session Resumption
Full TLS handhaskes are time and resource intenstive.
Multiple round trips increase latency
Client
Server
Repeated connections from the same client waste CPU cycles
Multiple round trips increase latency

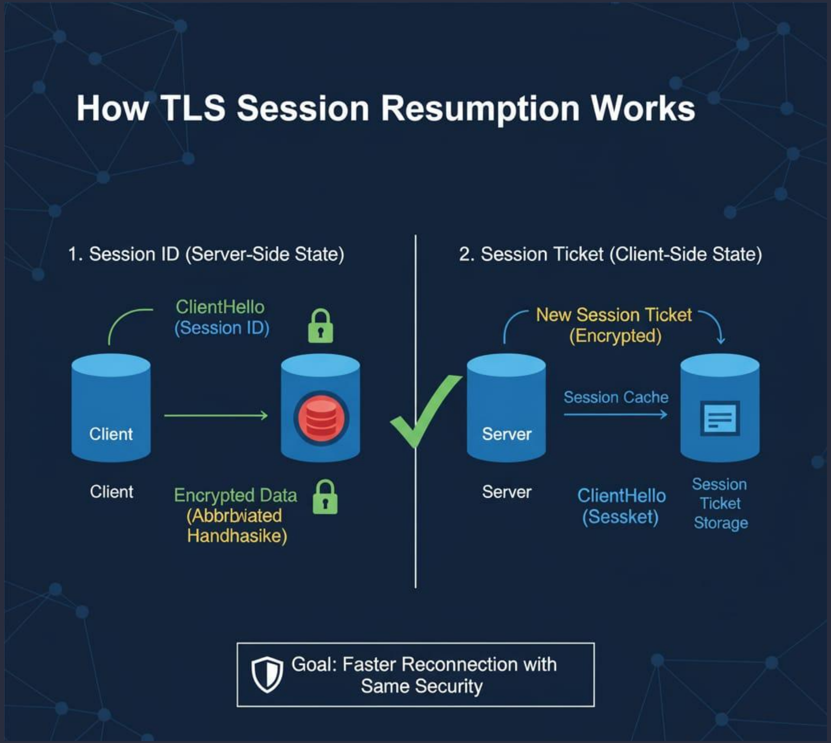
How TLS Session Resumption Works

Two techniques:

i) Session ID – Server remembers session details.

ii) Session Ticket – Client stores encrypted session data.

- Uses abbreviated handshake (fewer round trips).
- Maintains same security level.



How TLS Session Resumption Works
1. Session ID (Server-Side State)
ClientHello (Session ID)
Client
Client
Encrypted Data (Abbrbviated Handhasike)
2. Session Ticket (Client-Side State)
New Session Ticket (Encrypted)
Server
Session Cache
Server
ClientHello (Sessket)
Session Ticket Storage
Goal: Faster Reconnection with Same Security

Performance Impact and Modern Enhancements

- Handshake time reduced by 50–70%.
- Less CPU load and faster page loads.
- Widely used in HTTPS, CDNs, and APIs.
- TLS 1.3 supports 0-RTT for near-instant data transfer.