# HTTPS & TLS - Zusammenfassung

## HTTPS & TLS - Vollständiger Überblick

1. Was ist HTTPS? HTTPS (HyperText Transfer Protocol Secure) ist die sichere Variante von HTTP, bei der die Kommunikation zwischen Client (z. B. Webbrowser) und Server verschlüsselt wird. Grundlage dafür ist TLS (früher: SSL).

2. Was ist TLS? TLS (Transport Layer Security) ist ein Protokoll zur Absicherung der Datenübertragung im Internet. Es bietet:

- **Verschlüsselung** (Schutz vor Abhören)
- **Integrität** (Schutz vor Manipulation)
- **Authentifizierung** (Überprüfung der Identität des Servers)

3. Aufbau einer HTTPS-Verbindung (TLS 1.3)

## Phase 1: Handshake

#### Schritt 1: ClientHello

Der Client sendet:

- `client_random` (32 Byte Zufallszahl)
- `SupportedGroups` (z. B. x25519, secp256r1)
- `KeyShare` (öffentlicher Schlüssel für ECDHE)
- `SupportedVersions` (z. B. TLS 1.3)
- `CipherSuites` (z. B. TLS_AES_128_GCM_SHA256)
- `ALPN` (z. B. http/1.1, h2)
- `SNI` (Server Name Indication – z. B. example.com)

#### Schritt 2: ServerHello

Der Server antwortet mit:

- `server_random` (32 Byte Zufallszahl)
- `CipherSuite` (ausgewählte Verschlüsselung)
- `KeyShare` (sein öffentlicher Schlüssel)
- `SupportedVersion`
- (danach: Wechsel auf verschlüsselte Kommunikation)

4. Shared Secret und Schlüsselableitung

- Beide Seiten berechnen unabhängig: `shared_secret = client_secret * server_public = server_secret * client_public`
- Daraus wird **nicht direkt** der Session Key!
- Stattdessen: `session_key = HKDF(shared_secret, client_random, server_random)`

HKDF = HMAC-based Key Derivation Function. Diese sorgt für:

- unterschiedliche Schlüssel bei jedem Verbindungsaufbau
- Perfect Forward Secrecy (PFS)

5. Verschlüsselte Handshake-Nachrichten (nach ServerHello)

Vom Server:

- `EncryptedExtensions` (z. B. ALPN-Auswahl, SNI-Antwort)
- `Certificate` (z. B. Let's Encrypt)
- `CertificateVerify` (Signatur über bisherigen Handshake)
- `Finished` (MAC zur Absicherung)

Vom Client:

- `Finished` (Bestätigung, dass alles korrekt verarbeitet wurde)

Danach ist der Handshake abgeschlossen.

6. Anwendungsschicht Nach dem Handshake:

- Kommunikation (z. B. `GET /login`) erfolgt als `TLS Record` mit Typ `0x17` (Application Data)
- vollständig verschlüsselt mit dem Session Key

7. Wie werden diese Nachrichten übertragen? Alle TLS-Nachrichten werden:

- über ein einziges **TCP-Verbindung** (meist Port 443)
- als **TLS Records** gesendet:

| Typ | Bedeutung |
|---|---|
| 0x16 | Handshake-Nachricht |
| 0x17 | Application Data (verschlüsselte Nutzdaten) |
| 0x14 | Finished |
| 0x15 | Alert (Fehler etc.) |

8. Warum gibt es client_random & server_random?

- Um die Session Keys **jedes Mal unterschiedlich** zu machen
- Um **Replay-Angriffe** zu verhindern
- Um mehr Entropie in die HKDF einzubringen
- Diese Werte werden **nicht verschlüsselt** übertragen, da sie keine Geheimnisse enthalten

9. Warum wird der shared_secret nicht übertragen?

- Weil die Sicherheit auf dem Prinzip beruht, dass **beide Seiten denselben Schlüssel berechnen können, ohne ihn zu übertragen** (→ Diffie-Hellman)
- shared_secret wird **lokal berechnet**, nie verschickt

10. Was ist ALPN? ALPN (Application Layer Protocol Negotiation) erlaubt es, beim TLS-Handshake auszuhandeln, welches Protokoll auf Anwendungsebene verwendet wird:

- HTTP/1.1 → "http/1.1"
- HTTP/2 → "h2"
- MQTT, ACME etc.

11. Was ist SNI? SNI (Server Name Indication):

- Wird im `ClientHello` gesendet
- Zeigt an, für welchen Domainnamen der Client ein Zertifikat erwartet
- Der Server reagiert in `EncryptedExtensions` mit stillschweigendem OK oder Ablehnung (implizit durch Zertifikatwahl)

SNI ist **nicht verschlüsselt** und daher sichtbar z. B. für Firewalls

12. Wer schickt welche Informationen?

| **Nachricht** | **Von wem** | **Verschlüsselt?** |
|---|---|---|
| ClientHello | Client | ❌ |
| ServerHello | Server | ❌ |
| EncryptedExtensions | Server | ☑ |
| Certificate | Server | ☑ |
| CertificateVerify | Server | ☑ |
| Finished | Server | ☑ |
| Finished | Client | ☑ |
| Application Data | Beide | ☑ |