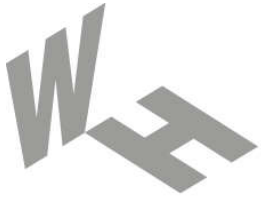


Westfälische Hochschule
Gelsenkirchen Bocholt Recklinghausen
University of Applied Sciences

Secure Socket Layer (SSL)
Transport Layer Security (TLS)

Prof. Dr. (TU NN)
Norbert Pohlmann

Institut für Internet-Sicherheit – if(is)
Westfälische Hochschule, Gelsenkirchen
http://www.internet-sicherheit.de

Inhalt

- Einleitung: Definitionen und Ziele
- Protokollaufbau: Header und Nachrichtentypen
- Protokollablauf
- Bedeutung von Zertifikaten in Browsern
- Authentikationsmethoden
- Schwachstellen und Angriffsmöglichkeiten
- SSL/TLS in der Praxis
- Zusammenfassung

Inhalt

- **Einleitung: Definitionen und Ziele**
- Protokollaufbau: Header und Nachrichtentypen
- Protokollablauf
- Bedeutung von Zertifikaten in Browsern
- Authentikationsmethoden
- Schwachstellen und Angriffsmöglichkeiten
- SSL/TLS in der Praxis
- Zusammenfassung

Einleitung
→ Hauptaufgaben von TLS/SSL

- Da das Internet offen ist und die gesetzlichen Rahmenbedingungen weltweit sehr unterschiedlich sind (Asien, AUS, Europa, usw.), ist die Nutzung der Verschlüsselung für die Kommunikation zwischen Client und Server von besonderer Sicherheitsbedeutung.

- Sehr viele Sicherheitsaspekte im Web, wie Eingabe von Passworten und Kreditkarten-Informationen haben mit der Einrichtung einer vertrauenswürdigen Verbindung zwischen Client und Server zu tun.

- Der vorherrschende Ansatz für die Verschlüsselung im Web ist die Verwendung von SSL(Secure Socket Layer)/TLS (Transport Layer Security).

Einleitung
→ Hauptaufgaben von TLS/SSL

- **1).**
  Die **Authentikation der Kommunikationspartner** unter Verwendung von **asymmetrischen Verschlüsselungsverfahren und Zertifikaten**.

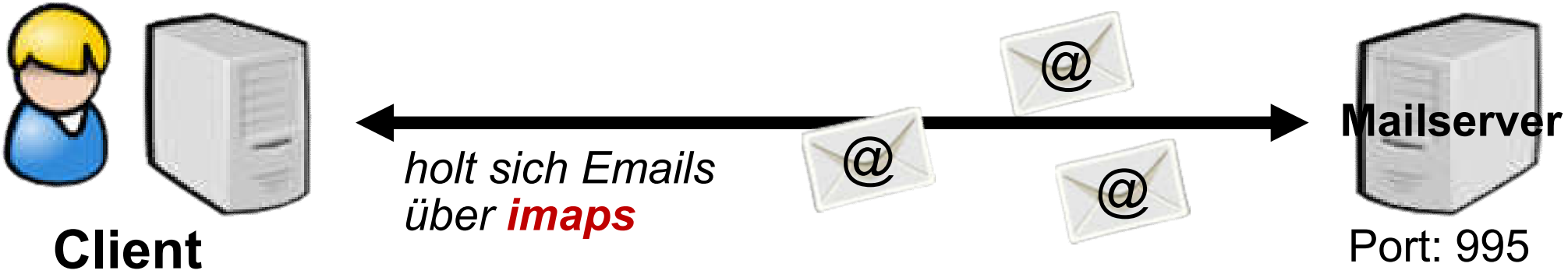
- **2).**
  Die **vertrauliche Ende-zu-Ende-Datenübertragung** mit Hilfe **symmetrischer Verschlüsselungsverfahren** unter der Nutzung eines gemeinsamen Sitzungsschlüssels.

- **3).**
  Die **Sicherstellung der Integrität der transportierten Daten** unter Verwendung von **Message Authentication Codes.**

Einleitung
→ Anwendungsbeispiele

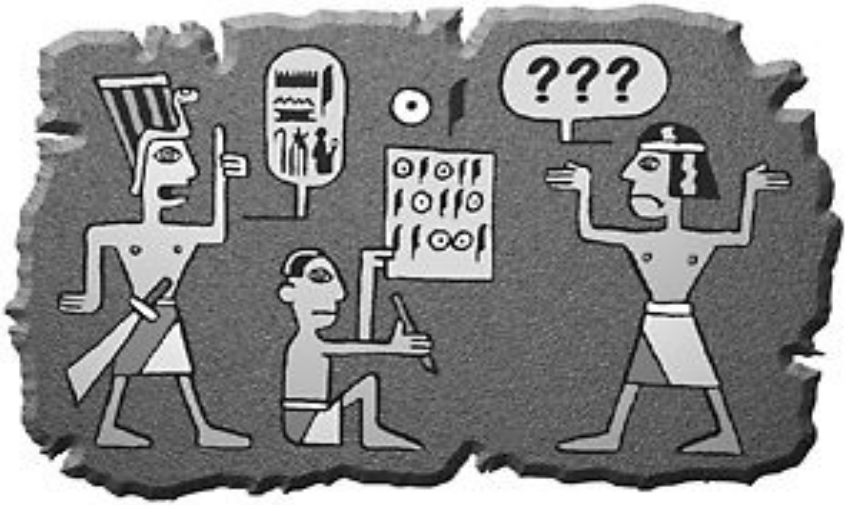
- Browser kommuniziert mit Webserver über sichere Verbindung

- Client holt sich E-Mails vom Mailserver über gesicherte Verbindung



Einleitung
→ Analogie

Nachricht verschlüsseln



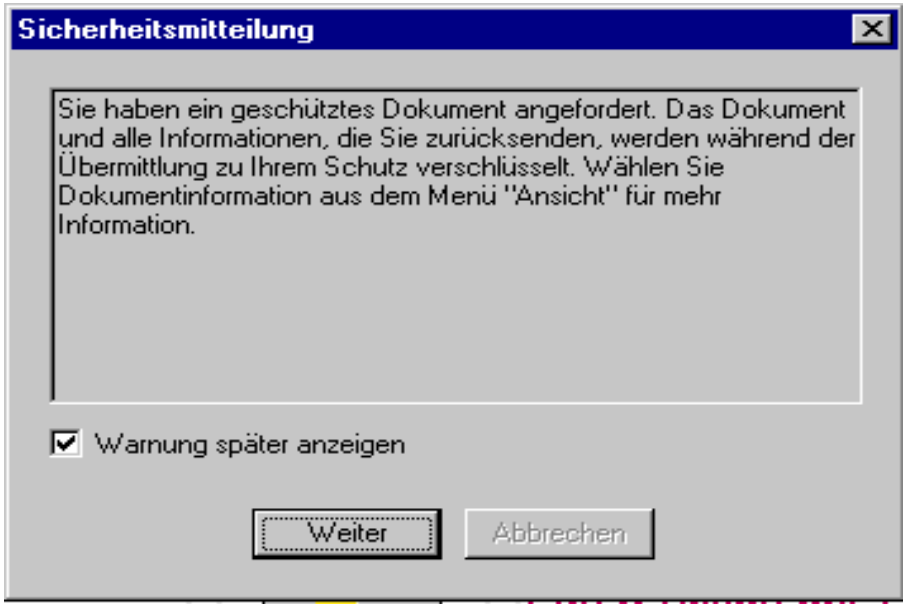
transportieren

mit Siegel versehen

sicher unterbringen

Einleitung
→ Browserhinweise

- 1. Benutzerhinweis



- 2. „Ungeöffnetes Schloß“ in der Statuszeile / Titelleiste des Browsers

Einleitung
→ Geschichte

- **Nov. 1993:** 1. Release von Mosaic (1. verbreiteter Webbrowser)
- **1994:** Nur neun Monate später:
  Netscape veröffentlicht die erste Version von **SSL** (1.0)
- Fünf Monate später: Release **SSL 2.0** veröffentlicht
- **Januar 1999:** SSL wird von der IETF im RFC 2246 als Standard festgelegt und umbenannt zu **Transport Layer Security** (TLS).
  Die Unterschiede zwischen SSL 3.0 und TLS 1.0 sind sehr klein.
- Doch dadurch entstanden Versions-Verwirrungen.
- So meldet sich TLS 1.0 im Header als Version SSL 3.1.
- Später wurde TLS durch weitere RFCs erweitert:
  RFC 2712, RFC 2817, RFC 2818, RFC 3268
- **April 2006:** In RFC 4346 wird die Version 1.1 von TLS standardisiert und damit RFC 2246 obsolet.
- **April 2008:** RFC 5246 (Version 1.2 von TLS) dies macht 2246 obsolet

Einleitung
→ TLS als Weiterentwicklung von SSL (1/3)

- **TLS (Transport Layer Security)** stellt die standardisierte Weiterentwicklung von SSL 3.0 dar.
- Die Änderungen zu SSL sind zwar nicht dramatisch, jedoch so groß, dass beide Protokolle nicht zu einander kompatibel sind, obwohl TLS einen Abwärtskompatibilitätsmodus zu SSL 3.0 beinhaltet.
- Folgende Neuerungen bietet TLS gegenüber SSL:
  - TLS unterstützt mehr Fehlermeldungen
  - neue kryptographische Operationen z.B. für Schlüsselberechnung
  - proprietäre Verschlüsselung Fortezza wurden entfernt
  - Festlegung einer Verschlüsselung, die jede Implementation enthalten muss: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA

Einleitung
→ TLS als Weiterentwicklung von SSL (2/3)

- **TLS 1.2:**
  - MD5 / SHA-1 Kombination zur Generierung von Pseudozufallszahlen abgelöst durch Ciphersuite-Spezifische Algorithmen. Neuer Standard nun P_SHA256
  - MD5/SHA-1 bei der digitalen Signatur durch einen einfachen Hashwert ersetzt. Signierte Elemente enthalten jetzt ein Feld, in dem der verwendete Hashalgorithmus angegeben wird
  - Signalisierung von Unterstützungen von Hash und Signaturfunktionen verbessert
  - Neue Erweiterungen und Cipher Suites wurden integriert. Standard-Ciphersuite ist nun TLS_RSA_WITH_AES_128_CBC_SHA. Alle IDEA und DES basierten Ciphersuites wurden aus dem Dokument entfernt.
  - OpenSSL als quasi Standard Implementation von TLS unter Unix ermöglicht noch kein TLS 1.2.

Einleitung
→ TLS als Weiterentwicklung von SSL (3/3)

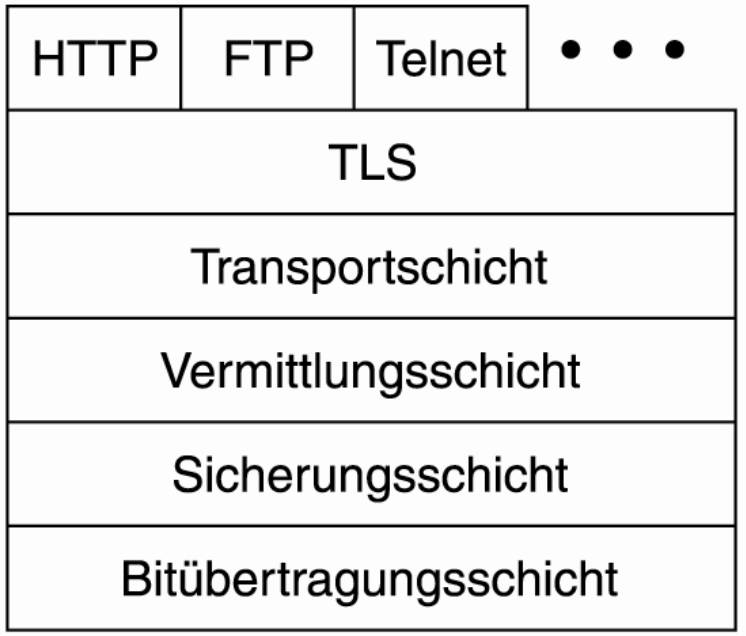
- TLS 1.0 wird von allen gängigen Browsern unterstützt
- Bei TLS 1.1 und 1.2 sehr unübersichtlich

Einleitung
→ Terminologie

- SSL = TLS vorallem auch in der Literatur!
- Bezeichnung wenn ein Protokoll SSL/TLS nutzt:
  - HTTP over SSL/TLS
  - HTTPS

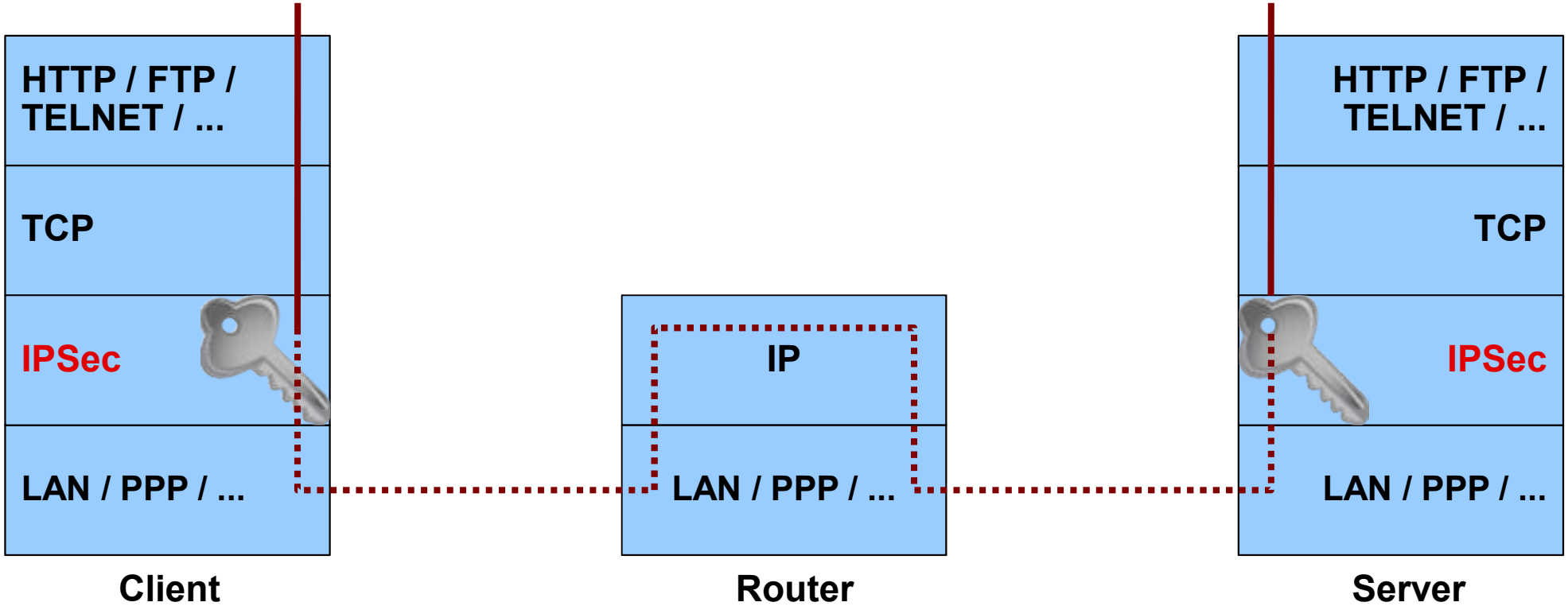
Einleitung
→ Schichten

- TLS kann eine Vielzahl höherer Protokolle unterstützen (HTTP, FTP, SMTP, POP3, Telnet, ...).
- TLS ist in **zwei Schichten** angeordnet.
- Den Kern des Protokolls bildet eine TLS-Datensatz-Protokollschicht, die einen sicheren Kanal zwischen Client und einem Server implementiert.

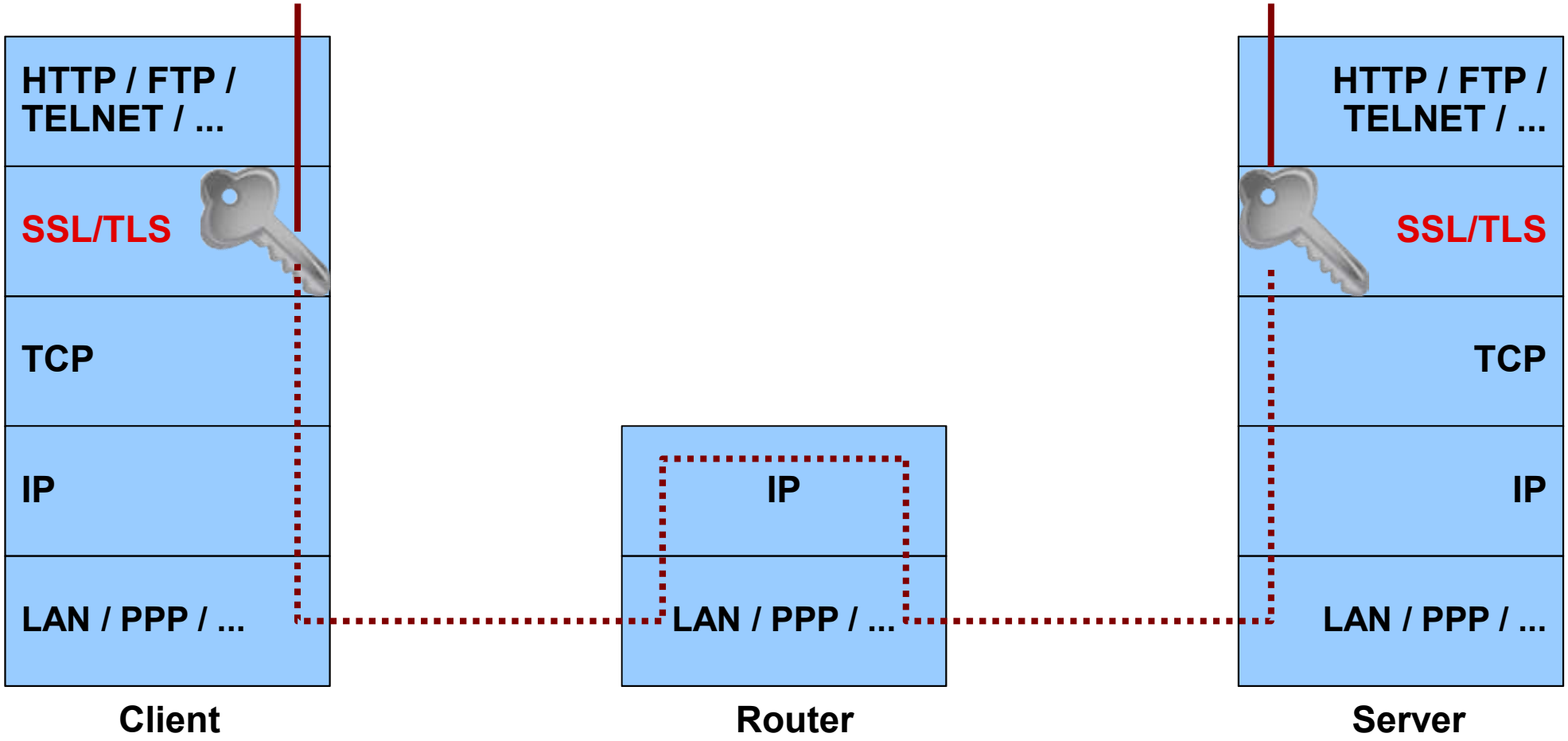


- Die genauen Eigenschaften des Kanals werden bei der Einrichtung festgelegt, können aber Nachrichtenfragmentierung und Komprimierung beinhalten, die in Kombination mit Authentikation, Integrität und Vertraulichkeit angewendet werden.

Einleitung
→ IPSec vs TLS/SSL (1/2)



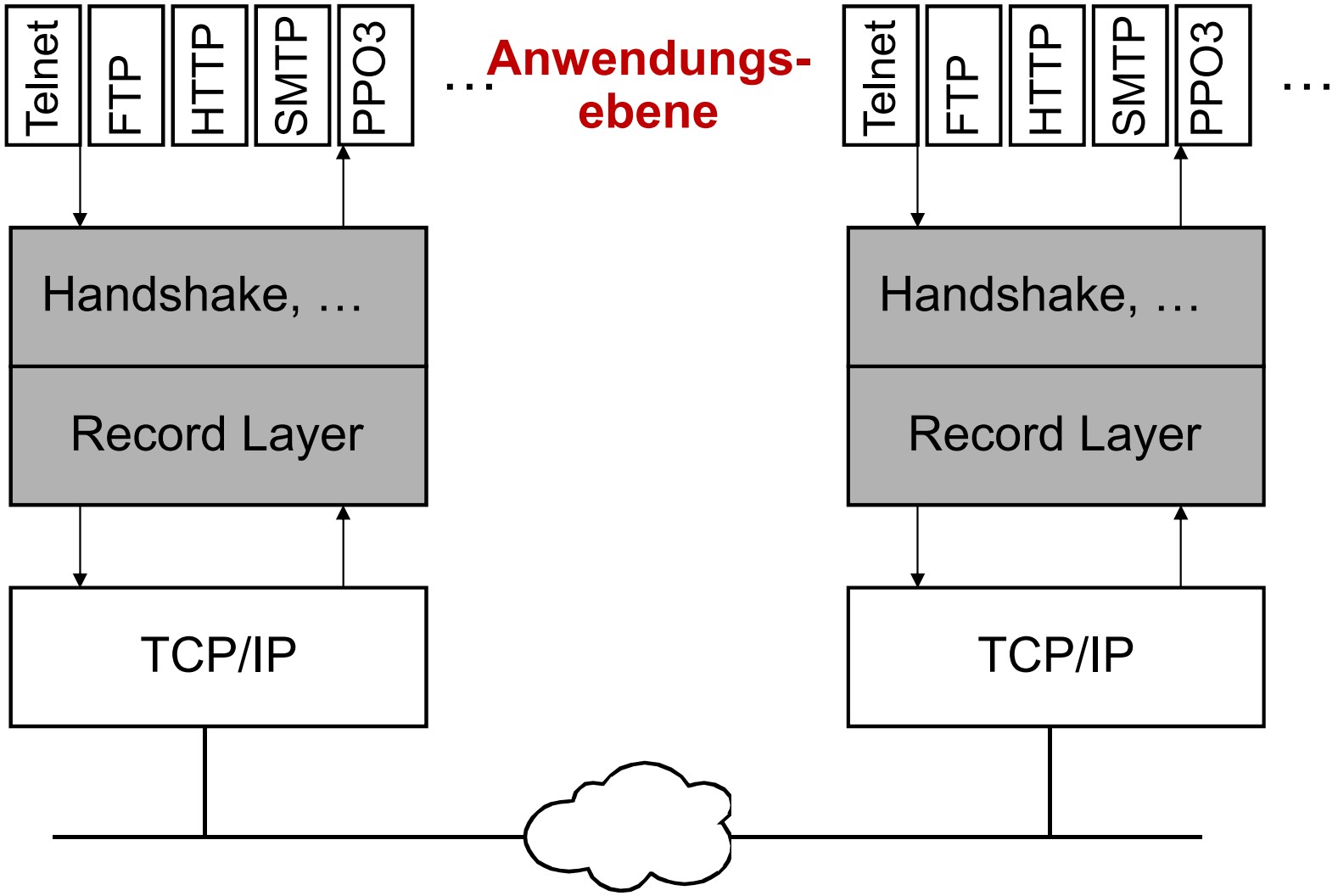
Einleitung
→ IPSec vs TLS/SSL (2/2)



Inhalt

- Einleitung: Definitionen und Ziele
- **Protokollaufbau: Header und Nachrichtentypen**
- Protokollablauf
- Bedeutung von Zertifikaten in Browsern
- Authentikationsmethoden
- Schwachstellen und Angriffsmöglichkeiten
- SSL/TLS in der Praxis
- Zusammenfassung

SSL/TLS-Protokoll
→ Schichteneinordnung (1/2)



SSL/TLS-Protokoll
→ Schichteneinordnung (2/2)

- SSL/TLS übernimmt die Aufgaben der Sitzungs- und Präsentationsschicht (Schichten 5 und 6) des ISO/OSI-Modells.
- Ein wesentlicher Vorteil der Sitzungsschicht gegenüber der TCP-Ebene besteht darin, dass Zustandsinformationen über einen längeren Zeitraum und über verschiedene Einzelverbindungen hinweg gespeichert und für die Verwaltung genutzt werden können.
- Für das zustandslose HTTP-Protokoll, das für jeden Zugriff auf eine Webseite eine neue TCP-Verbindung aufbaut, bedeutet das, dass mehrere solcher Verbindungen zu einer Sitzung gebündelt und damit effizienter als die jeweiligen Einzelverbindungen verwaltet werden können.
- Protokoll ist erweiterbar und flexibel, um Zukunftssicherheit vor allem bei den Verschlüsselungsalgorithmen zu gewährleisten
- **SSL/TLS arbeitet transparent, so dass es leicht eingesetzt werden kann, um Anwendungsprotokolle/-dienste ohne eigene Sicherheitsmechanismen vertrauenswürdige Verbindungen zur Verfügung zu stellen.**

SSL/TLS
→ Header Aufbau (1/2)

- SSL/TLS besteht aus **2 Schichten:**
  - 1. Schicht: Record Layer-Protokoll
  - 2. Schicht: vier SSL/TLS-Teilprotokollen (CCS, Alert, Handshake, Application)

- Besondere Bedeutung haben dabei die Nachrichtentypen des Handshake-Protokolls, welche u.a. zum Verbindungsaufbau dienen.

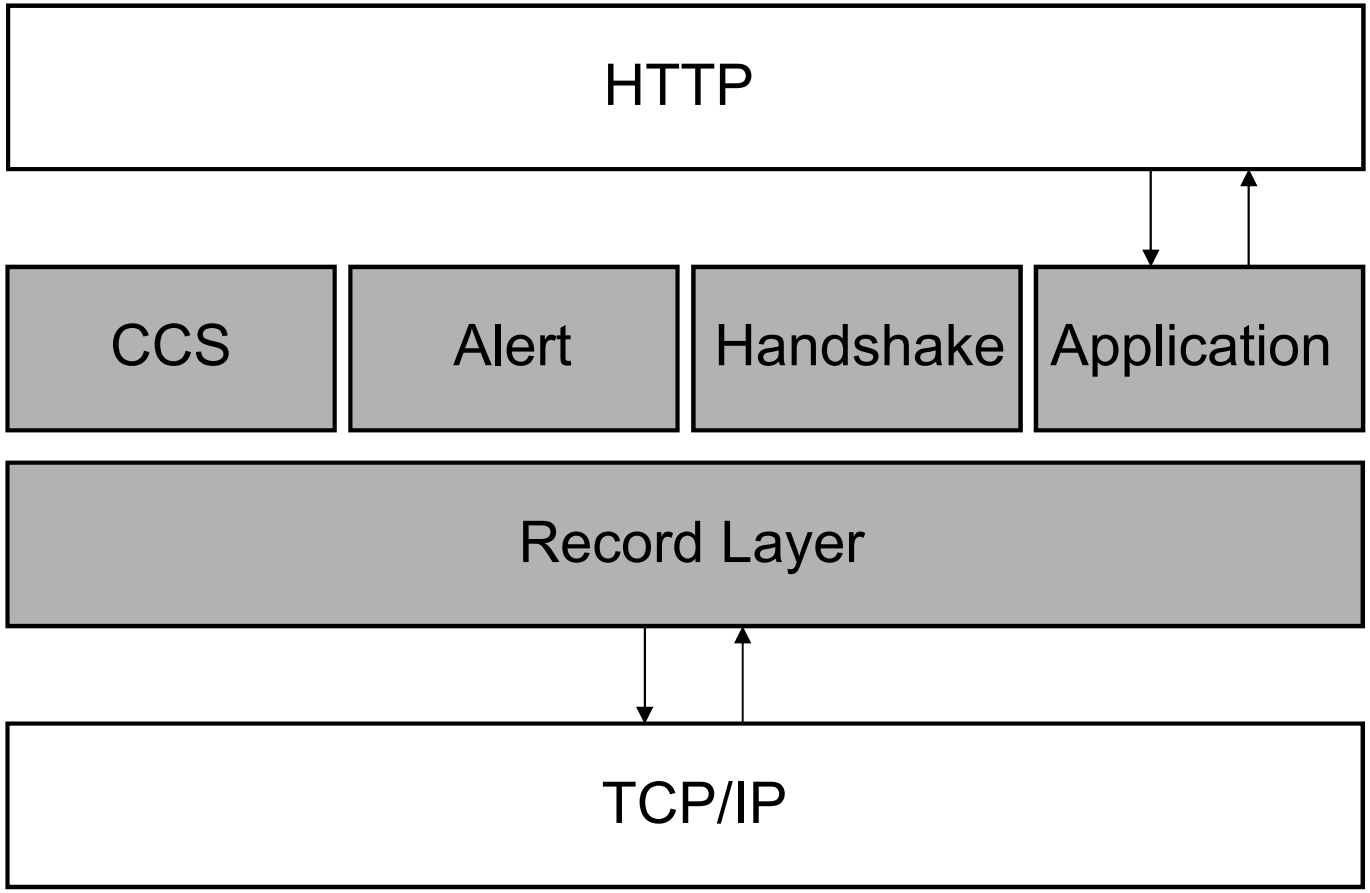
- Im selben Rechnersystem können Prozesse auch an SSL/TLS vorbei direkt auf TCP zugreifen.

- Die Entscheidung darüber fällen die Applikationen: z.B.

  – Browser erkennen die SSL/TLS-Erfordernis an der URL: https://... -> erzwingt SSL/TLS

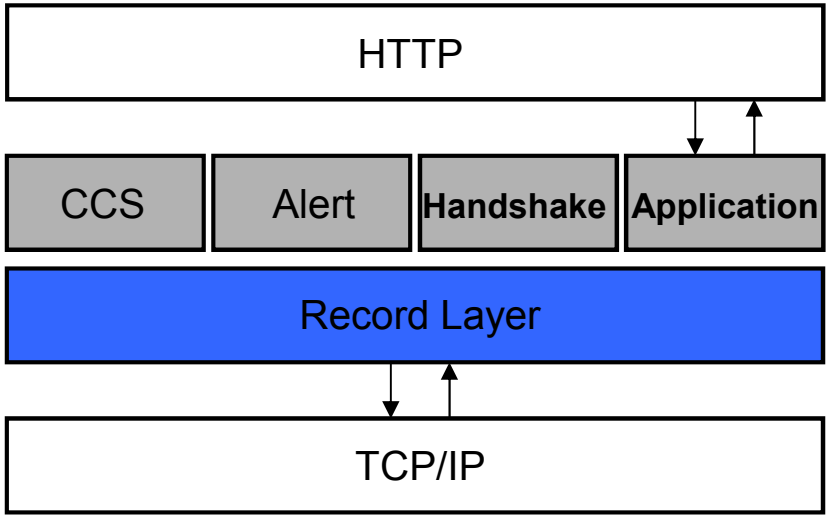
SSL/TLS
→ Header Aufbau (2/2)

- Aufbau der SSL/TLS-Schichten am Beispiel von https:



SSL/TLS – Record Layer Protokoll
→ Aufgaben

- **Das Record Layer Protokoll leitet die Nachrichten der anderen Protokolle verschlüsselt an TCP weiter**.
- Bietet zwei verschiedene Dienste die zusammen oder einzeln genutzt werden können:
  - Ende-zu-Ende Verschlüsselung
  - Sicherung der Nachrichten-Integrität und Authentizität
- Zu den Aufgaben des Record-Protokolls gehören:
  - die Fragmentierung der Daten der Anwendungsebene,
  - deren Kompression,
  - Berechnung von MACs und
  - Verschlüsselung der TLS-Records.



SSL/TLS – Record Layer Protokoll
→ Aufbau

- Gesamtgröße: 5 Byte

0	1	2	3	5
Type	Version Major	Version Minor	Length	

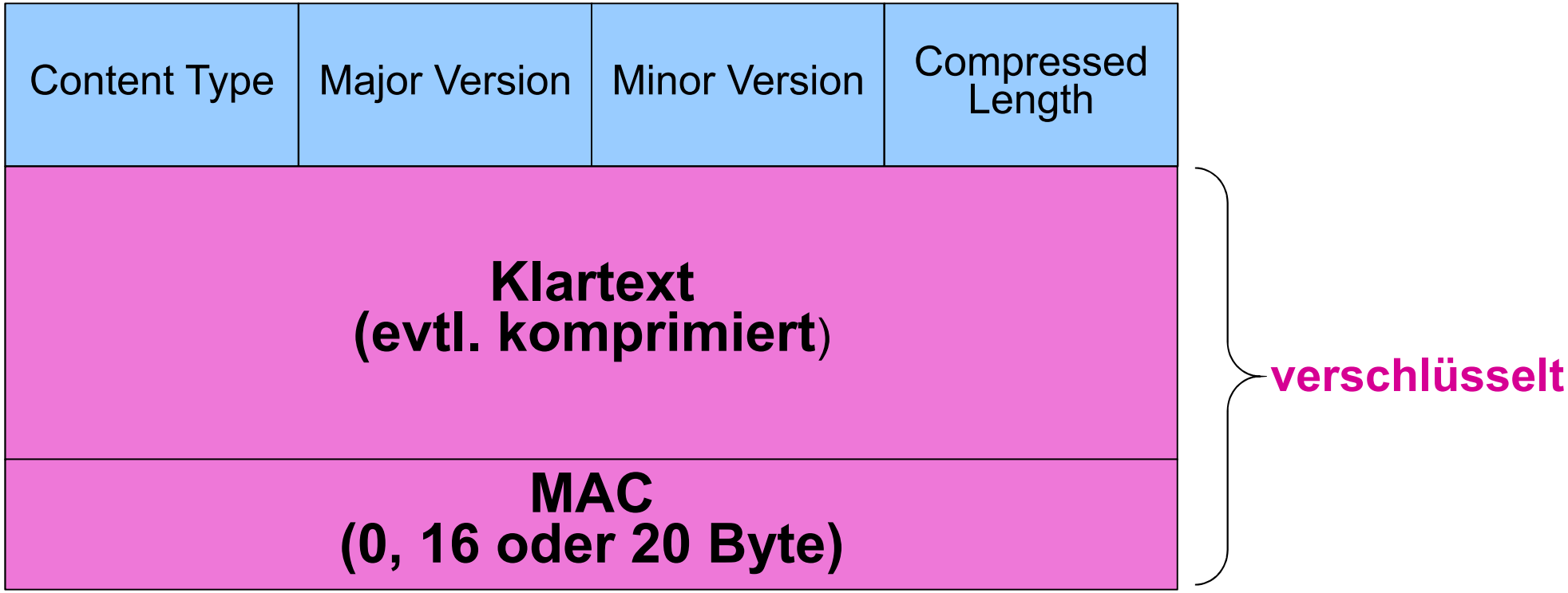
- Type (1 Byte) : Nummer des “höheren” SSL/TLS-Protokolls
  - Change Cipher Spec: 20
  - Alert: 21
  - Handshake: 22
  - Application Data: 23
- Version Major (1 Byte) : Hauptnummer der Version, SSLv3: 3, TLS: 3
- Version Minor (1 Byte) : Unterversion, SSLv3: 0, TLS: 1,2
- Length (2 Byte) : Länge der Nutzdaten in Byte.
  Maximalwert darf nicht grösser sein als 2^{14} + 2.048 (16.384 + 2.048 Byte).

SSL/TLS – Record Layer Protokoll
→ Ablauf

- Die Record-Schicht hat die Aufgabe, Datenpakete in SSL/TLS-Records mit einer maximalen Größe von 2^{14} Byte zu fragmentieren.
- Optional werden die Daten dann komprimiert. (Default nein)
  - Komprimierungsverfahren wird beim Handshake ausgehandelt
- Diese Daten + Sequenznummer werden dann mit einer MAC-Funktion gehasht.
- Die Daten und der MAC werden dann verschlüsselt.
- Die Kompression muss verlustfrei sein und darf die Länge des Fragments nicht mehr als 1024 Byte vergrößern.
  - Bei kleinen Blöcken kann es vorkommen, dass durch die Kompression der Block vergrößert anstatt verkleinert wird.

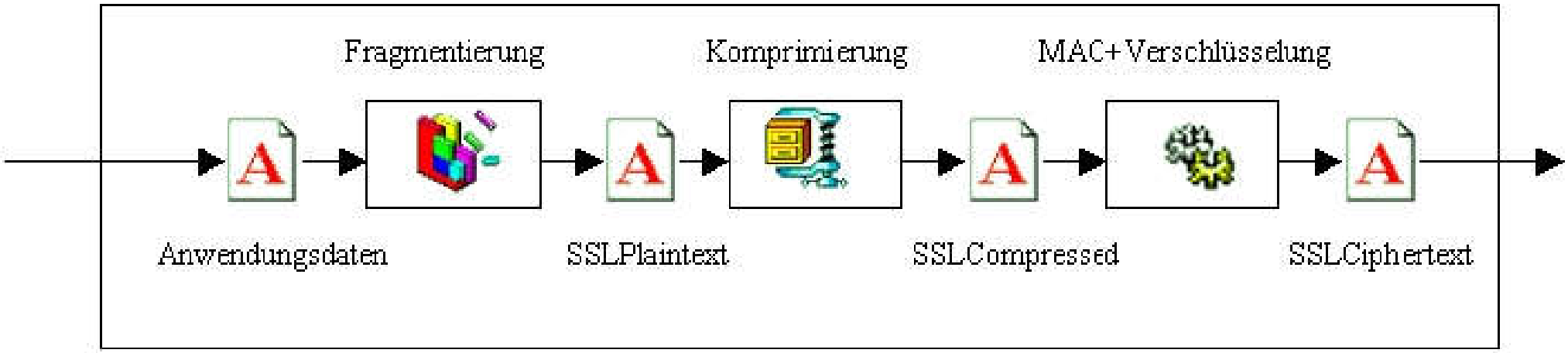
SSL/TLS – Record Layer Protocol
→ Verschlüsselte Daten

- Die eigentlichen Nutzdaten der Protokolle werden bei SSL/TLS verschlüsselt über das Application Data Protokoll übertragen.
- Der komplette Record-Layer-Header ist ungeschützt und somit lesbar.
- Zudem kann beim Handshake-Protokoll die Art der Handshake-Nachricht ausgelesen werden.

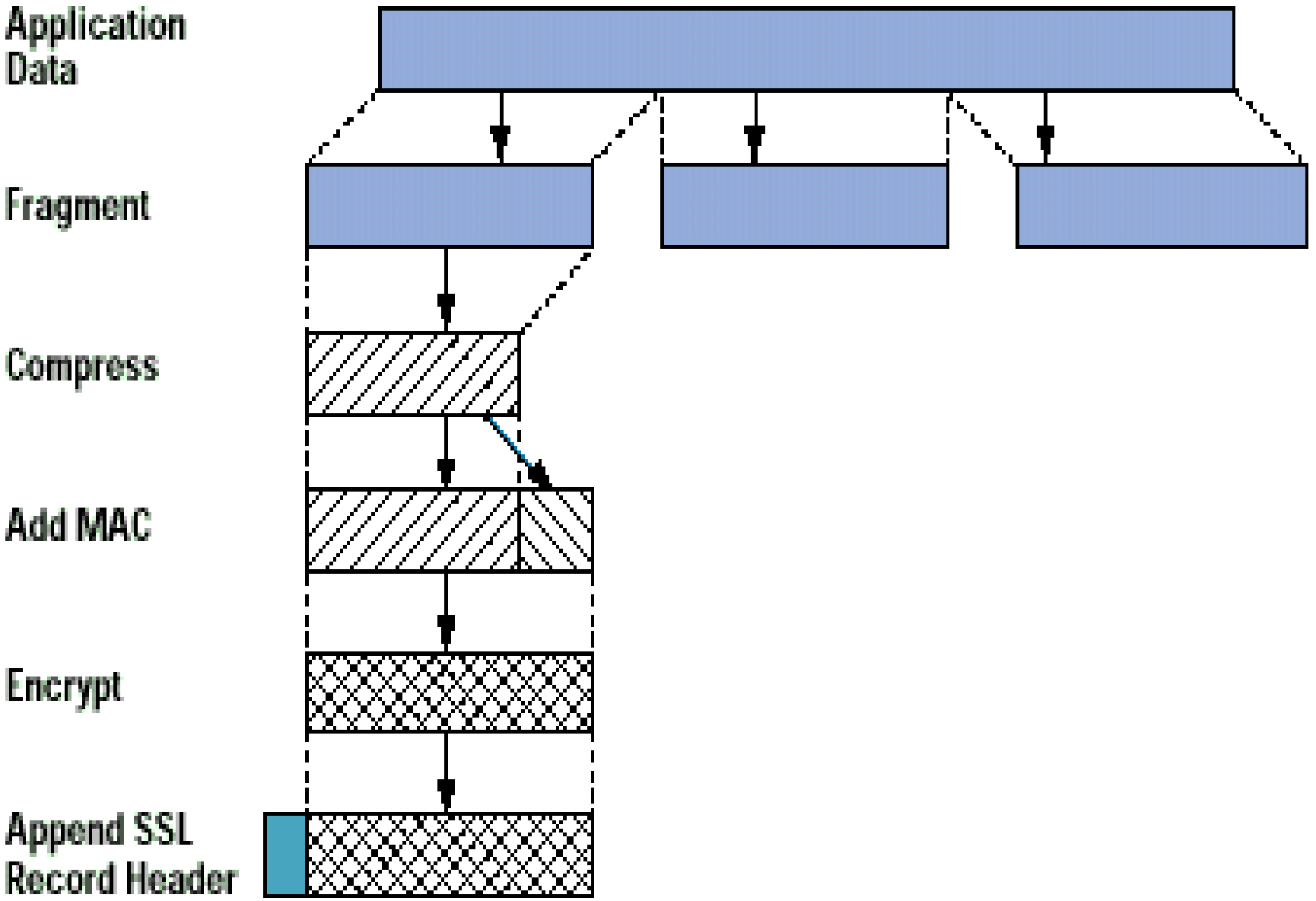


SSL/TLS – Record Layer Protocol
→ Verfahren des SSL/TLS-Record-Protokolls

1. Anwendungsdaten fragmentieren
2. Fragmente komprimieren
3. Berechnen und Hinzufügen des MAC
4. Verschlüsseln der neuen Pakete
5. Voranhängen des SSL/TLS-Headers



SSL/TLS – Record Layer Protocol
→ Übersicht



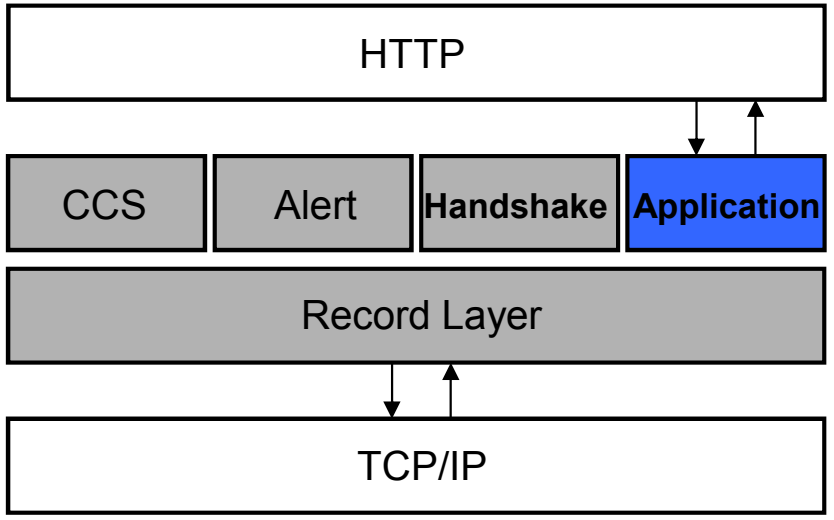
SSL/TLS – Record Layer Protocol
→ MAC Berechnung

- Als MAC-Verfahren wird das HMAC-Verfahren verwendet (siehe Kryptographie-Vorlesung).

MAC = hash (MAC_key | pad_2 |
hash (MAC_key | pad_1 | seq_num | Compressed.type |
Compressed.length | Compressed.fragment))

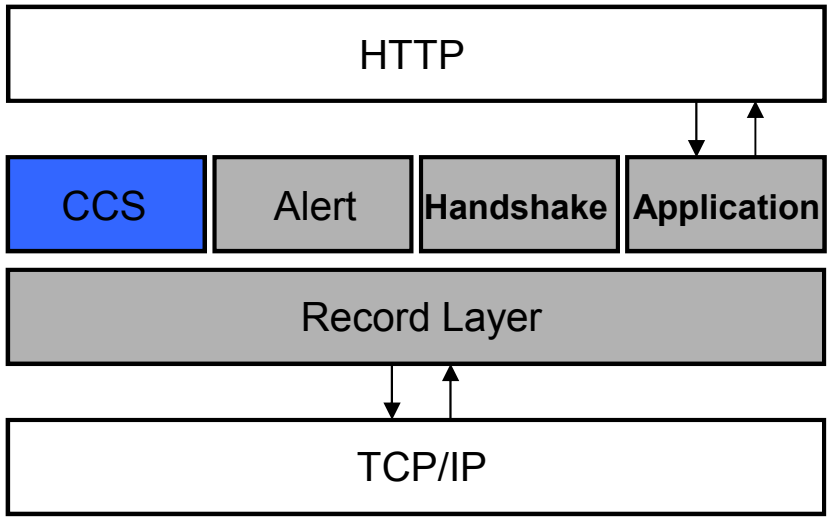
SSL/TLS – Application Data Protokoll
→ Aufgaben

- Das **Application Data Protokoll** ist definiert, um die Anwendungsdaten transparent, d.h. ohne Betrachtung des Inhalts, durchreichen zu können.
- Die anderen Protokolle oberhalb des Record Protokolls werden auf ihren Inhalt hin untersucht, dies ist bei den Daten des Application Data Protokoll nicht der Fall.
- Die Daten werden entsprechend der Sicherheitsparameter (aus dem Handshake Protokoll) fragmentiert, komprimiert, gegen Verfälschung geschützt und verschlüsselt.
- Gesamtgröße: Variabel



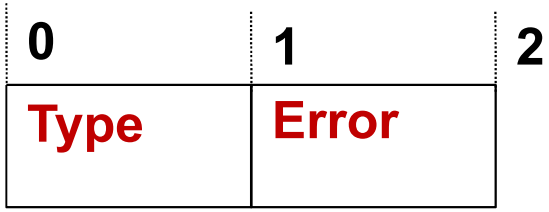
SSL/TLS – ChangeCipherSpec (CCS) Protokoll
→ Aufgaben

- Das **ChangeCipherSpec** Protokoll wird bei **Änderung des kryptografischen Algorithmus bzw. Parametern** genutzt .
- Es enthält nur eine Meldung bestehend aus einem Byte mit dem Wert 1.
- CCS bewirkt, dass der Empfänger die während des Handshakes ausgehandelten Parameter für die aktive Sitzung übernimmt.
- Wird eine vorhandene Sitzung reaktiviert, erfolgt die Meldung ChangeCipherSpec nach der Meldung ServerHelloDone.



SSL/TLS – Alert Protokoll
→ Aufgaben

- Das Alert Protokoll dient der **Signalisierung von besonderen Zuständen bzw. Problemen** (Bsp. Fehler, Verbindungsabbruch).
- Protokollnachricht enthält nur 2 Felder: Sicherheitslevel und Fehlercode.
- Gesamtgröße: 2 Byte



- Das erste Byte kann den Wert **warning**(1) oder **fatal**(2) haben und gibt den Grad der Fehlermeldung an.
- Wird der Wert “fatal” übertragen, beendet SSL/TLS sofort die Verbindung.
- Andere Verbindungen aus der Session bleiben bestehen, aber es kann keine neue Verbindung erzeugt werden.
- Das zweite Byte beschreibt den Fehler genauer.

SSL/TLS – Alert Protokoll
→ Fatal (1/2)

- **Fatale Alertmeldungen:**

Name	Inhalt
Bad_record_mac(20)	Falscher MAC für Record
Record_overflow(22)	Record erhalten mit einer Länge > 2^14 + 2048 Byte
Decompression_failure(30)	Dekomprimierungsfunktion erhält falschen Input
Handshake_failure(40)	Sender der Nachricht konnte die Sicherheitsparameter nicht akzeptieren
Illegal_parameter(47)	Feld im Handshake war inkonsistent mit anderen Feldern
unknown_ca(48)	CA Zertifikat konnte nicht gefunden werden oder ist nicht unter den vertrauenswürdigen CAs
access_denied(49)	Gültiges Zertifikat erhalten, aber nach der Anwendung der Zugangskontrolle wurde entschieden, nicht weiter zu machen
decode_error(50)	Länge der Nachricht falsch oder Nachricht konnte nicht decodiert werden, da ein Feld nicht korrekt ist

SSL/TLS – Alert Protokoll
→ Fatal (2/2)

- **Fatale Alertmeldungen:**

Name	Inhalt
decrypt_error(51)	Beim Handshake ist die kryptografische Operation fehlgeschlagen; Sender war nicht fähig die Signatur oder das Ende der Nachricht zu verifizieren
protocol_version(70)	Protokoll Version nicht unterstützt
Decompression_failure(30)	Dekomprimierungsfunktion erhält falschen Input
Handshake_failure(40)	Sender der Nachricht konnte die Sicherheitsparameter nicht akzeptieren
insufficient_security(71)	Server erwartet höhere Cipher Suite als der Client unterstützt
internal_error(80)	Interner Fehler unabhängig von Teilnehmern

SSL/TLS – Alert Protokoll
→ Warning

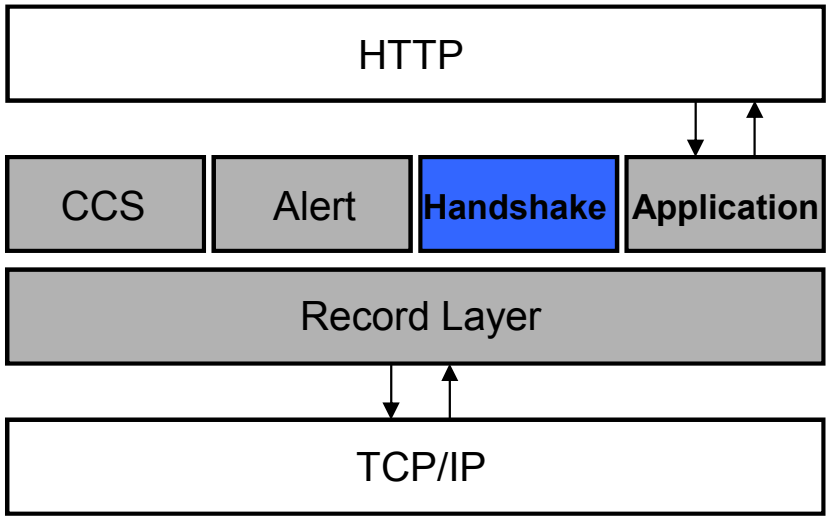
- **Warnungs Alertmeldungen:**

Name	Inhalt
decryption_failed_RESERVED(21)	Nicht mehr verwendet
no_certificate_RESERVED(41)	Nicht mehr verwendet
export_restriction_RESERVED(60)	Nicht mehr verwendet
bad_certificate(42)	Zertifikat beschädigt oder enthält Signatur, die nicht verifiziert werden konnte
unsupported_certificate(43)	Zertifikat eines nicht unterstützten Types
certificate_revoked(44)	Zertifikat wurde vom "Signer" annuliert
certificate_expired(45)	Zertifikat ist abgelaufen
certificate_unknown(46)	Sonstige Zertifikatsprobleme
user_canceled(90)	Handshake abgebrochen vom Benutzer
no_renegotiation(100)	Wenn nach dem client hello / hello die Parameter neu verhandelt werden sollen
unsupported_extension(110)	Versendet von Clients, die vom Server eine Erweiterung bekommen, die sie nicht verlangt haben

SSL/TLS – Handshake Protokoll
→ Aufgaben

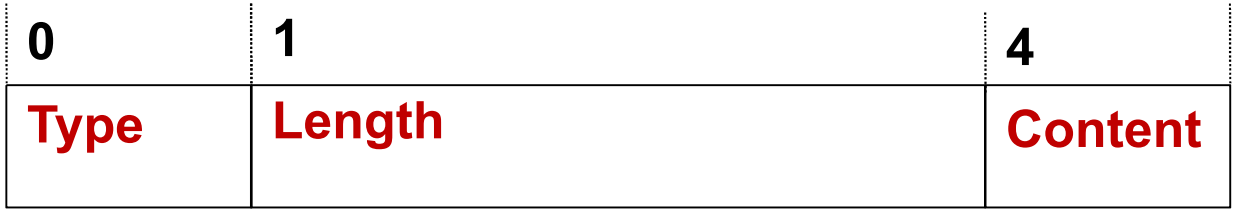
- Die Handshake-Nachrichten ermöglichen den **Aufbau einer SSL/TLS-Verbindung**.
- Dient u.a. zur
  - **Identifikation und Authentifizierung** der Kommunikationspartner, sowie
  - zum **Aushandeln kryptographischer Algorithmen, Schlüssel und Parameter**,

    die u.a. im SSL/TLS Record Layer Protokoll verwendet werden und zum Austausch benötigter geheimer Informationen.



SSL/TLS – Handshake Protokoll
→ Aufbau

- Gesamtgröße: 5 Byte



- Type (1 Byte): Zeigt eine von 10 möglichen Nachrichten an (siehe Abb.)
- Length (3 Byte): Länge der Nachricht in Bytes
- Content (1 Byte): Parameter, welche mit dieser Nachricht assoziiert sind

SSL/TLS – Handshake Protokoll
→ Nachrichten-Typen

Client Server Nachrichten-Typen

→ **ClientHello** (Zufallswerte, Sitzungsnummer, Liste Cipher Suites, …)

← **ServerHello** (Zufallswerte, ausgewählte Cipher Suites, …)
← **Certificate** (optional (aber fast immer), Zertifikat des Servers)
← **ServerKeyExchange** (optional, temp. RSA Schlüssel des Servers)
← **CertificateRequest** (optional, Anfrage einer Client Authentikation)
← **ServerHelloDone** (Ende dieser Phase)

→ **Certificate** (optional, Zertifikat des Clients)
→ **ClientKeyExchange** (Verschl. Pre-Master Secret oder DH-Key)
→ **CertificateVerify** (optional, Signatur des Clients)
→ **ChangeCipherSpec** (CCS-Protokoll)
→ **Finished** (MAC)

← **ChangeCipherSpec** (CCS-Protokoll)
← **Finished** (MAC)

SSL/TLS
→ Reservierte Ports

- Um das SSL/TLS-Protokoll zwischen Client und Server abzuwickeln, müssen sich beide Kommunikationspartner zunächst über die Verwendung des SSL/TLS-Protokolls verständigen.

- Die eigentlichen Protokolldaten werden anstatt im eigenen Anwendungsprotokoll (HTTP, SMTP, IMAP, …) über das **Application Data Protokoll** von SSL/TLS verschlüsselt übertragen.

- Anhand der **Portnummer** lässt sich erkennen, um welches ursprüngliche Anwendungsprotokoll es sich bei den übertragenen Daten handelt.

- Dazu wurden für SSL/TLS-basierte Anwendungsdienste spezielle Portnummern reserviert, über die diese Anwendungsdienste abgewickelt werden.

SSL/TLS
→ Übersicht der unterstützten Anwendungen (1/2)

- Name: **https - Hypertext Transfer Protocol Secure**
  Browsererkennung: https://...
  RFC: 2818
  Default Port: 443 (statt 80 bei http)

- Name: **pop3s – Post Office Protocol Secure**
  RFC: 2595
  Default Port: 995 (statt 110 bei pop3)

- Name: **smtp/ssl – Simple Mail Transfer Protocol Secure**
  RFC: 2487
  Default Port: 465 (statt 25 bei smtp)

- Name: **imap4/ssl – Internet Message Access Protocol Secure**
  RFC: 2595
  Default Port: 993 (statt 143 bei imap4)

SSL/TLS
→ Übersicht der unterstützten Anwendungen (2/2)

- Name: **ftps – File Transfer Protocol Secure**
  RFC: 4217
  Default Port: 989 (data), 990 (control) (statt 20,21 bei ftp)

- Name: **telnets – Teletype Network Secure**
  RFC: 2839
  Default Port: 992 (statt 23 bei telnet)

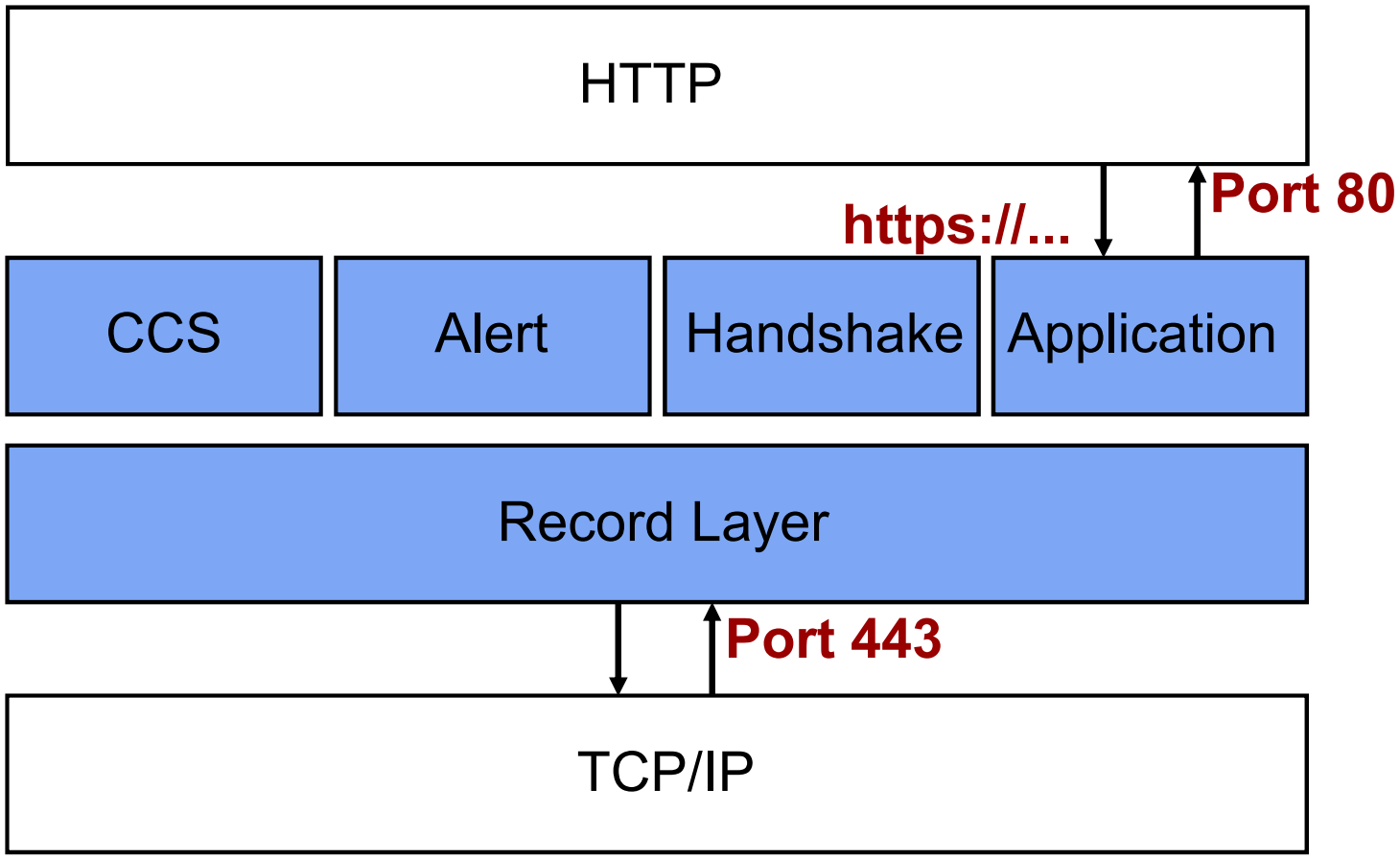
- Name: **sips – Session Initiation Protocol Secure**
  RFC: 3261
  Default Port: 5061 (statt 5060 bei sip)

- weitere Protokolle: irc, nntp, ...

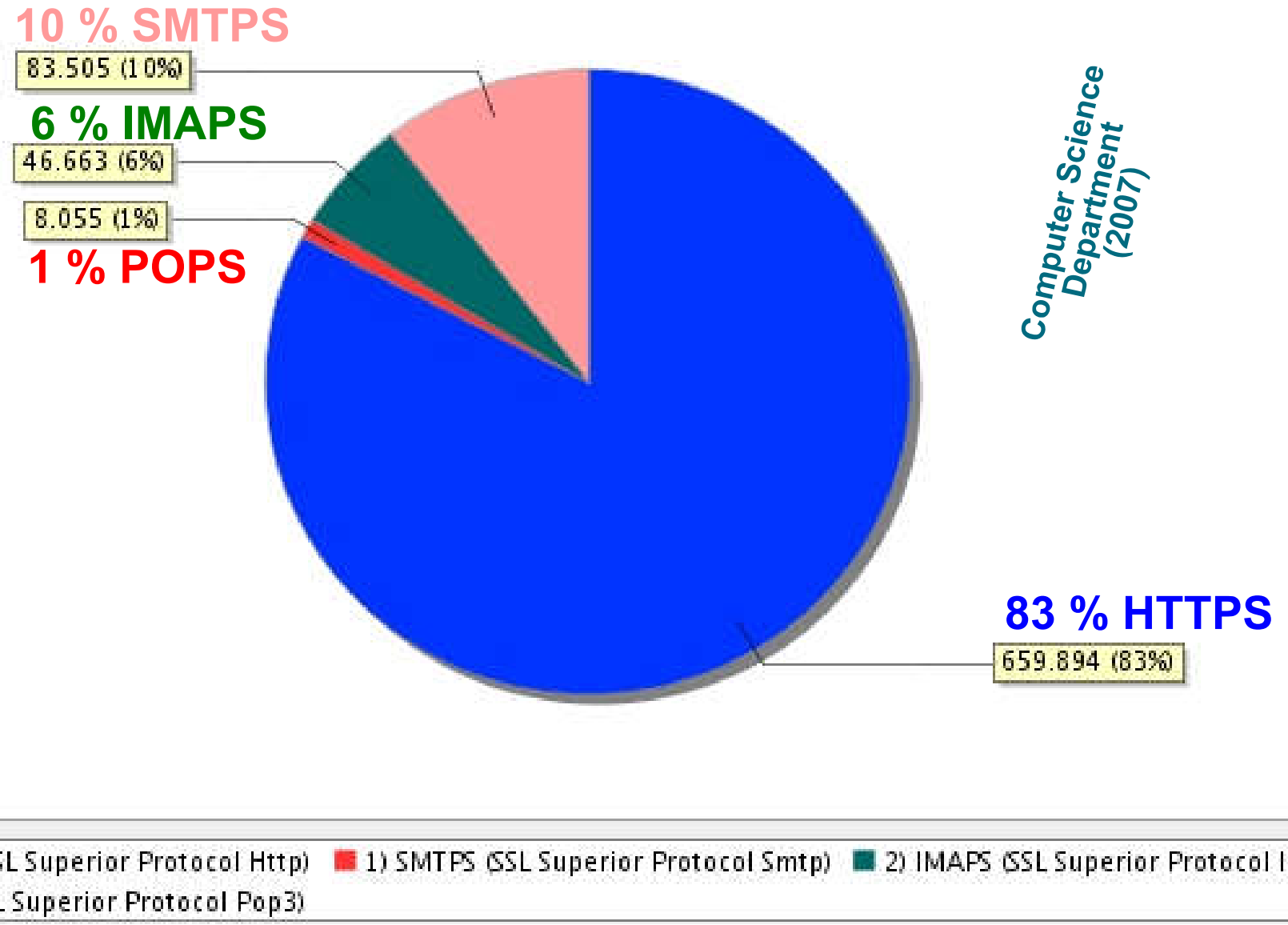
- Theoretisch kann jedes Anwendungsprotokoll SSL/TLS zur Verschlüsselung und Übertragung seiner Daten nutzen.

SSL/TLS
→ Beispiel: HTTP/HTTPS

- Aufbau der SSL/TLS-Schichten am Beispiel von https:

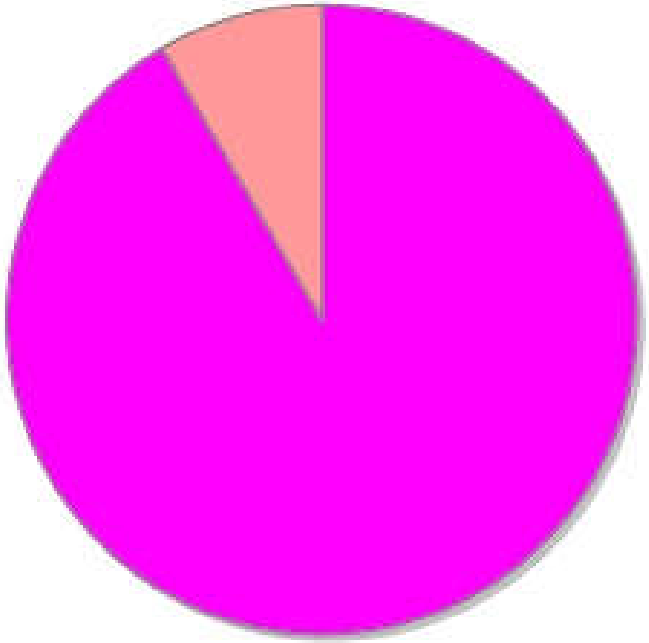


Verteilung gängigste Protokolle die SSL/TLS nutzen



Verhältnis SSL/TLS ⇔ ohne SSL/TLS

- Der Anteil an Protokollen, die SSL/TLS nutzen, ist noch sehr gering.
- Bsp. Verhältnis http zu https



Bei anderen Protokollen ist das Verhältnis noch weit schlechter und liegt meist etwa bei 99:1

Inhalt

- Einleitung: Definitionen und Ziele
- SSL/TLS Header und Nachrichtentypen
- **Protokollablauf**
- Bedeutung von Zertifikaten in Browsern
- Authentikationsmethoden
- Schwachstellen und Angriffsmöglichkeiten
- SSL/TLS in der Praxis
- Zusammenfassung

Protokollablauf
→ Informationsaustausch

- SSL/TLS ist ein zustandsbehaftetes Sicherheitsprotokoll, so dass Sitzungen zwischen Kommunikationspartnern etabliert werden können.
- Ein Client kann zu einem Zeitpunkt mehrere solche Sitzungen zum gleichen oder zu verschiedenen Servern unterhalten.
- SSL/TLS benutzt eine Session, um Zustandsinformationen über einen längeren Zeitraum zu speichern und nutzen zu können.
- Wie bei IPSec nutzt SSL/TLS für die bidirektionale Verbindung zwischen Client/Server zwei unterschiedliche Sitzungsschlüssel.
- Kryptographische Verfahren und Hashfunktion werden pro Sitzung ausgehandelt.
- SSL/TLS versucht möglichst wenig geheime Informationen über das nicht vertrauenswürdige Transportsystem Internet zu übertragen.
- Daher werden lediglich Basisinformationen ausgetauscht, womit die beteiligten Kommunikationspartner dezentral ihre Geheimnisse, wie die gemeinsamen Session Keys und MAC-Schlüssel berechnen.

Sitzungs- und Verbindungskonzepte
→ SSL/TLS-Session (1/2) – Übersicht

- Eine SSL/TLS-Session ist eine **Assoziation zwischen einem Client und einem Server**.
- Sessions werden über das Handshake-Protokoll aufgebaut.
- Eine Session definiert eine **Menge von kryptologischen Sicherheitsparametern**, welche gemeinsam über mehrere Verbindungen genutzt werden können.
- Der Sinn hinter der Session besteht darin, nicht jedes Mal eine neue zeitaufwendige Verhandlung der Sicherheitsparameter auszuführen.

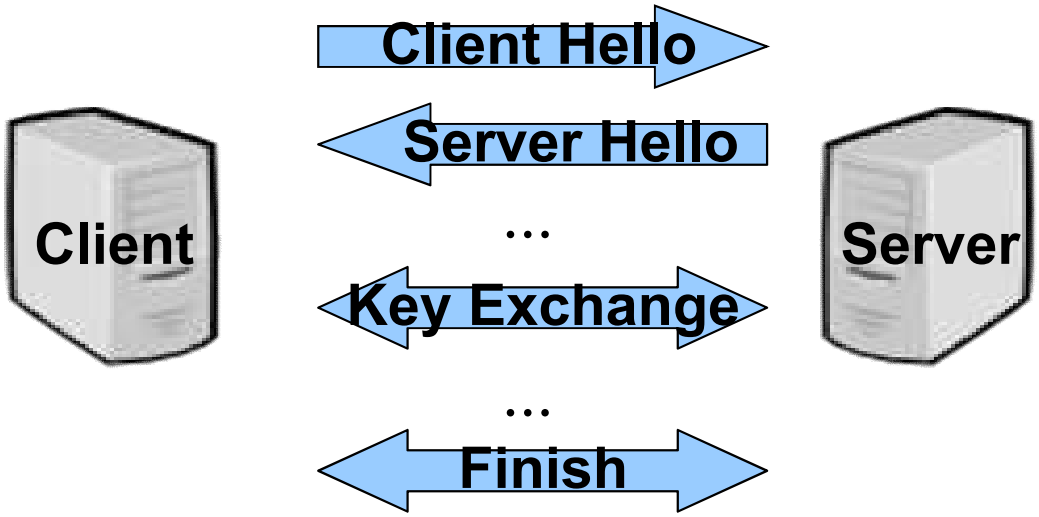


Abb.: Aushandeln einer Session zwischen Client und Server

Sitzungs- und Verbindungskonzepte
→ SSL/TLS-Session (2/2) – Session Zustände

- Ein **Sessionzustand** ist definiert über folgende Parameter:
  - Session Identifier: Eine zufällige Bytesequenz, vom Server erzeugt, um eine aktive oder wiederherstellbare Session zu identifizieren.
  - Peer certificate: Ein X509.v3 Zertifikat des Clients. Dieses Element kann Null sein.
  - Compression method: Der Kompressionsalgorithmus, welcher vor der Verschlüsselung benutzt wird.
  - Cipher spec: Spezifiziert den Verschlüsselungsalgorithmus, sowie den Hash-Algorithmus für den MAC. Zusätzlich werden noch Attribute, wie z.B. Hashlänge, definiert.
  - Master secret: 48-Byte Geheimnis, welches vom Client und Server benutzt wird.
  - Is resumable: Flag->zeigt an, ob diese Session für neue Verbindungen genutzt werden kann.

Sitzungs- und Verbindungskonzepte
→ SSL/TLS-Connection (1/3) – Übersicht

- Eine SSL/TLS-Connection ist ein **Transportkanal**, welcher einen spezifizierten Service bietet.
- Bei SSL/TLS handelt es sich um eine Peer-to-Peer Verbindung, welche nur vorübergehend aufgebaut sein kann.
- Jede **Connection** muss mit einer **Session** assoziert sein.
- Somit können **aus einer Session mehrere Connections** aufgebaut werden, welche auch parallel betrieben werden können.

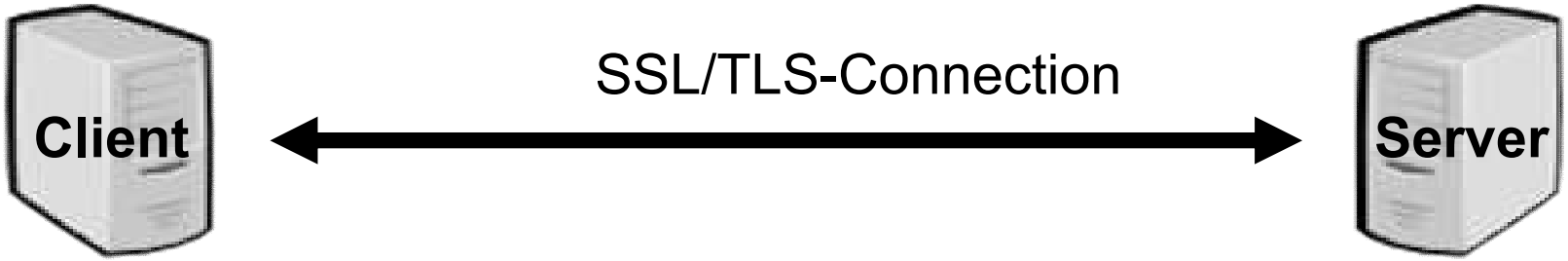


Abb. : SSL/TLS-Connection: Logischer Kanal zwischen Client und Server

Sitzungs- und Verbindungskonzepte
→ SSL/TLS-Connection (1/2) – Zustände

- Server and Client random: Byte-Sequenzen (Zufallszahlen), welche von Server und Client für jede Verbindung erzeugt werden.
- Server write MAC secret: Der geheime Schlüssel, der bei MAC-Operationen von Daten, die zum Server gehen, benutzt wird.
- Client write MAC secret: Der geheime Schlüssel, der bei MAC-Operationen von Daten, die zum Client gehen, benutzt wird.
- Server write key: Symmetrischer Schlüssel, der zur Verschlüsselung vom Server und bei der Entschlüsselung vom Client benutzt wird.
- Client write key: Symmetrischer Schlüssel, der zur Verschlüsselung vom Client und bei der Entschlüsselung vom Server benutzt wird.
- Initialization vectors: Wenn ein Blockchiffre in CBC-Mode benutzt wird, so wird ein Initialization Vector (IV) für jeden Schlüssel beibehalten.

Sitzungs- und Verbindungskonzepte
→ SSL/TLS-Connection (2/2) – Zustände

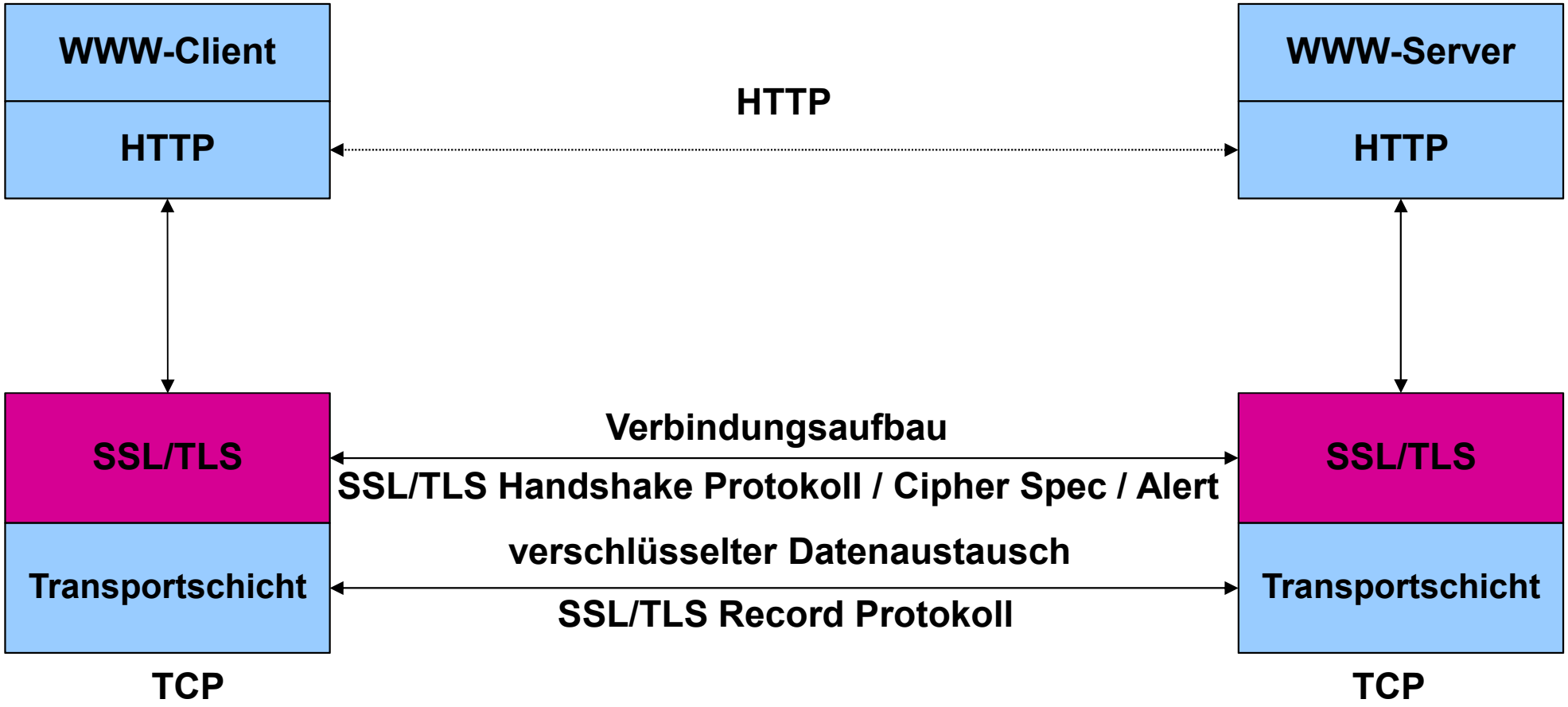
- Sequence numbers:
  Beide Parteien verwalten eigene Sequenznummern für die gesendeten und empfangenen Nachrichten jeder Verbindung.
  Falls einer der Parteien ein Change cipher spec verschickt oder erhält, so wird die Sequenznummer wieder auf Zero gesetzt.
- Sequenznummern können nicht größer als 2^{64} -1 werden.

Sitzungs- und Verbindungskonzepte
→ SSL/TLS: Session/Connection ⇔ Verfahren/Schlüssel

- An den Zuständen der Verbindungen wird deutlich, dass die verwendeten Schlüssel jeweils nur für eine Verbindung gelten.
- Für alle Verbindungen einer Sitzung werden die gleichen Verfahren genutzt.
- D.H. bei der erneuten Verwendung einer bereits bestehenden Verbindung kann auf das Aushandeln der Verfahren im Handshake verzichtet werden.

Protokollablauf
→ Übersicht des Protokollablaufs bei SSL/TLS

- Übersicht des Protokollablaufs bei SSL/TLS am Bsp. http:



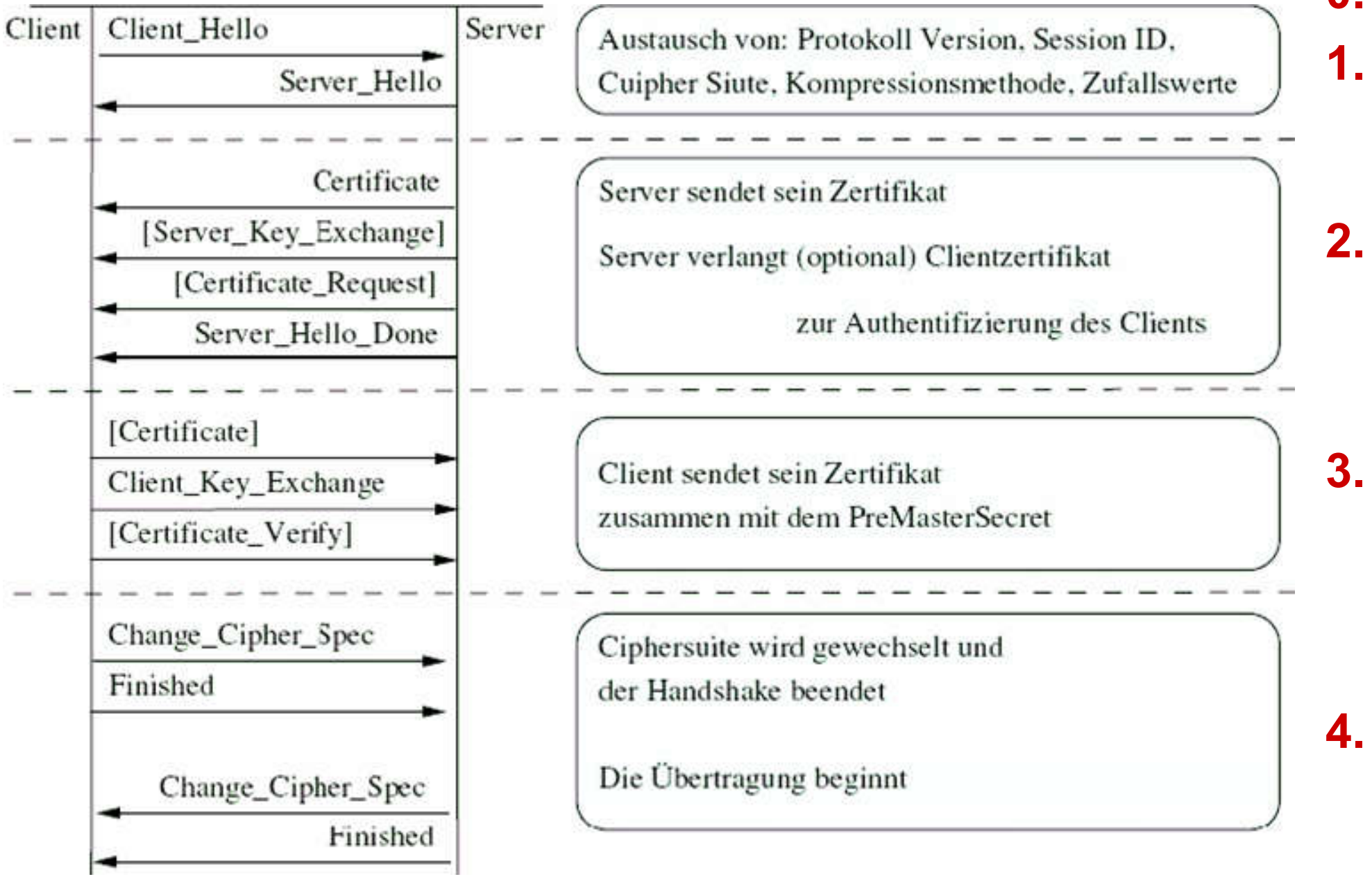
Protokollablauf
→ Schichten und Phasen

- Der Protokollablauf bei SSL/TLS erfolgt in zwei Schritten.
- **1. Schritt:**
  Verbindungsaufbau, unterteilt in **4 Phasen**:

  - *1. Phase:* Aushandlung der Sicherheitsparameter.
  - *2. Phase:* Serverauthentisierung (Optional) und Schlüsselaustausch
  - 3. *Phase:* Clientauthentisierung (Optional) und Schlüsselaustausch
  - *4. Phase:* Beendigung des Handshakes

- **2. Schritt:**
  Verschlüsselte und integritätsgesicherte Datenübertragung

SSL/TLS
→ Verbindungsaufbau vor der Datenübertragung



SSL/TLS - Verbindungsaufbau
→ Phase 0: HelloRequest

- Der Server **kann** ein “HelloRequest (0)” immer zum Client senden.
- Diese Nachricht hat keine Parameter.
- Sie wird vom Server geschickt, um den Client zu einem „ClientHello“ zu veranlassen.
  - Client antwortet nur, wenn er sich nicht in einem Handshake befindet.
- Soll nicht zum Aufbau einer Verbindung genutzt werden(!), dies ist die Aufgabe des Clients mit ClientHello (nächste Folie).
- Erhält der Server auf ein HelloRequest keine Antwort, **kann** die Verbindung geschlossen werden.

SSL/TLS - Verbindungsaufbau
→ Phase 1: ClientHello

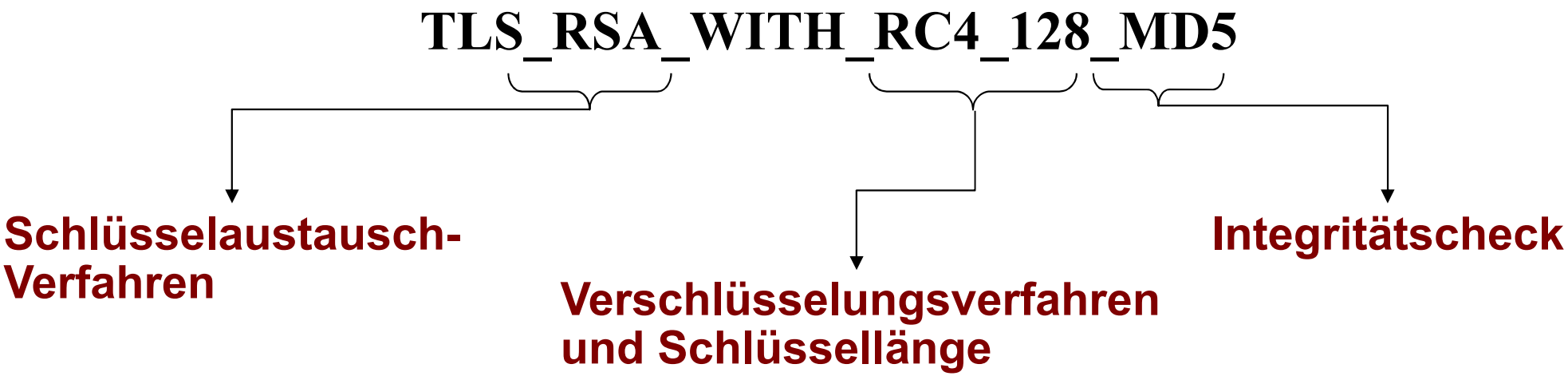
- In der ersten Phase werden die Sicherheitsparameter festgelegt.
- Mit der „**ClientHello (1)**“ Nachricht startet der Client den Aufbau der SSL/TLS-Verbindung.
- In einer bereits aktiven SSL/TLS-Verbindung führt diese Nachricht zu einer Neuverhandlung der Sicherheitsparameter.
- In dieser Klartextnachricht sind bereits wichtige Informationen zur Erzeugung des gemeinsamen geheimen Schlüssels enthalten:
  - RC (4 Byte Zeitstempel + 28 Byte Zufallszahl)
  - Sitzungsidentifikation,
  - Prioritätenliste der Cipher Suites (Kryptographie- und Kompressionsverfahren), welche der Client unterstützt.

SSL/TLS - Verbindungsaufbau
→ Phase 1: ServerHello

- Mit der **„ServerHello (2)“** Nachricht antwortet der Server auf das „ClientHello“ des Client.
- Die Nachricht erhält die gleichen Parameter, wie die „ClientHello“ Nachricht, teilweise allerdings mit leicht abgewandelter Bedeutung.
  - RS (32 Bit Zeitstempel + 28 Byte Zufallszahl)
- Zudem enthält die Nachricht die ausgewählte Cipher Suite **vom Server**, welche Client und Server nachfolgend nutzen.
- Hat der Client eine Sitzungsnummer vorgeschlagen, überprüft der Server diese Sitzungsnummer und übernimmt sie, sofern diese Sitzung noch nicht zu lange zurück liegt.
- Gibt der Server keine Sitzungsnummer an, so bedeutet dies, dass die gegenwärtige Sitzung später nicht erneut aufgenommen wird.

SSL/TLS
→ Cipher Suites

- Beim **“Server/Client Hello”** werden die sogenannten **Cipher Suites** ausgewählt.
- Dabei handelt es sicht um eine Kombination aus Schlüsselaustauschverfahren, Verschlüsselungsverfahren mit Schlüssellänge und ein Verfahren zum Integritätscheck.
- Aufbau einer Cipher Suite an einem Bsp.:



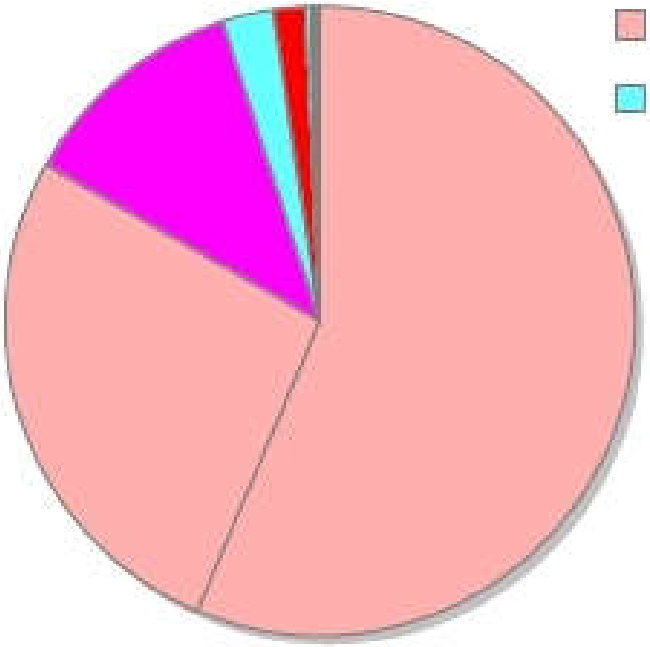
SSL/TLS – Cipher Suites
→ Unterstützte Algorithmen (1/2)

- In den RFCs sind die möglichen Cipher-Suites definiert.
- Jede Cipher-Suite definiert mögliche Kombinationen aus Schlüsselaustausch- und Authentifizierungsalgorithmus, Verschlüsselungsalgorithmus (inklusive Schlüssellänge) und einem MAC-Algorithmus (HMAC).
- Eine Liste der Cipher-Suites, welche der jeweilige Kommunikationsteilnehmer kennt, wird zwischen den beiden Kommunikationspartnern ausgetauscht.
- **MAC- Algorithmen:**
  - MD5 (bei neueren Cipher-Suites wird MD5 nicht mehr verwendet)
  - SHA-1, SHA256
- **Schlüsselaustausch:**
  - RSA, **DH-RSA**, DH-DSS, DHE-RSA, DHE-DSS, RSAExport

SSL/TLS – Cipher Suites → Unterstützte Algorithmen (2/2)

- **Verschlüsselung:**
  - Null – keine Verschlüsselung
  - DES40-CBC – Exportversion (alt)
  - DES-CBC (alt)
  - 3DES-EDE-CBC
  - RC2-CBC-40 – Exportversion (alt)
  - RC4-40 – Exportversion (alt)
  - RC4-128
  - IDEA-CBC (alt)
  - AES-128-CBC
  - AES-256-CBC
  - Fortezza (wird bei TLS nicht mehr verwendet)

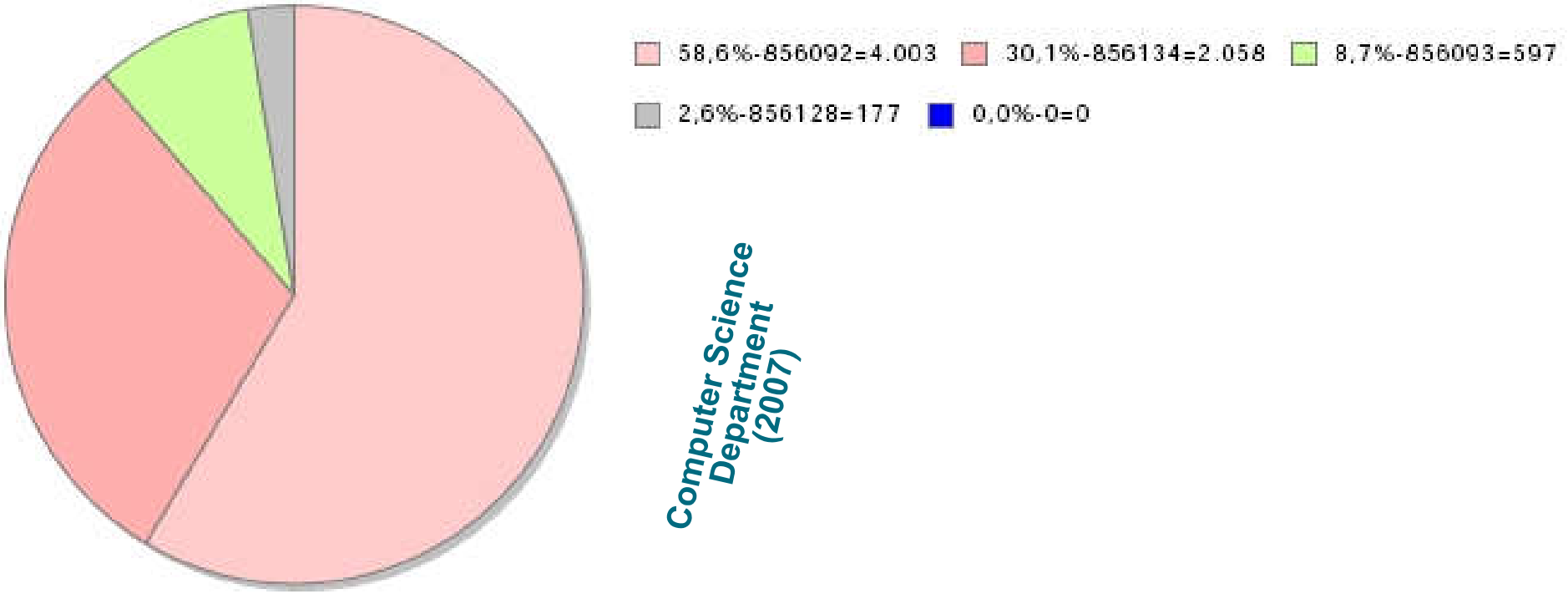
HTTPS – Verteilung meistgenutzte Cipher



659484	TLS_RSA_WITH_RC4_128_MD5)	56,4 %
659485	TLS_RSA_WITH_RC4_128_SHA)	1,6 %
659490	TLS_RSA_WITH_3DES_EDE_CBC_SHA)	2,6 %
659502	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA)	
659515	TLS_RSA_EXPORT1024_WITH_RC4_56_SHA)	
659520	TLS_RSA_WITH_AES_128_CBC_SHA)	0,2 %
659524	TLS_DHE_RSA_WITH_AES_128_CBC_SHA)	
659526	TLS_RSA_WITH_AES_256_CBC_SHA)	11,8 %
659530	TLS_DHE_RSA_WITH_AES_256_CBC_SHA)	26,9 %
659536	SSL_TRIPLE_DES_SHA_US)	0,3 %

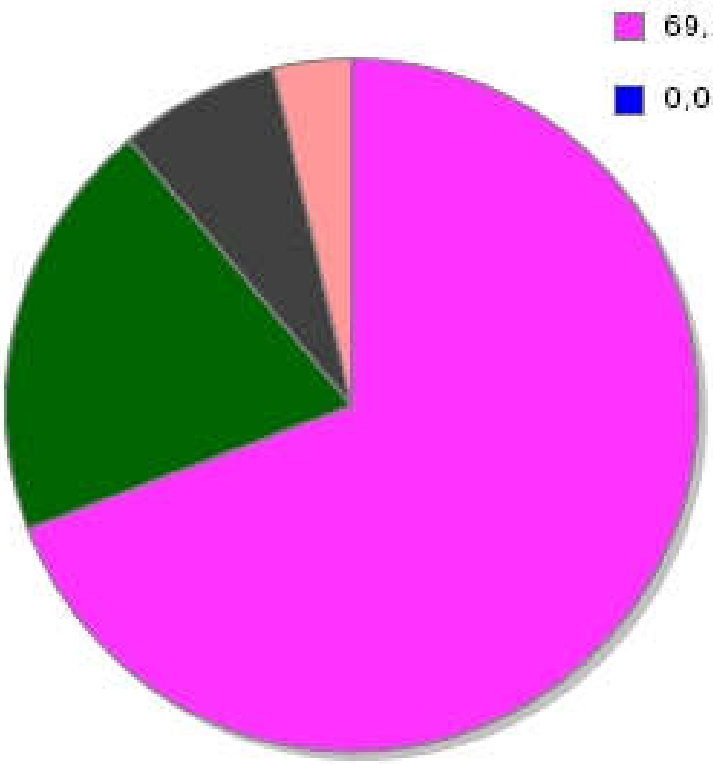
Computer Science Department (2007)

POPS – Verteilung meistgenutzte Cipher



	856092 TLS_RSA_WITH_RC4_128_MD5)	58,6 %
	856093 TLS_RSA_WITH_RC4_128_SHA)	8,7 %
	856128 TLS_RSA_WITH_AES_128_CBC_SHA)	2,6 %
	856134 TLS_RSA_WITH_AES_256_CBC_SHA)	30,1 %

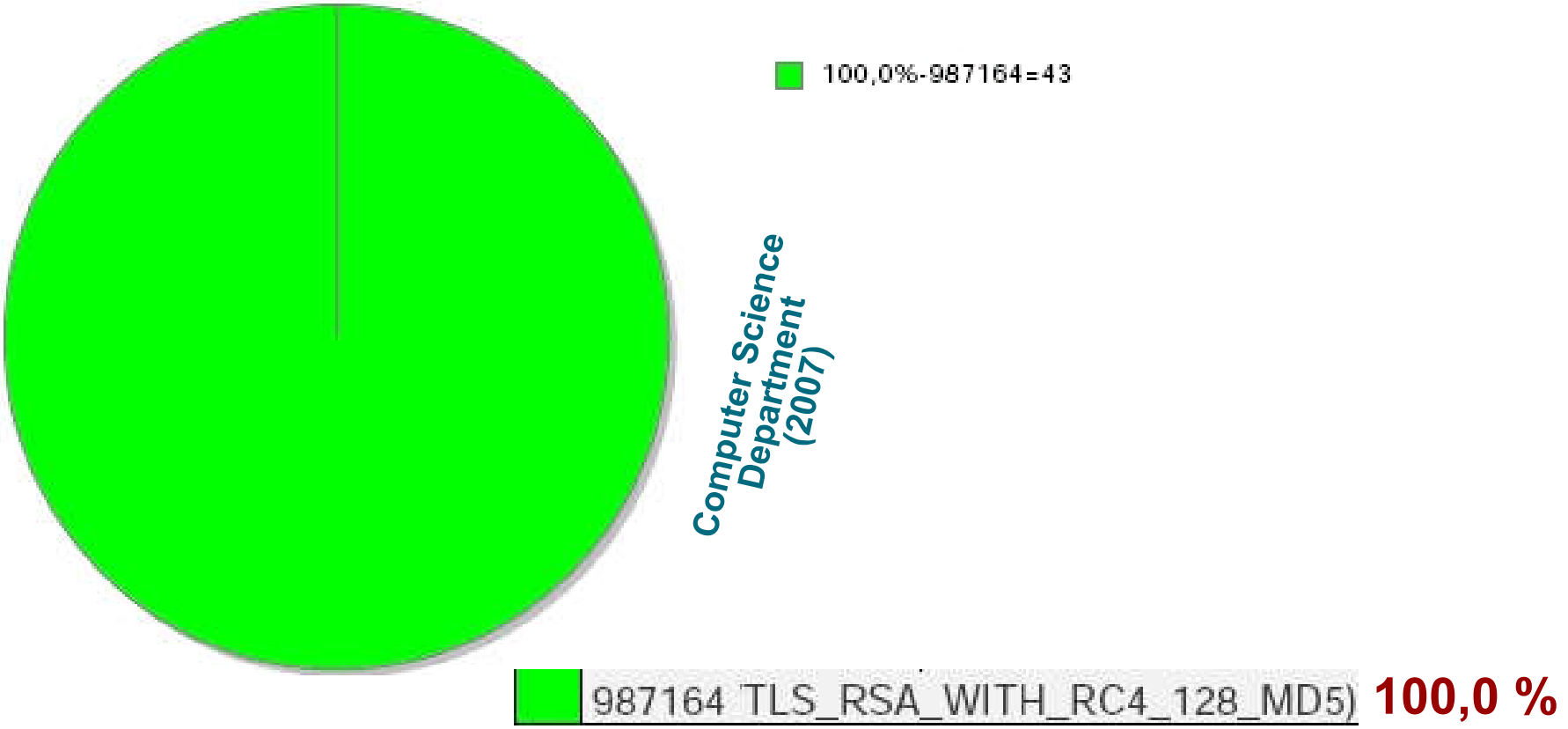
IMAPS – Verteilung meistgenutzte Cipher



Computer Science Department (2007)

	921628 TLS_RSA_WITH_RC4_128_MD5)	19,7 %
	921629 TLS_RSA_WITH_RC4_128_SHA)	3,6 %
	921664 TLS_RSA_WITH_AES_128_CBC_SHA)	7,5 %
	921670 TLS_RSA_WITH_AES_256_CBC_SHA)	69,2 %

SMTPS – Verteilung meistgenutzte Cipher



SSL/TLS - Verbindungsaufbau
→ Phase 2: Certificate

- In der zweiten Phase findet Serverauthentisierung und Schlüsselaustausch statt.
- Soll der Server authentifiziert werden, so muss er dem Client sein Zertifikat zukommen lassen **“Certificate (11)”**.
- Beim Server handelt es sich im Normalfall um ein X.509v3-Zertifikat, welches für eine Domäne einer Organisation von einer Zertifizierungsinstanz ausgegeben wurde.
- Dabei muss das Certificate zu der Cipher Suite passen!
  Wird RSA verwendet, muss das Zertifikat einen RSA-Schlüssel enthalten etc.

SSL/TLS - Verbindungsaufbau
→ Phase 2: ServerKeyExchange

- Diese Meldung wird nicht gesendet, wenn der Server ein Zertifikat mit festen Diffie-Hellman Parametern oder mit einem RSA-Schlüssel besitzt.
- Versendet der Server kein Zertifikat, so lässt er dem Client in der **“ServerKeyExchange (12)”** Nachricht einen temporären öffentlichen RSA Schlüssel zukommen.

SSL/TLS - Verbindungsaufbau
→ Phase 2: CertificationRequest

- Falls der Server auch eine Authentikation des Client fordert, sendet er eine **“CertificationRequest (13)”** Nachricht.

SSL/TLS - Verbindungsaufbau
→ Phase 2: ServerHelloDone

- Der Server beendet seine Übertragung dieser Phase mit seiner **“ServerHelloDone (14)”** Nachricht.
- Diese Nachricht ist notwendig, da die Nachrichten „Certificate“, „ServerKeyExchange“ und „CertificateRequest“ nicht notwendigerweise geschickt werden.
- So erkennt der Client das Ende der Phase 2.

SSL/TLS - Verbindungsaufbau
→ Phase 3: Certificate

- In der Phase 3 kann der Client vom Server authentisiert werden und der Schlüsselaustausch wird fortgeführt.
- Soll der Client authentifiziert werden, so muss er dem Server sein Zertifikat zukommen lassen **“Certificate (11)”**.

SSL/TLS - Verbindungsaufbau
→ Phase 3: ClientKeyExchange

- Der Client prüft zunächst die Gültigkeit des Server-Zertifikats.
- Anschließend übermittelt er dem Server mit der **“ClientKeyExchange (16)”** Nachricht seine geheime Basisinformation mit Hilfe des Kryptoverfahrens aus der ausgewählten Cipher Suite, das 48 Byte **Pre-Master Secret (Pre).**
- Haben sich Client/Server auf das RSA Verfahren geeinigt, so verschlüsselt der Client das Pre-Master Secret mit dem öffentlichen Schlüssel des Servers aus dem **“Certificate”** (Server-Zertifikat) oder aus dem **“ClientKeyExchange”**.
- Beim Diffie-Hellman-Schlüsselaustauschverfahren sendet der Client nur seinen öffentlichen Schlüssel an den Server zurück.
- Das Diffie-Hellman-Verfahren wird hier nicht zur Berechnung des geheimen Schlüssels genutzt, sondern zur dezentralen Berechnung des Pre-Master Secrets (Pre).

SSL/TLS - Verbindungsaufbau
→ Phase 3: CertificateVerify

- Mit der Nachricht **„CertificateVerify (15)“** signiert der Client Informationen, sofern er es in dieser Phase gesendet hat.

- Client „beweist“ dem Server dass er im Besitz des privaten Schlüssels für das Zertifikat ist. Ablauf:
  - Client bildet Hashwert über die Bytes aller bisher ausgetauschten Handshake-Nachrichten, beginnend mit ClientHello bis einschließlich ClientKeyExchange.
  - Client signiert diesen Hashwert mit seinem privaten Schlüssel.
  - Server bildet den gleichen Hashwert und überprüft die Signatur mit Hilfe dieses Wertes und dem öffentlichen Schlüssel aus dem Zertifikat.

SSL / TLS - Verbindungsaufbau
→ Master Secret Berechnung

- Das Pre-Master Secret und die ausgetauschten Zufallszahlen werden nun von Client und Server genutzt, um das 48 Byte Master Secret zu berechnen, aus dem die benötigten geheimen Schlüssel (Session Keys, Key zur MAC-Berechnung, ..) abgeleitet werden.
- Für diese Berechnung wurden die Hashfunktionen MD5 und SHA wie folgt genutzt:
- **Mastersecret = MD5 (Pre | SHA ('A‘ |Pre|Rc|Rs)) | MD5 (Pre | SHA ('BB‘ |Pre|Rc|Rs)) | MD5 (Pre | SHA ('CCC‘ |Pre|Rc|Rs))**
- In Version 1.2 wird die verwendete Funktion mit in der Ciphersuite ausgetauscht:
- **master_secret = PRF(pre_master_secret, "master secret", ClientHello.random + ServerHello.random) [0..47];**
- PRF = Pseudorandom Function (MD5, SHA-1, SHA-256, ...)

SSL / TLS - Verbindungsaufbau
→ Schlüsselerzeugung

- Alle Schlüssel (Session Keys, Key zur MAC-Berechnung, ..) werden nach dem einen gleichartigen Schema sowohl vom Client als auch vom Server aus dem Master Secret (ms) abgeleitet.
- Dazu generiert das Protokoll so lange eine Folge von Schlüsselblöcken key_block, bis alle Schlüssel damit konstruiert sind.

Alt:

- **key_block = MD5 (ms | SHA ('A‘ |ms|Rs|Rc)) |**
  MD5 (ms | SHA ('BB‘ |ms|Rs|Rc)) |
  MD5 (ms | SHA ('CCC‘ |ms|Rs|Rc))

Neu (Version 1.2):

- **PRF(master_secret, "key expansion",server_random + client_random)**
- PRF = Pseudorandom Function (MD5, SHA-1, SHA-256, ...)

SSL / TLS - Verbindungsaufbau
→ Phase 4: ChangeCipherSec + Finished

- Die Phase 4 beendet den Handshake.
- Mit der **“ChangeCipherSpec”** Nachricht (CCS-Protokoll) zeigen Client und Server, dass sie ab jetzt die ausgehandelten Verfahren/Parameter nutzen.
- Mit der **“Finished (20)”** Nachricht symbolisieren beide, dass ihr Teil des Verbindungsaufbaus abgeschlossen ist.
- Diese Meldung ist die erste Nachricht, die bereits mit den neuen Sicherheitsparametern für das Record Protokoll behandelt wird.

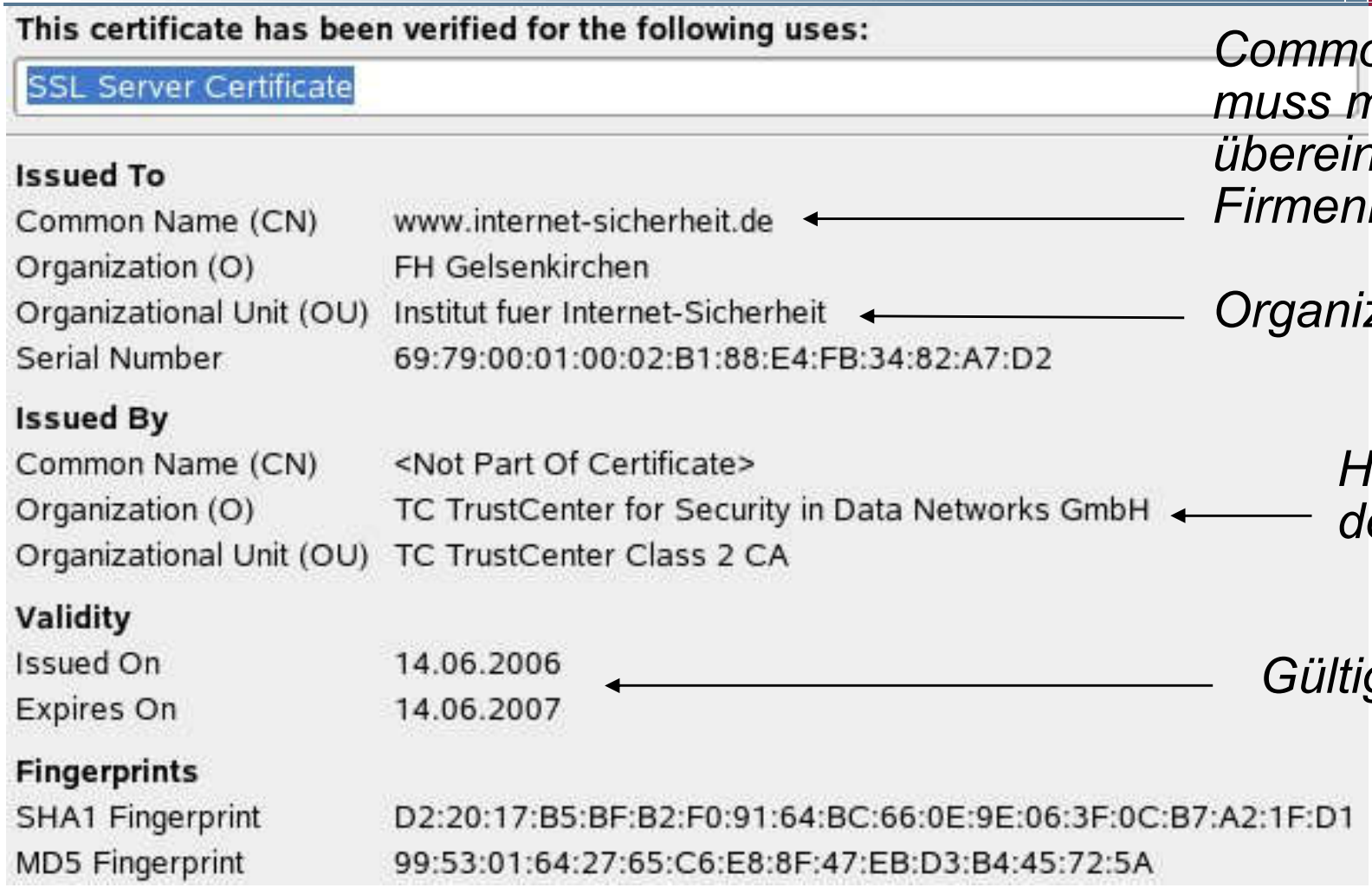
Inhalt

- Einleitung: Definitionen und Ziele
- Protokollaufbau: Header und Nachrichtentypen
- Protokollablauf
- **Bedeutung von Zertifikaten in Browsern**
- Authentikationsmethoden
- Schwachstellen und Angriffsmöglichkeiten
- SSL/TLS in der Praxis
- Zusammenfassung

SSL/TLS
→ Domänen-Zertifikate

- **Domänen-Zertifikate stellen eine Art “Ausweis” der Domäne dar.**
- **Wesentliche Inhalte eines Domänen-Zertifikates sind:**
  - Name der Organisation, deren Authentizität durch dieses Domänen-Zertifikat bestätigt wird
  - Public-Key der Organisation (Domäne)
  - Name der ausstellenden Zertifizierungsstelle.
- **Aufgabe eines Domänen-Zertifikats:**
  - Eindeutige Zuordnung eines Public-Key's zu einer Organisation.
  - Für die Korrektheit dieser Zuordnung und dass die Organisation, die den passenden privaten Schlüssel besitzt, auch existiert, verbürgt sich die ausstellende Zertifizierungsstelle.
  - Diese signiert das Domänen-Zertifikat, wodurch es für Dritte ohne die Kenntnis des privaten Schlüssels der Zertifizierungsstelle unmöglich wird, das Domänen-Zertifikat zu verändern.
- **Der Aufbau von Zertifikaten ist standardisiert (RFC 2459):**
  - Wichtigster Standard für digitale Zertifikate zurzeit X.509v3.

SSL/TLS – Domänen-Zertifikate
→ Zertifikatsinformationen



Common Name: muss mit der URL übereinstimmen, Firmennamen enthalten

Organizational Unit

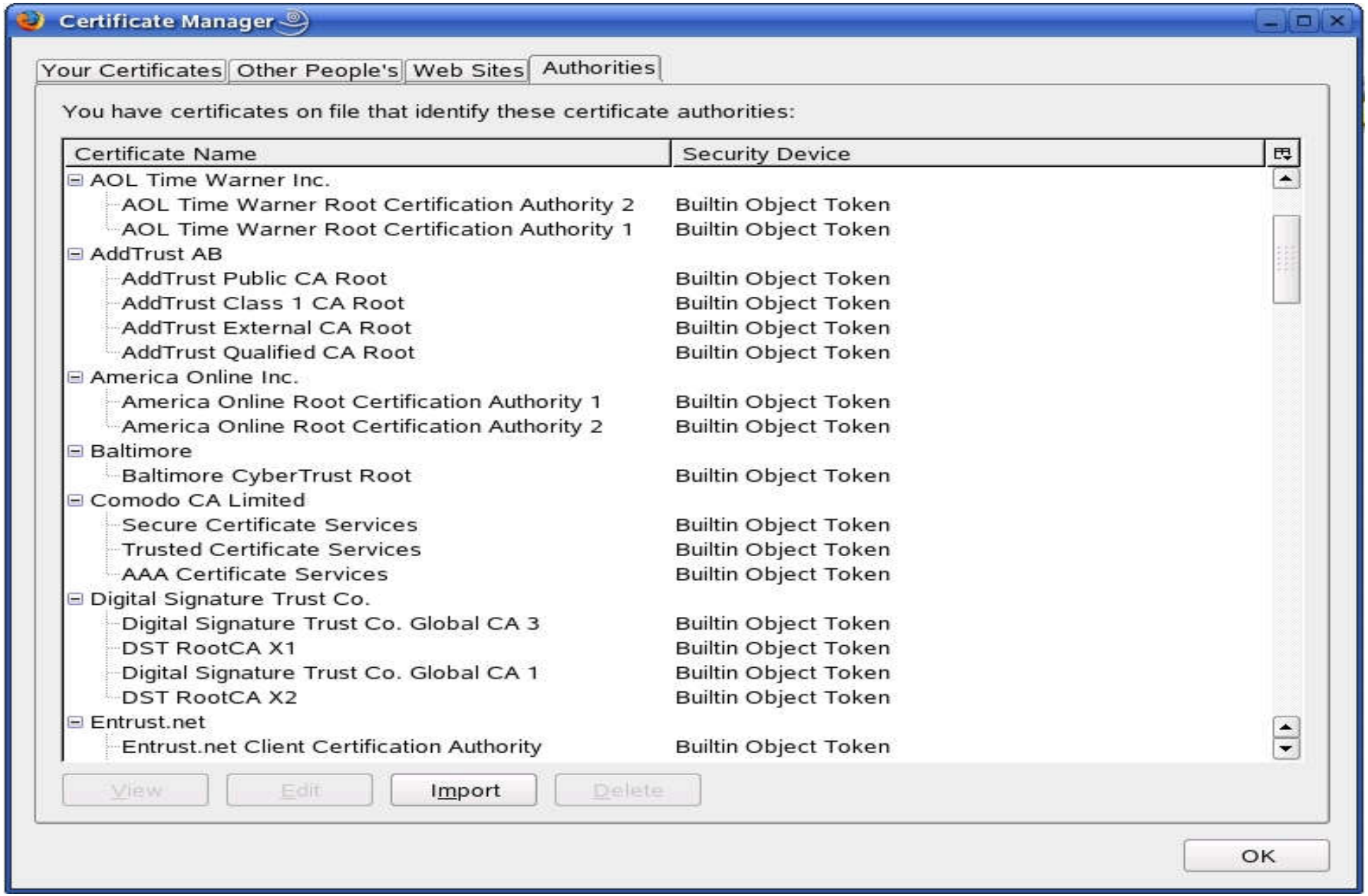
Herausgeber des Zertifikates

Gültigkeitszeitraum

SSL/TLS – Domänen-Zertifikate
→ Root-Zertifikat

- Ein Root-Zertifikat ist das oberste Zertifikat im Verzeichnisbaum.
- Damit ein Domain-Zertifikat vom Browser geprüft werden kann, muss dem Browser das verwendete Root-Zertifikat der Zertifizierungsstelle bekannt sein.
- Dazu können Root-Zertifikate zusätzlich zu den im Browser schon vorhandenen installiert werden.
- Die gängigsten Browser haben heutzutage schon die wichtigsten Root-Zertifikate der größten (zumeist amerikanischen) Zertifizierungsstellen vorinstalliert.
- Einmal akzeptierte Zertifikate können natürlich auch im Browser verwaltet, also aufgelistet und gegebenenfalls gelöscht werden.
- Besucht man eine Seite, dessen Zertifikat bzw. Zertifizierungsstelle dem Browser unbekannt ist, so erhält der Benutzer einen Hinweis, mit der Möglichkeit das Zertifikat zu prüfen, ihm einmalig oder dauerhaft zu vertrauen, oder das Zertifikat in den Browser zu laden, wodurch zukünftig an dieser Stelle kein Hinweis mehr erscheinen würde.

Root-Zertifikate in Browsern



Root-Zertifikate in Browsern

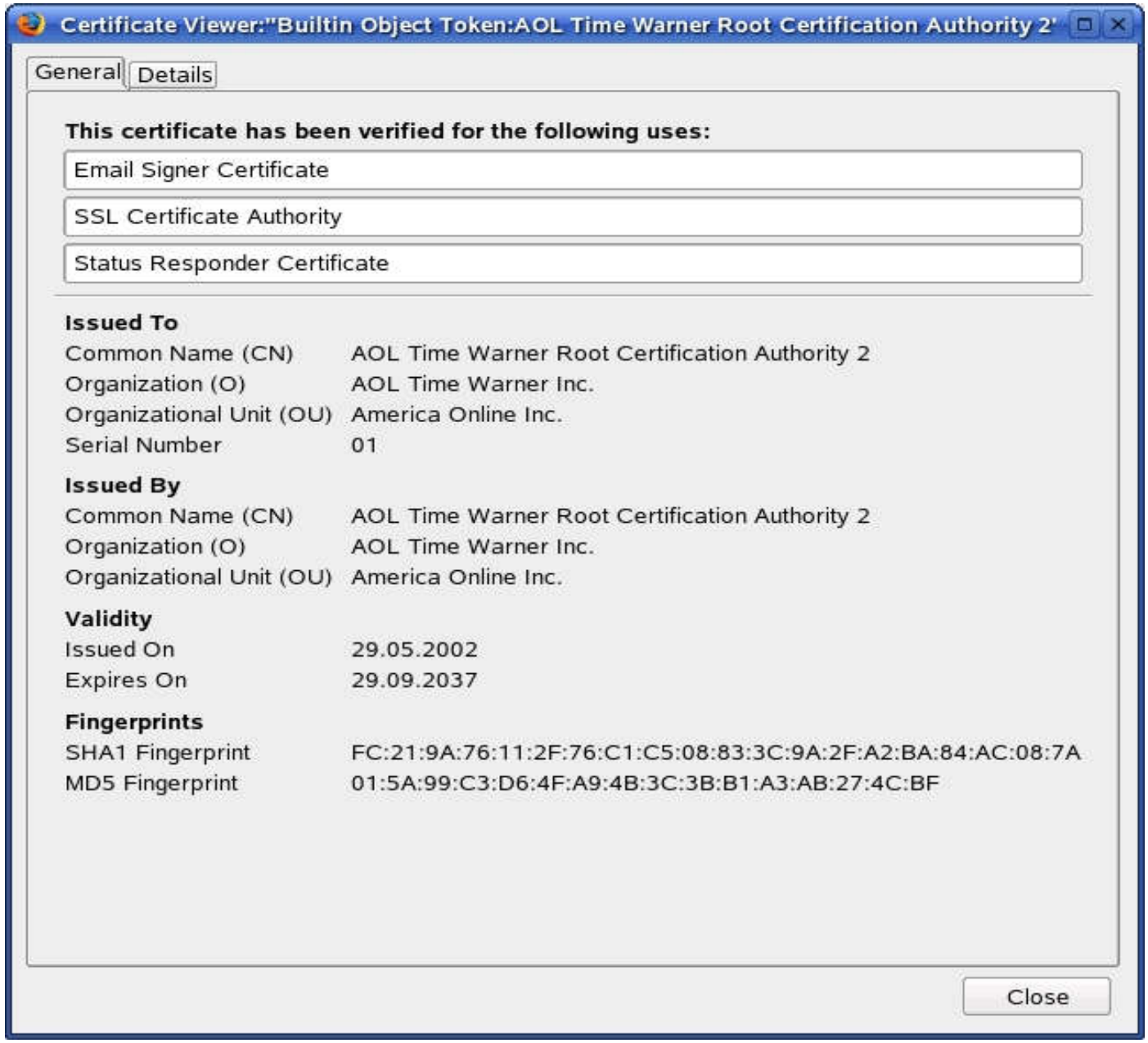
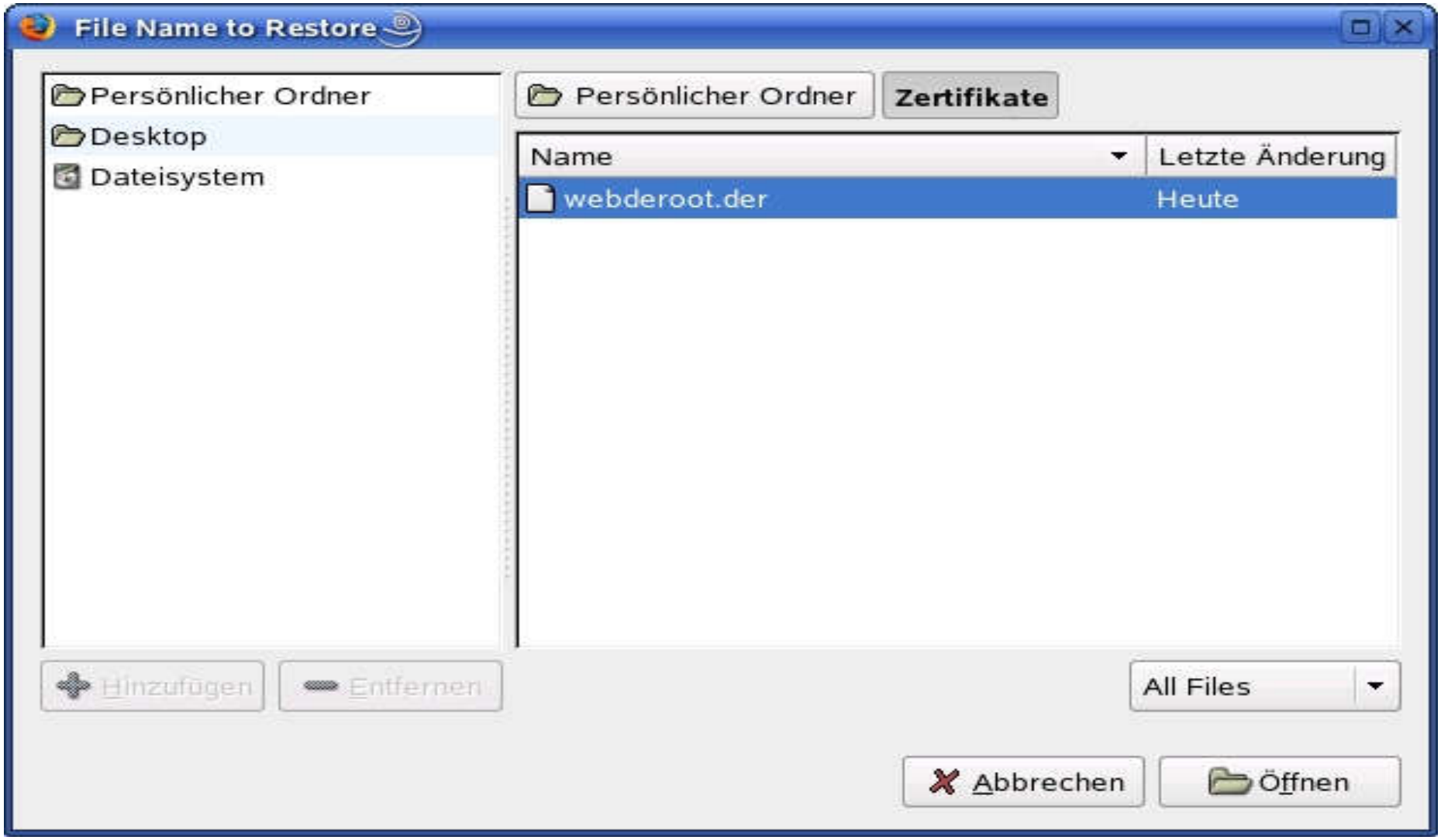


Abb.: Detaillierte Informationen eines Zertifikats anzeigen lassen

SSL/TLS – Domänen-Zertifikate
→ Neue Zertifikate im Browser einfügen (1/2)

- Neue Zertifikate lassen sich auf 2 Arten installieren:

1. Manuell importieren



SSL/TLS – Domänen-Zertifikate
→ Neue Zertifikate im Browser einfügen (2/2)

- 2. Zertifikat downloaden

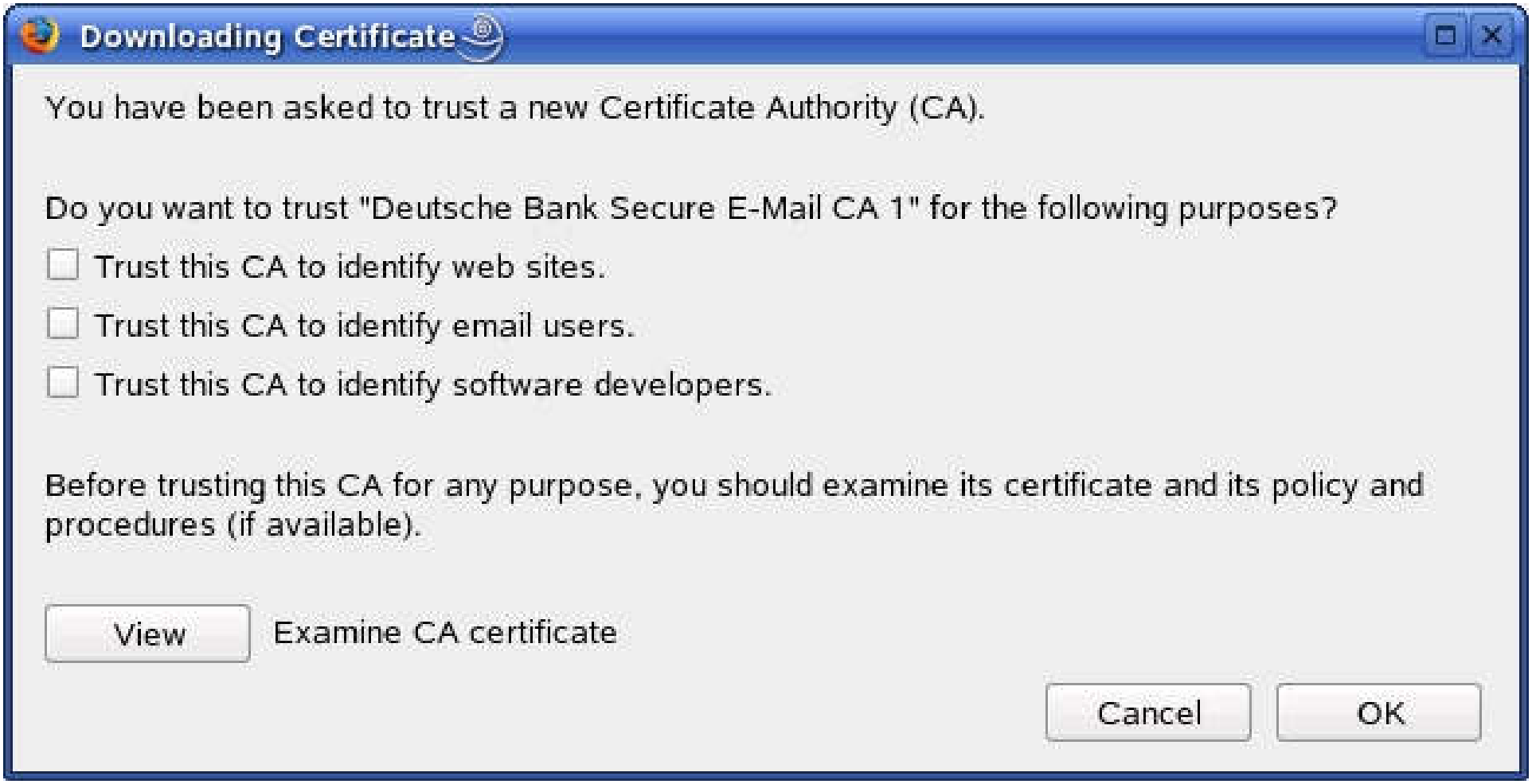


Abb.: Beim Download lässt sich festlegen, für welche Zwecke man das Zertifikat zulassen will

SSL/TLS – Domänen-Zertifikate
→ fehlende Zertifikate im Browser (1/2)

- Bei unbekannten/fehlenden Zertifikaten bieten sich 3 Möglichkeiten:

SSL/TLS – Domänen-Zertifikate
→ fehlende Zertifikate im Browser (2/2)

- **Die 3 Möglichkeiten im Detail:**
- 1. Dem Zertifikat permanent vertrauen.
- 2. Dem Zertifikat für die aktuelle Browser-Session vertrauen.
- 3. Das Zertifikat ablehnen.
  Dies führt in der Regel dazu, dass die Seite, welche das Zertifikat anbietet, nicht angezeigt werden kann.

Inhalt

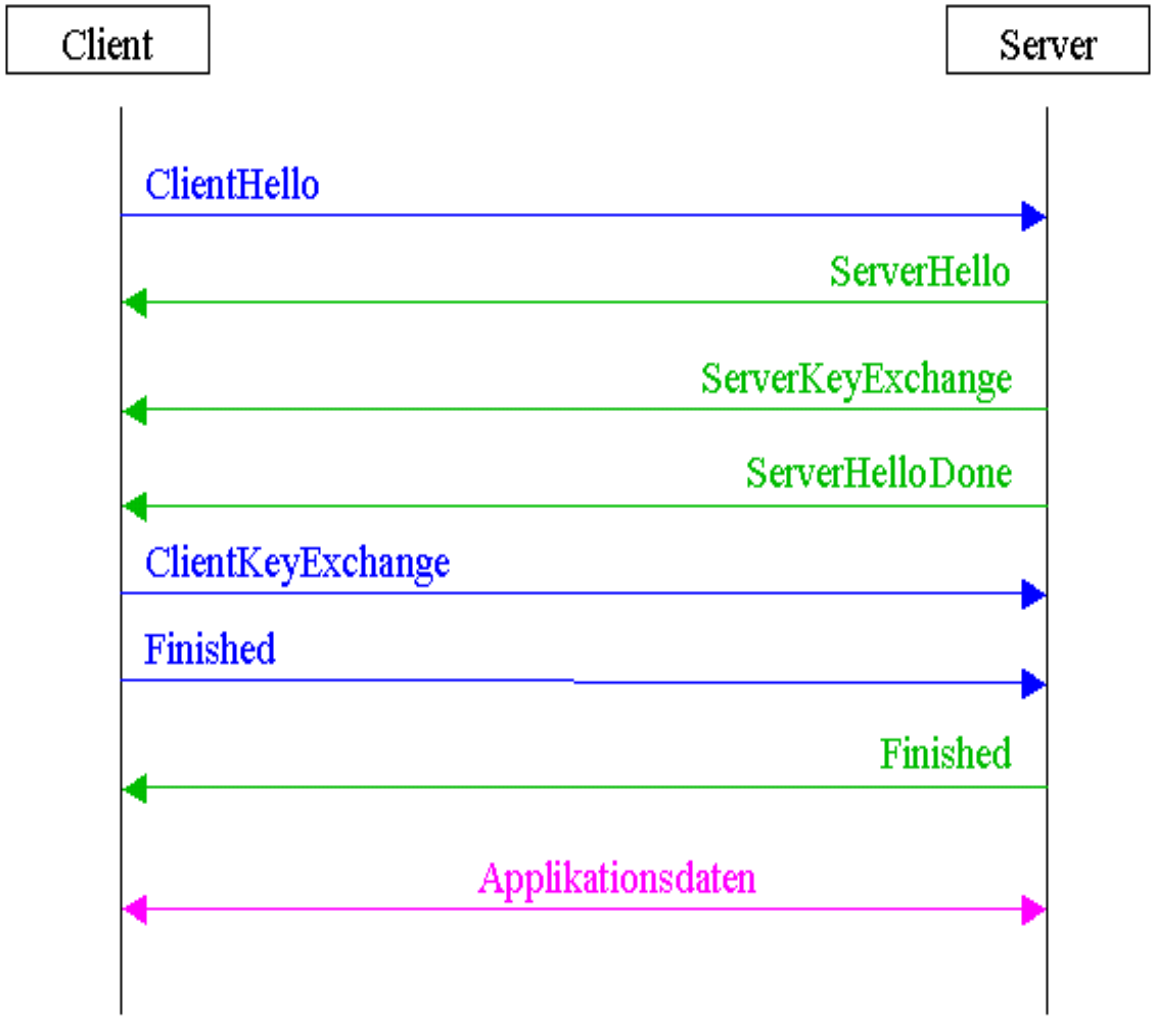
- Einleitung: Definitionen und Ziele
- Protokollaufbau: Header und Nachrichtentypen
- Protokollablauf
- Bedeutung von Zertifikaten in Browsern
- **Authentikationsmethoden**
- Schwachstellen und Angriffsmöglichkeiten
- SSL/TLS in der Praxis
- Zusammenfassung

Authentifikationsmethoden
→ Einführung

- SSL/TLS benutzt Public Key Zertifikate für die Authentifizierung von Server und Client.
- **Es werden drei 3 Verbindungsarten unterschieden:**
  - 1. Server und Client ohne Authentifizierung
  - 2. Server authentifiziert, Client anonym
  - 3. Server und Client authentifiziert

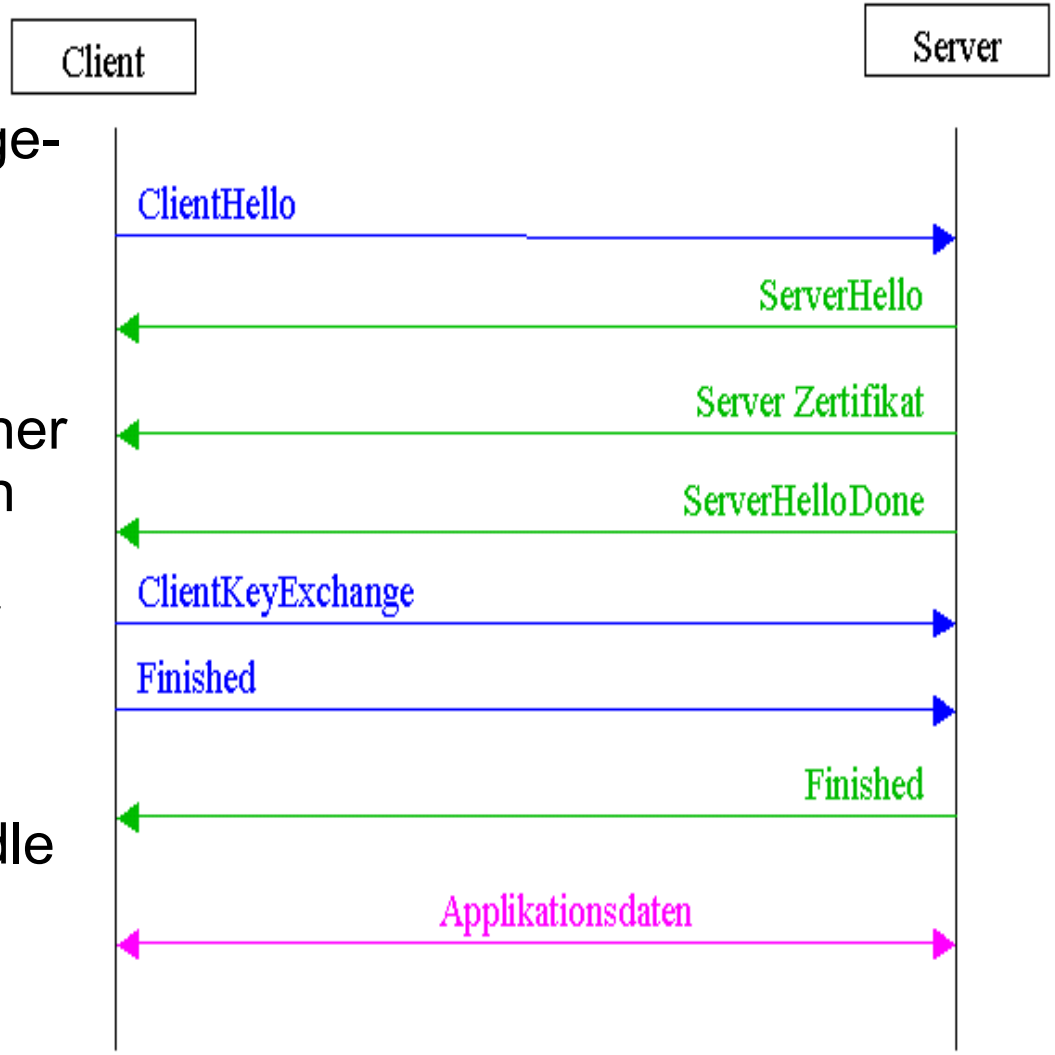
Authentifikationsmethoden
→ Server und Client ohne Authentifizierung

- Bei dieser Verbindungsart verlangt weder der Server noch der Client eine Authentifizierung seines Kommunikationspartners durch ein Zertifikat.
- Nach Beendigung des Verbindungsaufbaus können Applikationsdaten ausgetauscht werden.
- Die beiderseits anonyme Verbindung ist nicht sicher gegenüber aktiven Man-In-The-Middle-Angriffen.
- **Diese Verbindungsart sollte daher vermieden werden!**



Authentifikationsmethoden
→ Server authentifiziert, Client anonym

- Hier teilt der Server seinen öffentlichen Schlüssel dem Client nicht durch eine ServerKeyExchange-Nachricht mit, sondern durch die Übermittlung seines Zertifikates.
- Kann der Client das Zertifikat verifizieren, so kann sich dieser sicher sein, dass nach einem erfolgreichen Verbindungsaufbau eine SSL/TLS-Verbindung zu genau jenem Server zustande gekommen ist, dessen Zertifikat empfangen wurde.
- Hier kann sich ein Man-In-The-Middle in Richtung des Clients nun nicht mehr als falscher Server ausgeben, womit durch die Server-Authentifizierung aktive Angriffe verhindert werden.



Authentifikationsmethoden
→ Server Authentikation im Detail

Ablauf:

1. Client wählt Internetseite an

2. Server übermittelt dem Client sein öffentliches Domänen-Zertifikat mit dem öffentlichen RSA-Schlüssel

3. Client prüft das Domänen-Zertifikat auf Gültigkeit

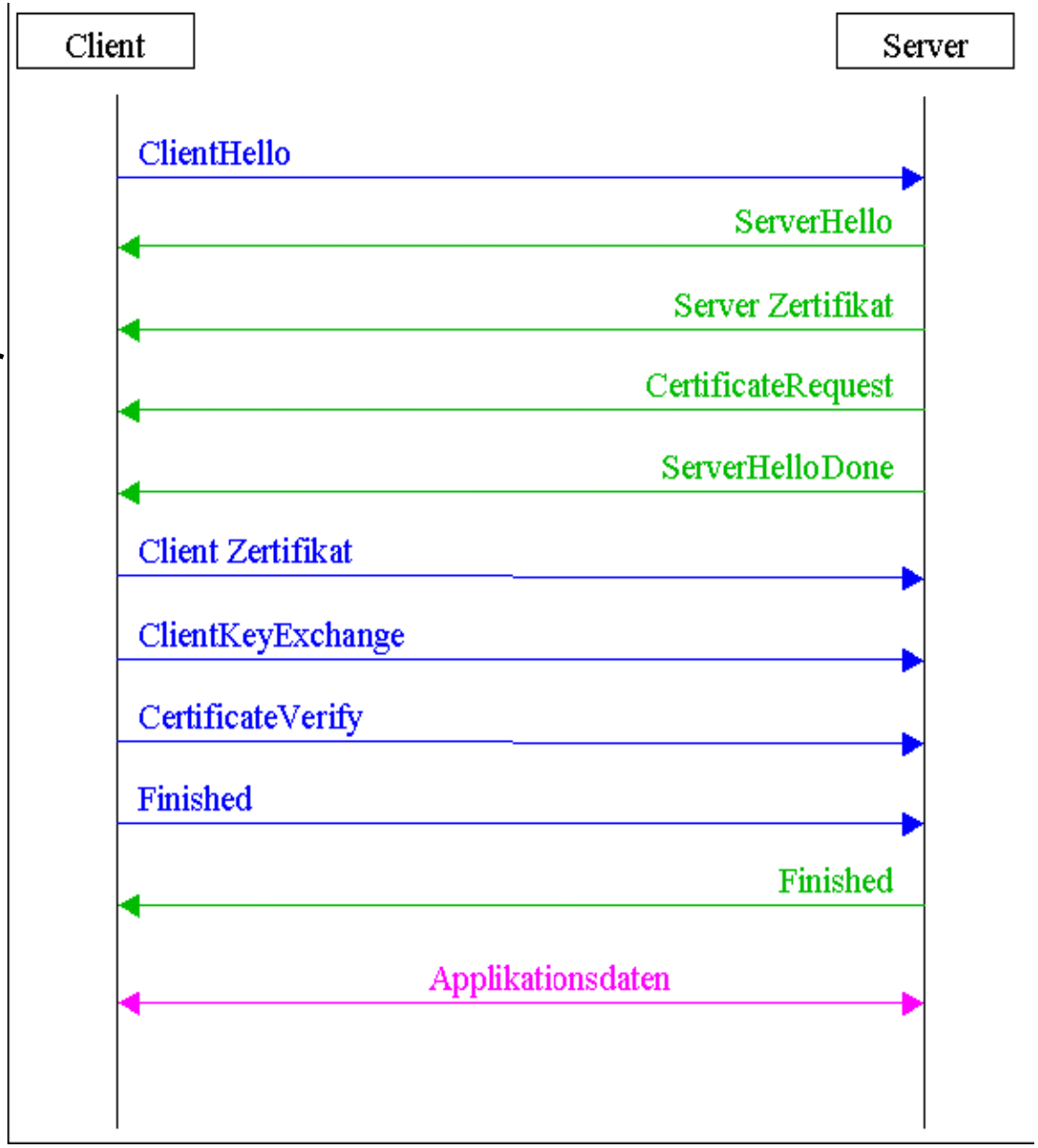
4. Client generiert Zufallszahl (Pre-Master Secret), verschlüsselt diese für den Server mit dem öffentlichen RSA aus dem Domänen-Zertifikat

5. Server entschlüsselt die Zufallszahl (Pre-Master Secret), generiert die Session Keys und verschlüsselt die Daten nun mit dem Sessionkey für den Client.

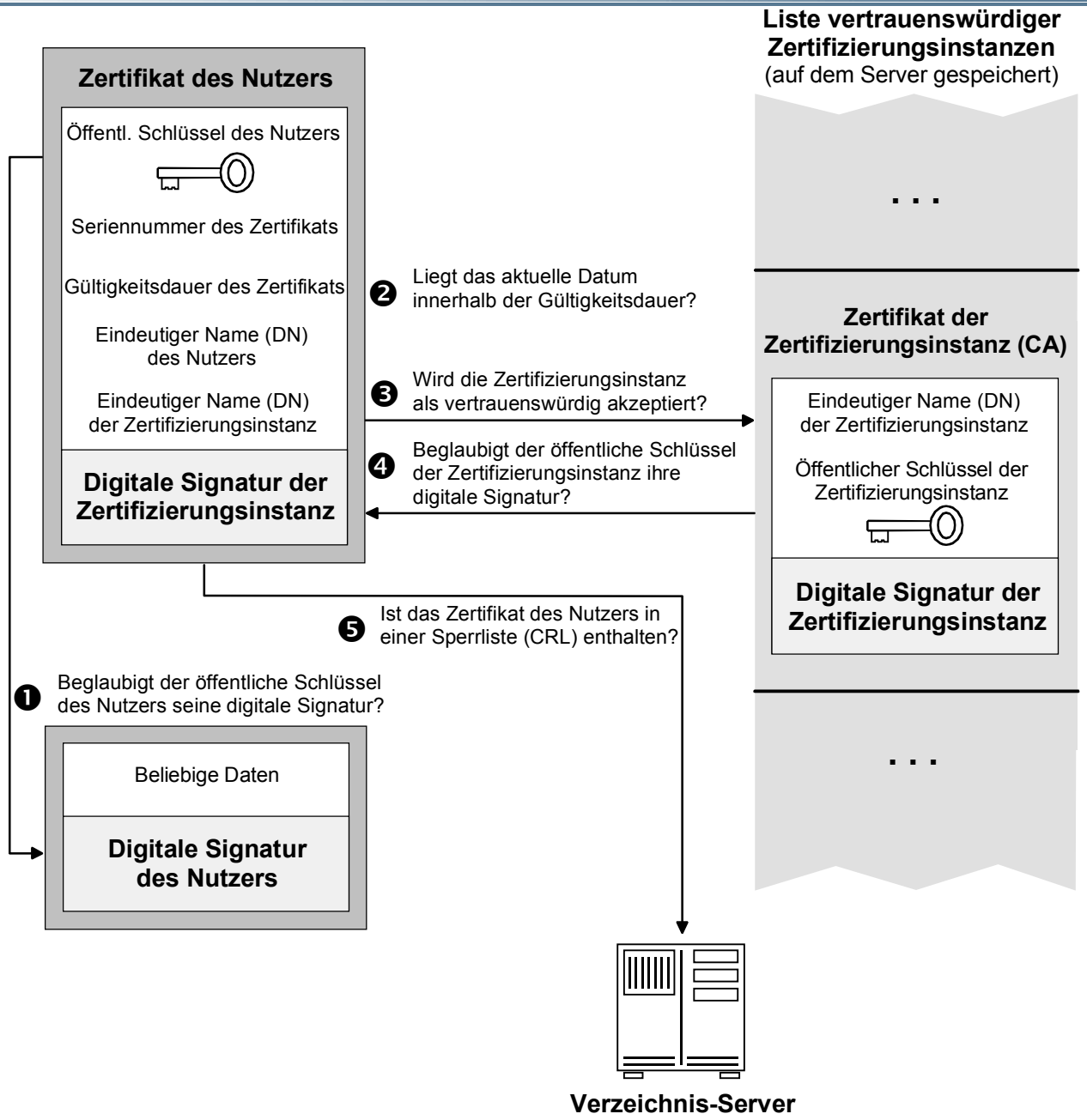
6. Verschlüsselter Datenaustausch zwischen Server und Client, und damit die Gewissheit, dass der Server der echte ist.

Authentifikationsmethoden
→ Server und Client authentifiziert

- Der Server kann mit der CertificateRequest-Nachricht auch eine Client-Authentifikation über ein Zertifikat des Clients beantragen.
- Dieser stellt das Zertifikat mit der ClientCertificate-Nachricht zu und beweist mit der CertificateVerify-Nachricht, dass er auch tatsächlich jener ist, dessen Name im Zertifikat erscheint.
- Falls der Client kein Zertifikat besitzt, welches in der vom Server bekannt gegebenen Liste auftaucht, wird der Verbindungsaufbau abgebrochen.



Überprüfung eines Client-Zertifikats



SSL/TLS-Zertifikate
→ Ablauf der Client-Authentikation im Detail (1/2)

- Der Server überprüft die digital signierten Daten mit dem öffentlichen Schlüssel des Client, der dem Client-Zertifikat entnommen werden kann.
- Verläuft die Überprüfung erfolgreich, ist die Zusammengehörigkeit des öffentlichen und des geheimen Schlüssels des Clients verifiziert.
- Außerdem steht damit fest, dass die signierten Daten nach der Signatur nicht verändert wurden.
- Die Zusammengehörigkeit vom öffentlichen Schlüssel und dem DN (Distinguished Name) des Client-Zertifikats ist damit hingegen nicht bewiesen.
- Der Server überprüft den Gültigkeitszeitraum des gesendeten Client-Zertifikats.

SSL/TLS-Zertifikate
→ Ablauf der Client-Authentikation im Detail (2/2)

- Der Server überprüft anhand des DN (Distinguished Name) und der ihm vorliegenden Liste, ob es sich bei der ausstellenden CA (Certification Authority oder Zertifizierungsinstanz) um eine bekannte CA handelt.
- Das Zertifikat des Client wird dann mittels des öffentlichen Schlüssels der ausstellenden CA überprüft.
- Der öffentliche Schlüssel der CA wird der Liste der bekannten CAs entnommen.
- Dieser optionale Schritt erlaubt es, eine CRL (Certificate Revocation List) einzubinden.
- Hier kann überprüft werden, ob das Zertifikat des Client in der Zwischenzeit gesperrt wurde, obwohl der Gültigkeitszeitraum noch nicht abgelaufen ist.

SSL/TLS-Zertifikate
→ SSL/TLS Authentikation → Bewertung

- **Die Qualität der SSL/TLS-Client-Authentikation hängt ab von:**
  - Der Vertrauenswürdigkeit der Zertifizierungsinstanz
  - Dem Level, auf dem die Zertifizierungsinstanz die Authentizität des Teilnehmers überprüft
  - Der Bereitstellung einer CRL (Certificate Revocation List) durch die Zertifizierungsinstanz

- **Verschiedene Zertifizierungs-Anbieter bieten unterschiedliche Klassen von Zertifikaten an:**
  - Ein Zertifikat der Klasse 1 bekommt der Teilnehmer schon nach der Zusendung des Namens und der E-Mail-Adresse, ohne dass seine Identität näher geprüft wird.
  - Für ein Zertifikat der Klasse 2 wird z.B. die Zusendung einer Kopie des Ausweises oder des Führerscheins verlangt.
  - Bei höheren Klassen muss der Teilnehmer sich persönlich (z.B. mit Hilfe des Ausweises) bei einer Registration Authority (RA) authentisieren.
  - Hierbei stellt sich die Frage, wie vertrauenswürdig eine solche Registration Authority (RA) ist.

Authentifikationsmethoden
→ Welche Methode setzt sich durch?

- Zurzeit wird die ausschließliche Authentifizierung des Servers am häufigsten genutzt.
- Zukünftig wird sicherlich beidseitige Authentifizierung der Standard sein, denn diese Verbindungsart bietet einfach das größte Maß an Vertraulichkeit, Integrität und Sicherheit für eine wirklich sichere Verbindung.

Inhalt

- Einleitung: Definitionen und Ziele
- Protokollaufbau: Header und Nachrichtentypen
- Protokollablauf
- Bedeutung von Zertifikaten in Browsern
- Authentikationsmethoden
- **Schwachstellen und Angriffsmöglichkeiten**
- SSL/TLS in der Praxis
- Zusammenfassung

Schwachstellen und Angriffsmöglichkeiten
→ Schwachstellen (1/2)

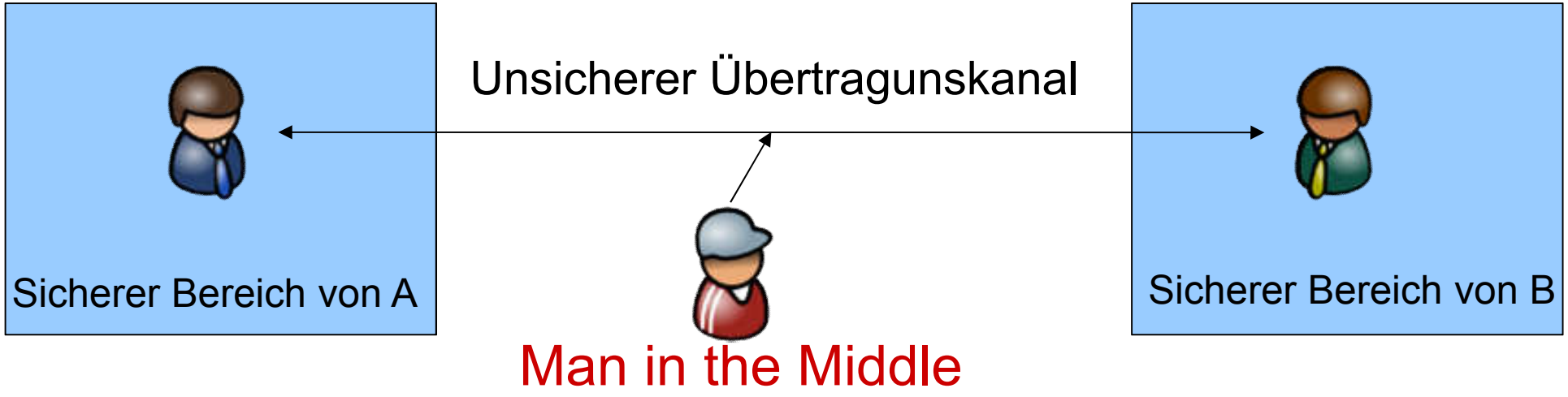
- Die größte Schwäche von SSL/TLS liegt in der Abhängigkeit von Zertifikaten.
- Ohne diese ist das Protokoll verwundbar durch Man-In-The-Middle-Angriffe.
- In der Praxis werden meist Zertifikate nicht überprüft, wodurch die beispielsweise von https gebotene Sicherheit verloren geht.
- Die Haupt-Angriffspunkte des SSL/TLS Protokolls liegen dabei in den Nachrichten des Handshake-Protokolls.

Schwachstellen und Angriffsmöglichkeiten
→ Schwachstellen (2/2)

- Der Record-Layer ist komplett auslesbar, so ist es möglich herausfinden, welches „Unterprotokoll“ gesendet wird.
- Zudem lässt sich erkennen, welche Handshake-Nachrichten gesendet werden.
- Desweiteren erhält jeder die Listen der Cipher-Suites, welche sich Server und Client anbieten.
- Dies sind interessante Informationen, die nicht nur für Angriffe auf den Datenverkehr, sondern auch für statistische Erhebungen genutzt werden können.
- So lässt sich etwa aus dem häufigen Auftreten von Alert-Protokollen schliessen, dass der Client oder eben der Server Probleme hat und evtl. nicht richtig arbeitet.
- Vielleicht deutet dies aber auch auf Angriffsversuche hin.
- Die Listen der Cipher-Suites geben z.B. Auskunft darüber, wie häufig schwache Verschlüsselungen zum Einsatz kommen.

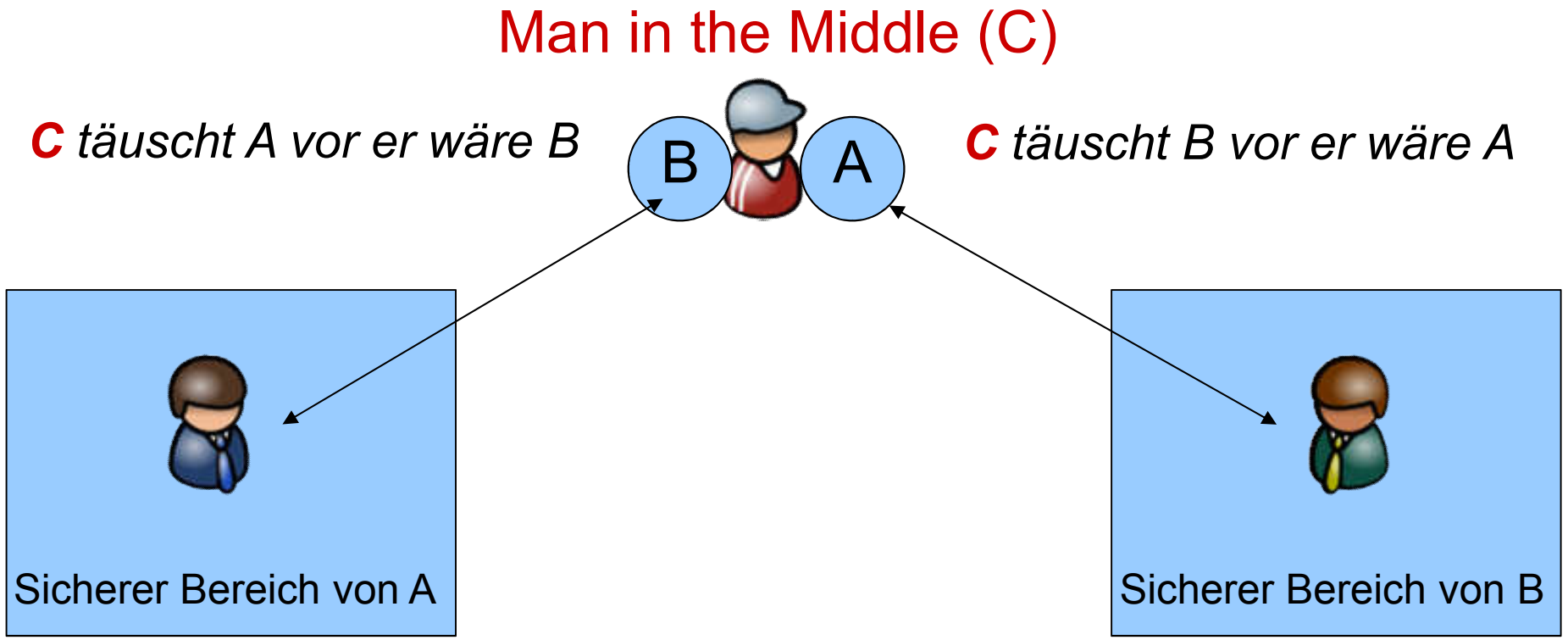
Schwachstellen und Angriffsmöglichkeiten
→ Der Man-In-The-Middle Angriff (1/5)

- Zwei Kommunikationspartner A und B wollen miteinander kommunizieren.
- Beide kennen einander nicht und befinden sich selbst in einer geschützten Umgebung.
- Der Kommunikationskanal zwischen A und B ist jedoch öffentlich und kann von jedem abgehört werden.
- So kann etwa C die Kommunikation zwischen A und B belauschen, sogar verändern und anschließend an den Bestimmungsort weiterleiten, ohne dass der Empfänger dies bemerkt.



Schwachstellen und Angriffsmöglichkeiten
→ Der Man-In-The-Middle Angriff (2/5)

- Verändert der **Man in the Middle** Nachrichten, so wird von aktiven ansonsten von passiven Angriffen gesprochen.
- A und B wollen sich nun auf eine vertrauliche Verschlüsselung einigen, die von C nicht gelesen werden kann.
- Dies ist nach der dargestellten Situation unmöglich.

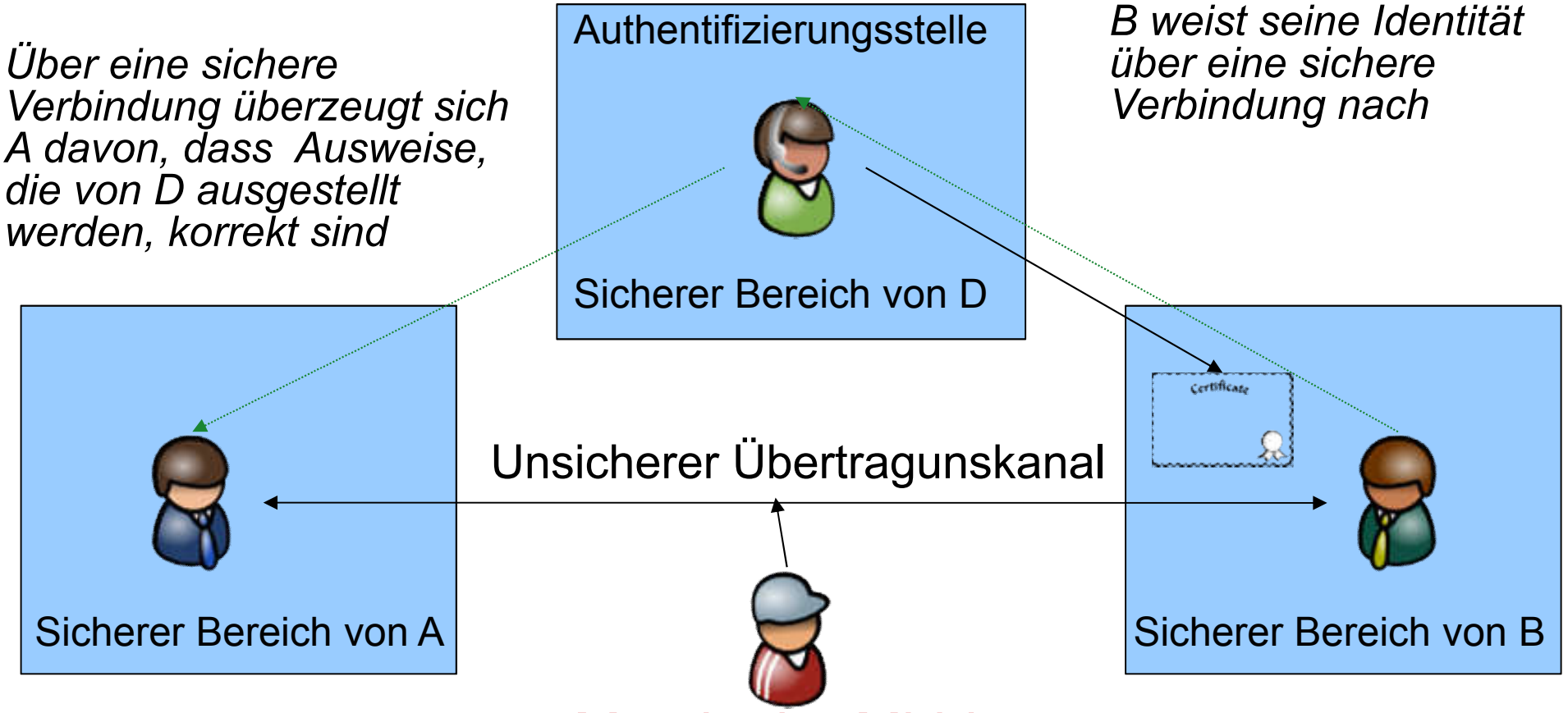


Schwachstellen und Angriffsmöglichkeiten
→ Der Man-In-The-Middle Angriff (3/5)

- C nimmt nun Nachrichten von A entgegen, bearbeitet sie und leitet sie an B weiter bzw. umgekehrt.
- Wenn A eine verschlüsselte Verbindung mit B aufbaut, so tut A dies in Wirklichkeit mit C.
- C richtet in Folge eine sichere Verbindung zu B ein, als wäre dies der Wunsch von A.
- Selbst wenn A nun eine verschlüsselte Nachricht an B schickt, kann C diese entschlüsseln, einsehen und anschließend wieder für B verschlüsseln, usw.
- Das Problem lässt sich durch die Nutzung von Zertifikaten lösen, mit dem sich B gegenüber A eindeutig ausweisen kann.
- Mit dem Zertifikat darf sich natürlich niemand anderes als B selbst ausweisen können, zumindest muss A die Möglichkeit haben zu prüfen, ob das Zertifikat wirklich von B selber stammt.

Schwachstellen und Angriffsmöglichkeiten
→ Der Man-In-The-Middle Angriff (4/5)

- B darf sich dieses Zertifikat nicht selbst ausstellen können, denn sonst könnte sich auch C ein Zertifikat anfertigen, in dem sich C als B ausweist.
- Die folgende Abb. zeigt, dass eine 3. vertrauenswürdige Person D benötigt wird:



Schwachstellen und Angriffsmöglichkeiten
→ Der Man-In-The-Middle Angriff (5/5)

- A muss sich einmalig davon überzeugen, dass alle Zertifikate die von D ausgestellt werden, korrekt sind.
- Bevor die Authentifizierungsstelle ein Zertifikat ausstellt, muss der Antragsteller seine Identität eindeutig nachweisen.
- Dies muss oder soll nicht auf elektronischem Wege geschehen. Idealerweise muss sich B physisch an den Aufenthaltsort von D begeben und dort sein Zertifikat beantragen.
- Die in der Abb. dargestellte Situation zeigt die Rahmenbedingungen, in der SSL/TLS nun eine sichere Verbindung zwischen A und B gewährleisten kann.

Schwachstellen und Angriffsmöglichkeiten
→ Weitere Angriffsarten

- **Weitere Angriffsarten sind:**
  - Frame Spoofing
  - Web Spoofing
  - Bleichenbacher-Angriff
  - Falsche Zertifikate
  - Ciphersuite-Rollback-Angriff
  - Key-Exchange-Version-Rollback
  - Version-Rollback-Angriff
  - Million-Questions-Angriff

Schwachstellen und Angriffsmöglichkeiten
→ Schutzmaßnahmen

- Stets die neueste SSL/TLS Version nutzen soweit möglich
- Auf optische Täuschungen achten bzw. zugesandte Links prüfen
  Bsp.: Original: www.nordpol.de Täuschung: www.n0rdpol.de
- Signierte Zertifikate nutzen und evtl. Zertifikate genauer betrachten bevor man sie akzeptiert
- Immer die neuesten Browserversionen benutzen
- Wenn möglich starke Verschlüsselungen nutzen:

Server: Schwache Verschlüsselungen, die vom Client offeriert werden, zurückweisen und Verbindung zum Client nicht aufbauen.

Client: Nur starke Verschlüsselungen im Browser anbieten.
In den Browsern kann die Cipher Suite ausgewählt werden.
Hier sollten nur sichere Suite angeboten werden.

Inhalt

- Einleitung: Definitionen und Ziele
- Protokollaufbau: Header und Nachrichtentypen
- Protokollablauf
- Bedeutung von Zertifikaten in Browsern
- Authentikationsmethoden
- Schwachstellen und Angriffsmöglichkeiten
- **SSL/TLS in der Praxis**
- Zusammenfassung

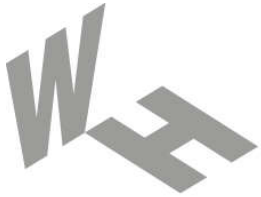
SSL/TLS in der Praxis
→ Grundlagen

- Einsatzgebiete: Loginbereiche, Onlinebanking, Onlineshops
- viele Webpräsenzen arbeiten noch mit veralteten SSL/TLS Versionen
- viele Webseiten arbeiten mit zu schwachen Verschlüsselungen bzw. erlauben diese.
- Dies sollte eigentlich von einer Webseite zurückgewiesen werden.
- grade Onlinebanking-Seiten weisen hier gravierende Mängel auf
- großer Nachholbedarf im Umgang mit SSL/TLS vorhanden

SSL/TLS in der Praxis
→ Vorteile / Nachteile

- **Vorteile:**
  - Der Vorteil des SSL/TLS-Protokolls ist die Möglickiet, jedes höhere Protokoll auf der Basis des SSL/TLS-Protokolls zu implementieren.
  - Damit ist eine Unabhängigkeit von Anwendungen und Systemen gewährleistet.

- **Nachteil:**
  - Der Nachteil der SSL/TLS-verschlüsselten Übertragung besteht darin, dass der Verbindungsaufbau auf Serverseite sehr rechenintensiv und deshalb etwas langsamer ist.



Westfälische Hochschule
Gelsenkirchen Bocholt Recklinghausen
University of Applied Sciences

SSL/TLS - Protokollmitschnitte

Prof. Dr. (TU NN)
Norbert Pohlmann

Institut für Internet-Sicherheit – if(is)
Westfälische Hochschule, Gelsenkirchen
http://www.internet-sicherheit.de

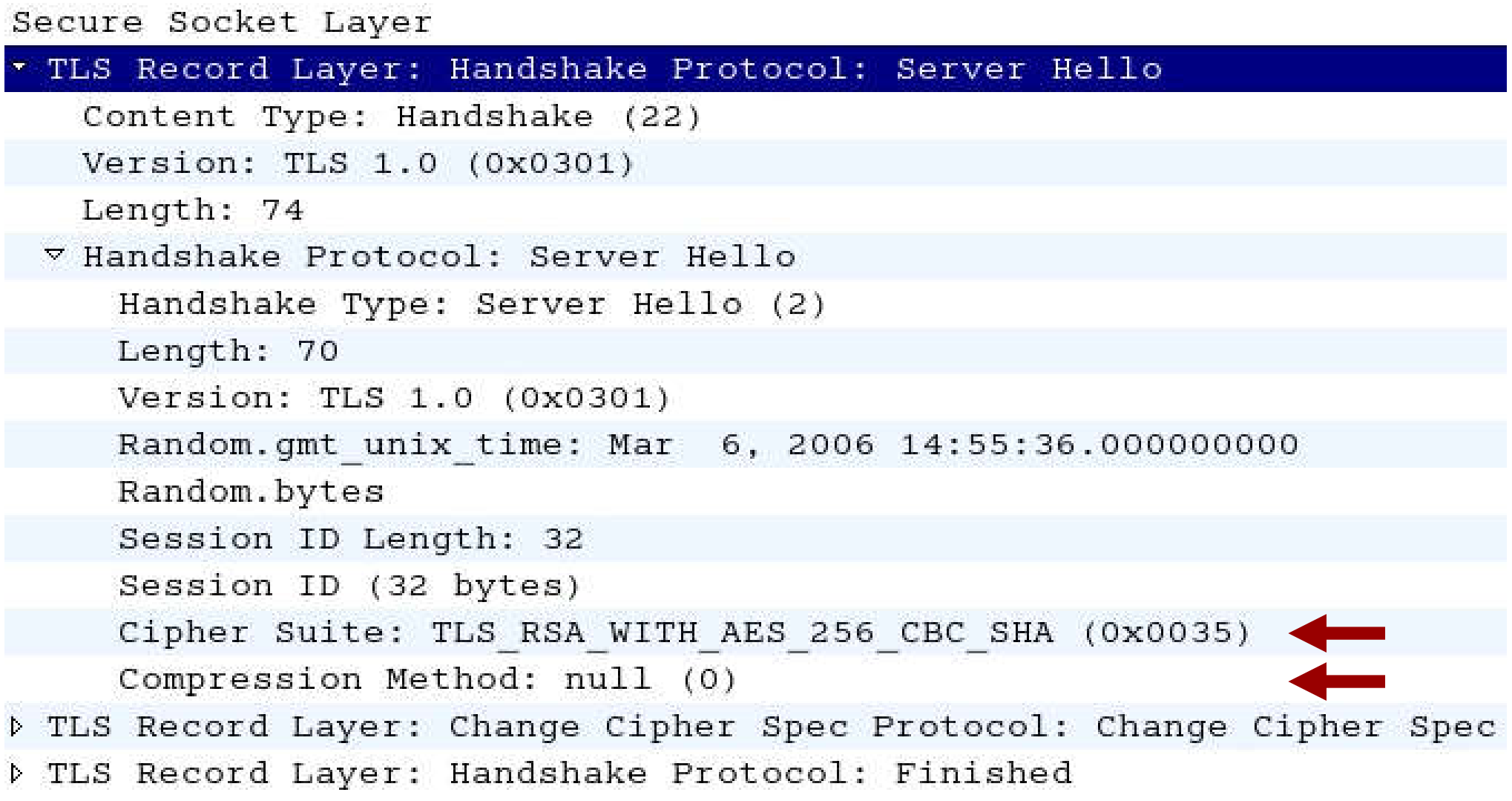
SSL/TLS Mitschnitte
→ ClientHello

```
Secure Socket Layer
▾ TLS Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 115
  ▾ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 111
      Version: TLS 1.0 (0x0301)
      Random.gmt_unix_time: Feb 10, 1970 09:35:21.000000000
      Random.bytes
      Session ID Length: 32
      Session ID (32 bytes)
      Cipher Suites Length: 40
    ▸ Cipher Suites (20 suites)
      Compression Methods Length: 1
    ▸ Compression Methods (1 method)
```

SSL/TLS Mitschnitte
→ CipherSpec Liste aus SSLv2.0 ClientHello

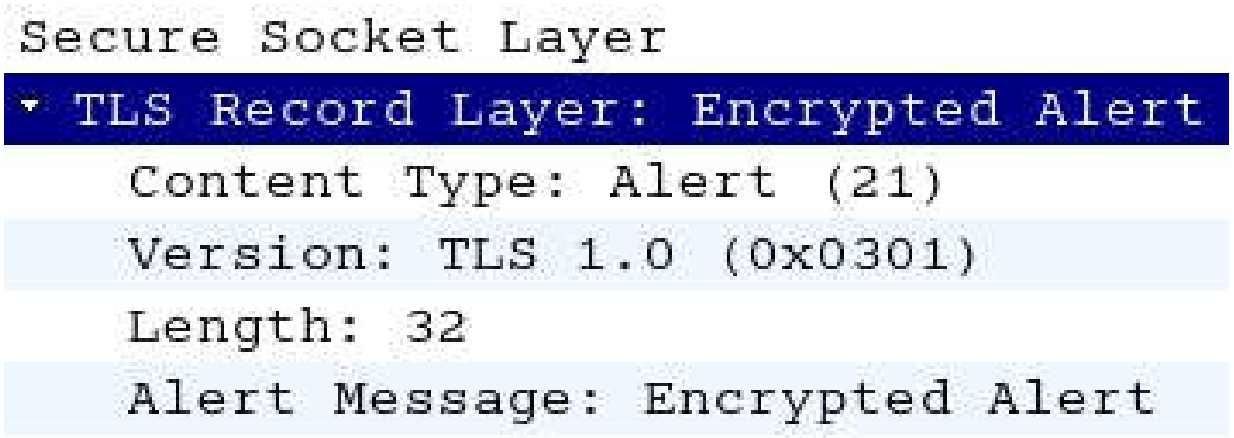
```
Cipher Specs (26 specs)
  Cipher Spec: SSL2_RC4_128_WITH_MD5 (0x010080)
  Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x030080)
  Cipher Spec: SSL2_DES_192_EDE3_CBC_WITH_MD5 (0x0700c0)
  Cipher Spec: SSL2_DES_64_CBC_WITH_MD5 (0x060040)
  Cipher Spec: SSL2_RC4_128_EXPORT40_WITH_MD5 (0x020080)
  Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x040080)
  Cipher Spec: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x000039)
  Cipher Spec: TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x000038)
  Cipher Spec: TLS_RSA_WITH_AES_256_CBC_SHA (0x000035)
  Cipher Spec: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x000033)
  Cipher Spec: TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x000032)
  Cipher Spec: TLS_RSA_WITH_RC4_128_MD5 (0x000004)
  Cipher Spec: TLS_RSA_WITH_RC4_128_SHA (0x000005)
  Cipher Spec: TLS_RSA_WITH_AES_128_CBC_SHA (0x00002f)
  Cipher Spec: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x000016)
  Cipher Spec: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x000013)
  Cipher Spec: SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (0x00feff)
  Cipher Spec: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x00000a)
  Cipher Spec: TLS_DHE_RSA_WITH_DES_CBC_SHA (0x000015)
```

SSL/TLS Mitschnitte
→ ServerHello

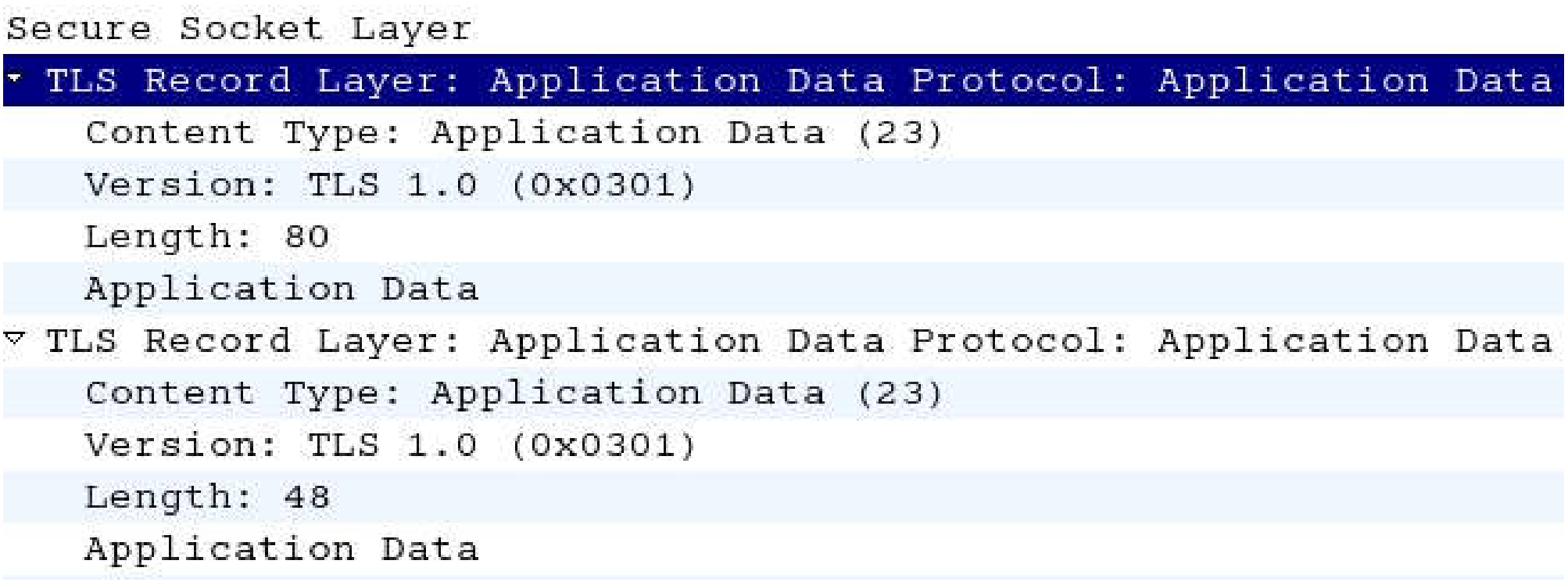


SSL/TLS Mitschnitte
→ ClientKeyExchange, ChangeCipherSpec

SSL/TLS Mitschnitte
→ Alert Nachricht



SSL/TLS Mitschnitte
→ Application Data



Inhalt

- Einleitung: Definitionen und Ziele
- Protokollaufbau: Header und Nachrichtentypen
- Protokollablauf
- Bedeutung von Zertifikaten in Browsern
- Authentikationsmethoden
- Schwachstellen und Angriffsmöglichkeiten
- SSL/TLS in der Praxis
- **Zusammenfassung**

Zusammenfassung (1/3)

- **SSL (Secure Socket Layer) / TLS (Transport Layer Security)** ist ein Sicherheitsprotokoll.
- SSL ist **applikationsunabhänig** und setzt logisch auf einem Transportprotokoll auf.
- SSL/TLS bündelt 3 Sicherheitsdienste:
  - **Authentifizierung**
  - **Verschlüsselung**
  - **Integritätsüberprüfung**

- **TLS (Transport Layer Security)** 1.0, 1.1, 1.2 stellen die standardisierte Weiterentwicklung von SSL 3.0 dar.
- Gängigste Protokolle die SSL/TLS nutzen: https, ftps, imaps, pop3s, smtps, telnets, sips, ...
- Die eigentlichen Protokolldaten werden anstatt im eigenen Anwendungsprotokoll über das **Application Data- / Record Layer-Protokoll** von SSL/TLS verschlüsselt und integritätsgesichert übertragen (z.B. HTTP <-> HTTPS).

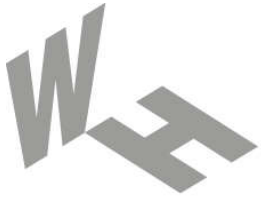
Zusammenfassung (2/3)

- Anhand der **Portnummer** lässt sich erkennen, um welches ursprüngliche Protokoll es sich bei den übertragenen Daten handelt.
- SSL/TLS besteht aus **2 Schichten**, nämlich dem Record Layer-Protokoll und den vier SSL/TLS-Teilprotokollen:
  - ChangeCipherSpec
  - Alert
  - Handshake
  - Application Data
- Der Protokollablauf bei SSL/TLS erfolgt in zwei Schritten:
  1. Schritt:
  Verbindungsaufbau, unterteilt in 4 Phasen:
  - 1. Phase: Aushandlung der Sicherheitsparameter.
  - 2. Phase: Serverauthentisierung (Optional) und Schlüsselaustausch
  - 3. Phase: Clientauthentisierung (Optional) und Schlüsselaustausch
  - 4. Phase: Beendigung des Handshakes

  2. Schritt:
  Verschlüsselte und integritätsgesicherte Datenübertragung

Zusammenfassung (3/3)

- SSL/TLS beinhaltet 2 wichtige Instanzenkonzepte:
  - SSL/TLS-Session
  - SSL/TLS-Connection
- **Cipher Suites** sind eine Kombination aus **Schlüsselaustauschverfahren**, **Verschlüsselungsverfahren** mit **Schlüssellänge** und ein **Verfahren zum Integritätscheck**.
- Aufgabe eines Domänen-Zertifikats:
  Eindeutige Zuordnung eines Public-Key's zu einer Organisation.
- SSL/TLS bietet 3 Verbindungsarten:
  1. Server und Client ohne Authentifizierung
  2. Server authentifiziert, Client anonym
  3. Server und Client authentifiziert
- großer Nachholbedarf bzgl. des richtigen Umgangs mit SSL/TLS vorhanden



Westfälische Hochschule
Gelsenkirchen Bocholt Recklinghausen
University of Applied Sciences

Secure Socket Layer (SSL)
Transport Layer Security (TLS)

Vielen Dank für Ihre Aufmerksamkeit
Fragen ?

Prof. Dr. (TU NN)
Norbert Pohlmann

Institut für Internet-Sicherheit – if(is)
Westfälische Hochschule, Gelsenkirchen
http://www.internet-sicherheit.de