Hochschule
Bonn-Rhein-Sieg

Prof. Dr. Martin Leischner
Netzwerksysteme und TK

Sicherheit in Netzen
Modul 5: TLS – Transport Layer Security
Teil 2

1. TLS-Einordnung (OSI-Referenzmodell, Geschichte)
2. TLS-Datenübertragung
3. TLS Schlüssel und Algorithmen
4. **TLS-Handshake**
5. **TLS-Implementierung (PDUs, TLS-Sockets)**
6. **Ausgewählte Angriffe gegen TLS**

Prof. Dr. Martin Leischner
Netzwerksysteme und TK

Sicherheit in Netzen
Modul 6: TLS – Transport Layer Security

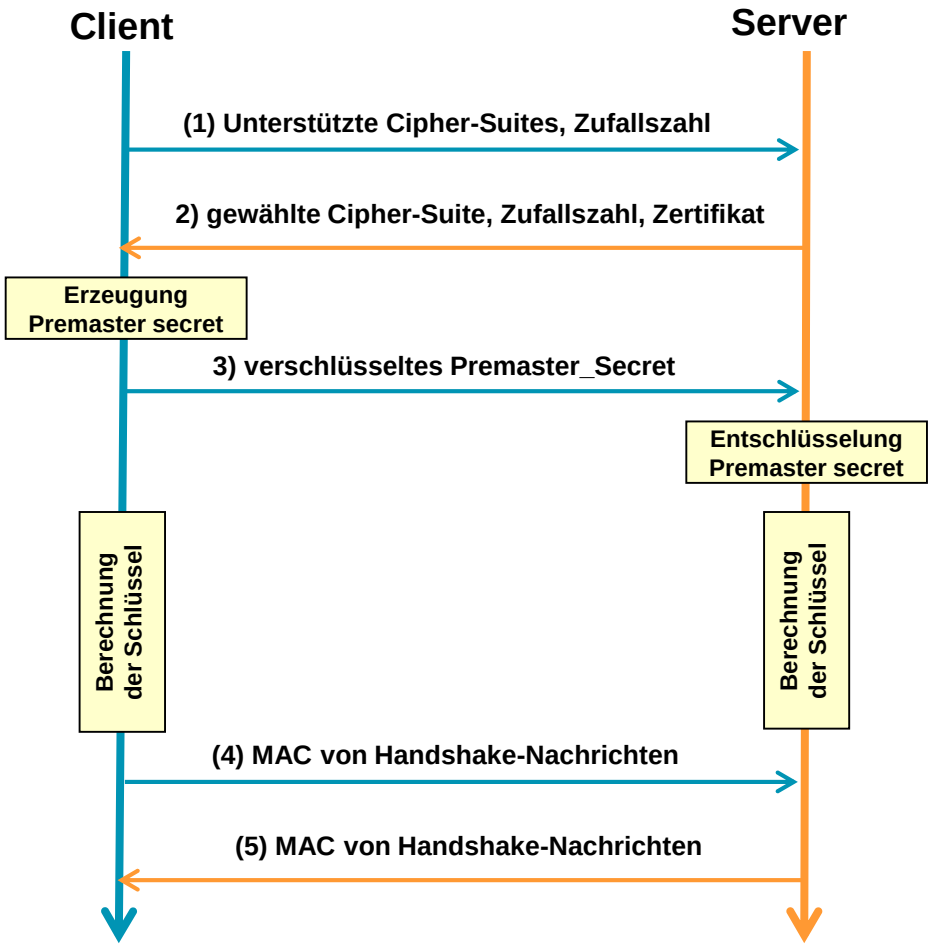
1. TLS-Einordnung (Geschichte, OSI-Referenzmodell)
2. TLS-Datenübertragung
3. TLS Schlüssel und Algorithmen
4. **TLS-Handshake**
5. TLS-Implementierung (PDUs, SSL-Sockets)
6. Ausgewählte Angriffe gegen TLS

TLS-Handshake: Einführung

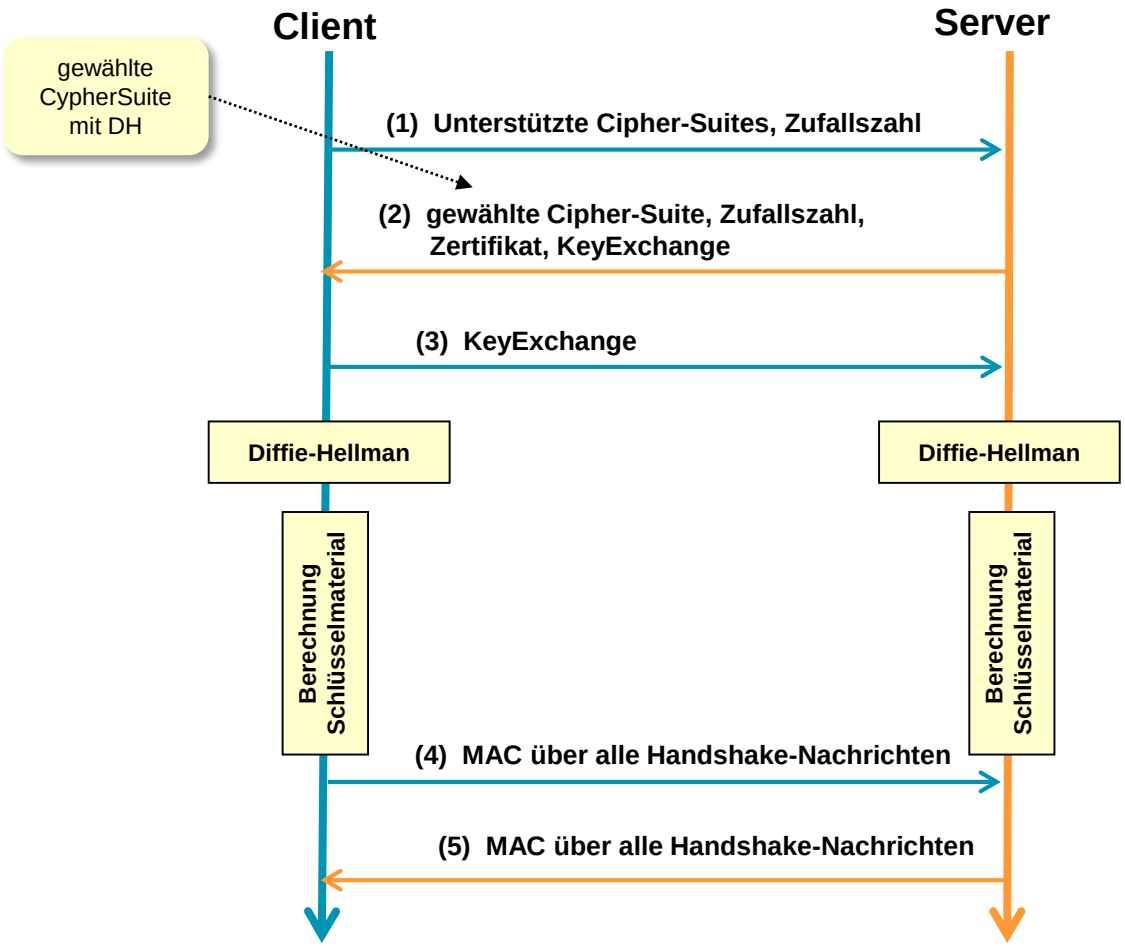
- ***Ziel des Handshakes*** **ist es, das** ***Pre-Master-Secret*** **zu erzeugen, von dem alle weiteren Schlüssel abgeleitet werden**
- **Es werden** ***verschiedene Methoden*** **für die Authentifizierung und den Schlüsselaustausch unterstützt**
- **Für die Erzeugung des Pre-Master-Secrets gibt es (*neben pre-shared keys*) *zwei grundsätzliche Verfahren*:**
  - ***RSA-Verfahren (Woher kommt das Pre-Master-Secret?)***
    - **Das Pre-Master-Secret ist eine vom Client erzeugte Zufallszahl, die verschlüsselt zum Server geschickt wird.**
    - **Das *ClientKeyExchange* ist das verschlüsselte** Pre-Master-Secret
    - **Ein *ServerKeyExchange* ist nicht notwendig und entfällt**
  - ***Diffie-Hellman-Verfahren (Woher kommt das Pre-Master-Secret?)***
    - **Mit dem Diffie-Hellman-Verfahren das Pre-Master-Secret berechnet**
    - **Im ClientKeyExchange und ServerKeyExchange werden die unterschriebenen Diffie-Hellman-Werte (öffentliche Schlüssel) ausgetauscht**
    - **Wenn die Diffie-Hellman-Werte für jede TLS-Verbindung gleich sind, spricht man von *(statischem) Diffie-Hellman***
    - **Wenn die Diffie-Hellman-Werte für jede TLS-Verbindung unterschiedlich sind und damit jedes mal neu berechnet werden, spricht man von *Ephemeral Diffie-Hellman* (→ Perfect Forward Secrecy)**

εφημερος *(ephēmeros)* „für einen Tag“

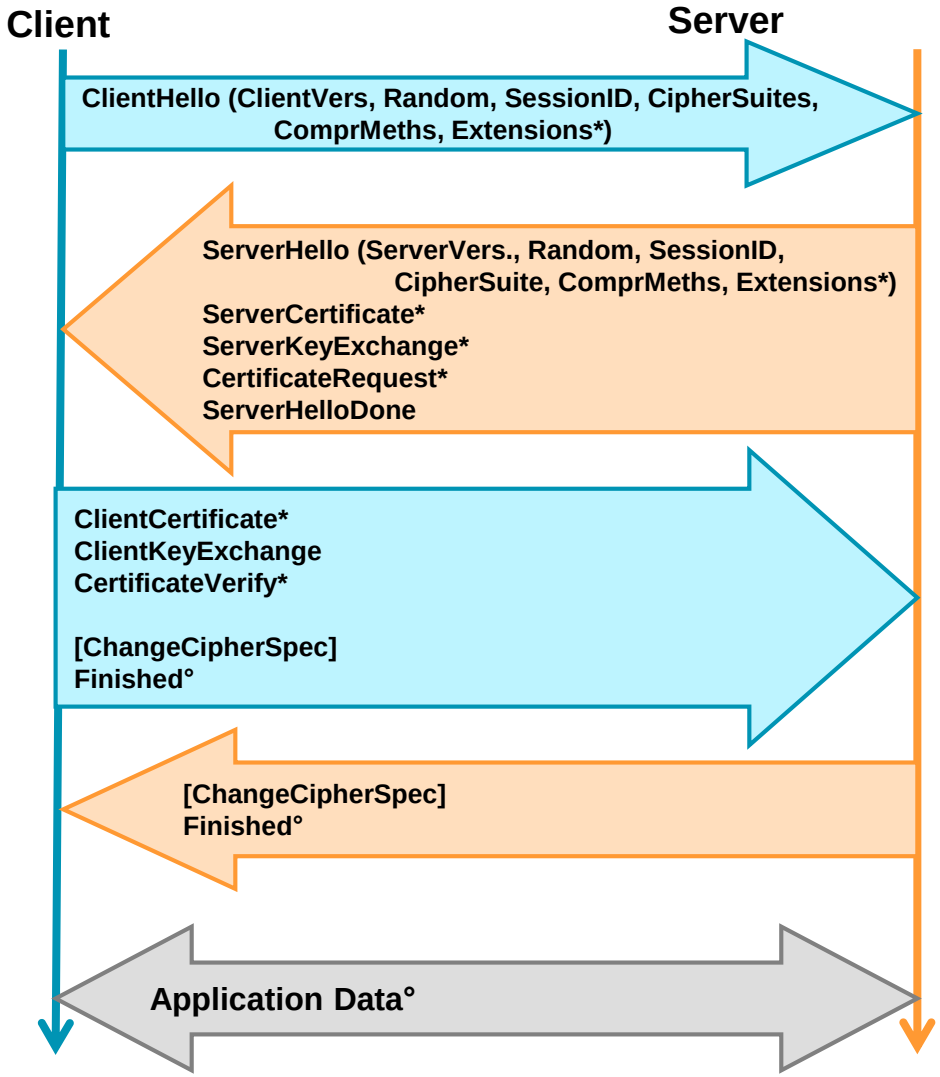
Handshake mit „RSA encrypted premaster secret"-Protokoll



Handshake mit Diffie-Hellman-Key-Exchange



Ablauf des TLS1.2-Handshake-Protokolls (im Detail)



** = optional ° = verschlüsselt*
[ ] = Sonderprotokollelement

- **ServerCertificate***: wird für die üblicherweise eingesetzte Serverauthentifizierung gesendet
- **ServerKeyExchange***: Wird nur gesendet, wenn das ServerCertificate nicht genug Info enthält, um Premastersecret zu erzeugen. Dies ist der Fall bei ephemeralem oder anonymem (zertifikatslosem) Diffie-Hellman
- **CertificateRequest*:** wird gesendet, falls Server eine Clientzertifizierung fordert
- **CertificateVerify***: Signierter Hash über die bisherigen Nachrichten. Nachweis, dass der Client auch den dazugehörigen privaten Schlüssel besitzt.
- **ChangeCipherSpec:** Ein Byte, das anzeigt, dass alle weiteren Nachrichten verschlüsselt übertragen werden (° = verschlüsselt)
- **Finished°**: Signierter MAC über alle vorangegangenen Handshakenachrichten. Erste verschlüsselte Nachricht, um Verschlüsselung zu verifizieren. Muss direkt nach ChangeCipherSpec gesendet werden. Datenaustausch erst dann, wenn Finish des Kommunikationspartners überprüft.

Server Key Exchange Message (RFC 5246, Kap. 7.4.3)

- **Server Key Exchange Message wird nur gesendet, wenn öffentlicher Schlüssel im Zertifikat nicht ausreicht. Das ist der Fall für:**
  - **DHE_DSS:** Ephemeral DH. Die DH-Werte werden mit einem DSS-Schlüssel signiert. Überprüfung des signierten DH-Wertes mittel Zertifikat. (DSS = Digital Signature Standard)
  - **DHE_RSA:** Ephemeral DH. Die DH-Werte werden mit einem RSA-Schlüssel signiert. Überprüfung des signierten DH-Wertes mittel Zertifikat.
  - **DH_anon:** Diffie-Hellman-Wert, ohne Einsatz von Zertifikaten. Gefahr von Man-in-the-Middle-Angriffen
- **Server Key Exchange Message darf nicht gesendet werden, wenn DH-Wert aus dem öffentlicher Schlüssel im Zertifikat gewonnen werden kann. Das ist der Fall für:**
  - **RSA:** Client erzeugt Pre-Master-Secret. Öffentlicher Schlüssel aus Zertifikat wird zur Verschlüsselung des Pre-Master_Secrets verwendet.
  - **DH_DSS:** Statischer DH. Zur Signatur wird DSS verwendet.
  - **DH_RSA:** Statischer DH. Zur Signatur wird RSA verwendet.

Authentifizierung bei RSA-basiertem Verfahren

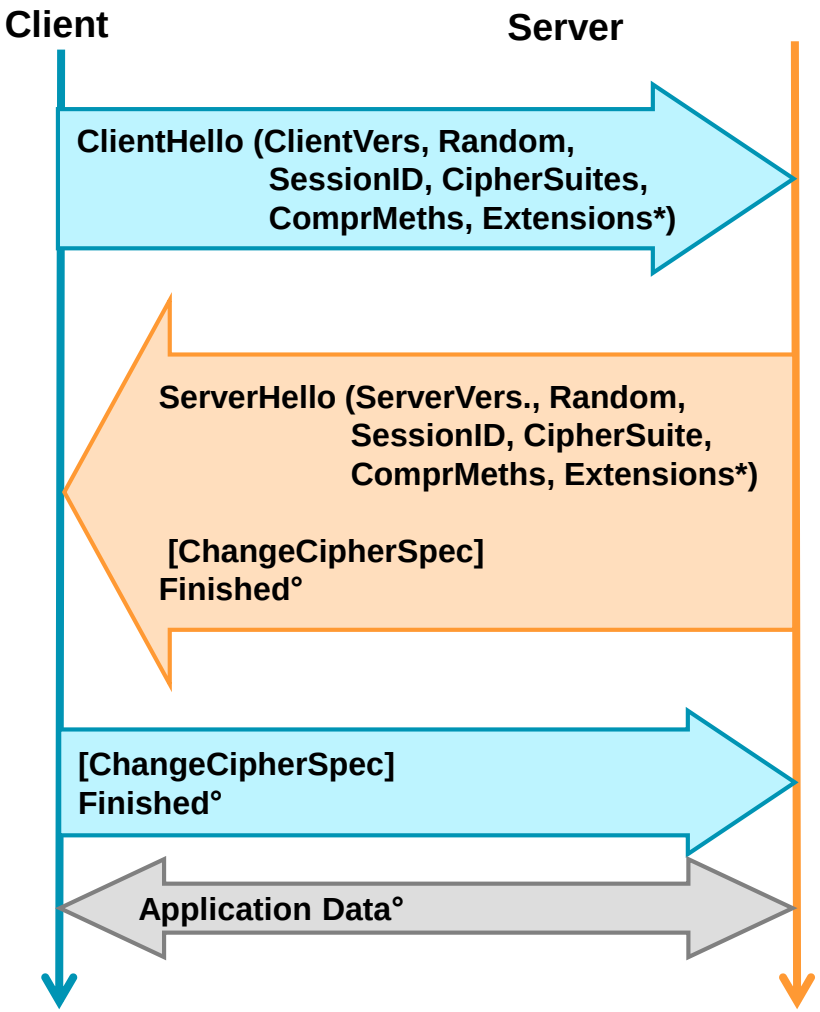
- **Serverauthentifizierung**
  - Der Client prüft das Serverzertifikat.
  - Das reicht, denn der Client weiß, dass nur der Server das verschlüsselte Pre-Master-Secret entschlüsseln kann.
- **Clientauthentifizierung**
  - Im Fall von CertificateRequest durch Server, muss der Client sein Zertifikat zusammen einem signierten Hashwert über das Master-Secret und alle Handshake Nachrichten senden.
  - Clientauthentifizierung findet im Standardfall nicht statt *(Warum?)*
- **Finished-Nachricht von Client und Server**

  In der RSA-Finished-Nachricht sind gehashed und verschlüsselt enthalten:
  - Master-Secret
  - Alle Handshake Nachrichten

Authentifizierung bei DH-basiertem Verfahren

- **Serverauthentifizierung**
  - **Fall 1: statisches Diffie-Hellman-Verfahren:**
    - **Der Server sendet dem Client sein Zertifikat, das den DH-Wert als öffentlichen Schlüssel enthält**
    - **Der Client überprüft das Server-Zertifkat**
  - **Fall 2: ephimerales Diffie-Hellman-Verfahren**
    - **Der Server sendet dem Client einen ephemeralen DH-Wert der mit dem vereinbarten Signaturalgorithmus signiert ist. Um Replay-Attacken zu verhindern, wird der DH-Wert mit den Zufallswerte aus der Hello-Nachricht gesalzen.**
    - **Der Client überprüft die Signatur der DH-Werts**
- **Clientauthentifizierung**
  - **Fall 1: statisches Diffie-Hellman-Verfahren: analog Serverauthentifizierung**
  - **Fall 2: ephimerales Diffie-Hellman-Verfahren: analog Serverauthentifizierung**
- **Beim statischen Diffie-Hellman-Verfahren wird stets das gleiche Pre-Master-Secret erzeugt.**
  Das Master-Secret wird jedoch stets verschieden sein, denn die Zufallswerte aus den Hello-Nachrichten gehen in das Master-Secret ein.

Wiederaufnahme einer TLS-Session (TLS Session Resumption)



- Clientseitige Wiederaufnahme der Verbindung durch Bezugnahme auf bestehende Sitzungs-ID
- Server-Annahme durch ServerHello mit Sitzungs-ID
- CipherSuites und Kompressions-algorithmen müssen mit den bereits ausgehandelten Algorithmen kompatibel sein.
- Verwendet bereits bestehendes Master_Secret

Das neue Protokoll TLS 1.3 (draft-ietf-tls-tls13-10 / Oct 19, 2015)

Motivation/Ziele:

- ***Unnötige Optionen*** **werden entfernt .**
  (Kompression, veraltete Verschlüsslungsverfahren, …)
- **RSA Zertifikate sind noch erlaubt, aber nicht mehr das RSA-Schlüsselaustauschverfahren.**
- **DES Verschlüsselungsverfahren wird nicht mehr erlaubt. Durch Optimierung der erlaubten Ciphersuites wird höhere Sicherheit erreicht.**
- **Perfect Forward Secrecy wird forciert.**
- **Beobachtung: Viele Sicherheitslücken entstehen durch fehlerhaft Implementierungen. Daher Vereinfachung von TLS, führt zur einfacheren Implementierbarkeit von TLS.**
- **Unterstützung im Einsatz durch "Common Guidelines" und "Best Practices for using TLS".**
- **'1 round trip time' Handshake für optimale Schnelligkeit**
  (reduzierte Paketanzahl)
- **Migration wird durch backwards compatibility mode sichergestellt.**

Implementierungen: Von TLS benutzte TCP-Ports

Identifier	Ports	Zweck
https	443/tcp	TLS-HTTP-Protokoll (RFC2818, 05/00)
smtps	465/tcp	SMTP über TLS (inoffiziell)
nntps	563/tcp	TLS-Network News Transfer Protocol
smtps	587/tcp	SMTP mit Service Extension for TLS (RFC6409, 11/11)
ldaps	636/tcp	TLS-Lightweight Directory Access Protocol
ftps	989/tcp 990/tcp	FTP über TLS (RFC4217, 10/05)
	992/tcp	Telnet über TLS
imaps	993/tcp	TLS-IMAP4
pop3s	995/tcp	TLS-POP3

Sicherheit in Netzen
Modul 6: TLS – Transport Layer Security

1. TLS-Einordnung (Geschichte, OSI-Referenzmodell)
2. TLS-Datenübertragung
3. TLS Schlüssel und Algorithmen
4. TLS-Handshake
5. TLS-Implementierung (PDUs, SSL-Sockets)
6. **Ausgewählte Angriffe gegen TLS**

Downgrade-Angriffe auf TLS

Vorgehensweise

- Angreifer schaltet sich als Man-in-the-Middle in die Kommunikation ein.
- Angreifer löscht starke Cipher-Suites
- Folge davon: Client und Server einigen sich auf ein schwaches Verfahren

Lösung

Signierter Hashwert über alle ausgetauschten Nachrichten in der Finished-Nachricht

Hierdurch kann können Sender und Empfänger überprüfen, ob TLS-Schlüsselaustausch unverändert beim gegenüber angekommen ist.

Poodle-Schwachstelle

- Veröffentlich: 14. Oktober 2014
- Voraussetzung alte Protokollversion SSL 3.0 auf Server und Client werden noch unterstützt
- Test des Browsers durch *https://www.poodletest.com/*
  - Firefox: 11/2014 (vul.), 11/2015 (o.k.)
  - IE: 11/2014 (vul.), 11/2015 (o.k.)
  - Chrome: 11/2014 (o.k.), 11/2015 (o.k.)
- Voraussetzung für den Angriffs ist:
  - JavaScript-Code in den Browser des Opfers einschleusen und
  - den verschlüsselten Netzwerkverkehr überwachen und manipulieren können (Man-in-the-Middle)
- Dann Downgrade auf SSL 3.0 forcieren
- RC4 Verschlüsselung + CBC haben erwiesene Schwachstellen

- Genaue Beschreibung des Angriffs:
  https://www.openssl.org/~bodo/ssl-poodle.pdf
  https://www.imperialviolet.org/2014/10/14/poodle.html