SSL/TLS und SSL-Zertifikate

Konzepte von Betriebssystem-Komponenten

Informatik Lehrstuhl 4

Motivation

- Sichere, verschlüsselte End-to-End Verbindung
- Möglichkeit zur Authentifizierung einer oder beider Seiten
- Robust gegen Man-in-the-Middle Angriffe

Überblick

- Motivation
- Public Key Infrastructure
  - Certificate Authorities
  - Root Certificates
  - Hierarchie
- SSL-Zertifikate
  - Aufbau
  - Signieren und Validieren
  - Vor- und Nachteile von Zertifikatshierarchien
- SSL/TLS
  - Konzept und Entwicklung
  - Der SSL Handshake

Überblick

- Motivation
- Public Key Infrastructure
  - Certificate Authorities
  - Root Certificates
  - Hierarchie
- SSL-Zertifikate
  - Aufbau
  - Signieren und Validieren
  - Vor- und Nachteile von Zertifikatshierarchien
- SSL/TLS
  - Konzept und Entwicklung
  - Der SSL Handshake

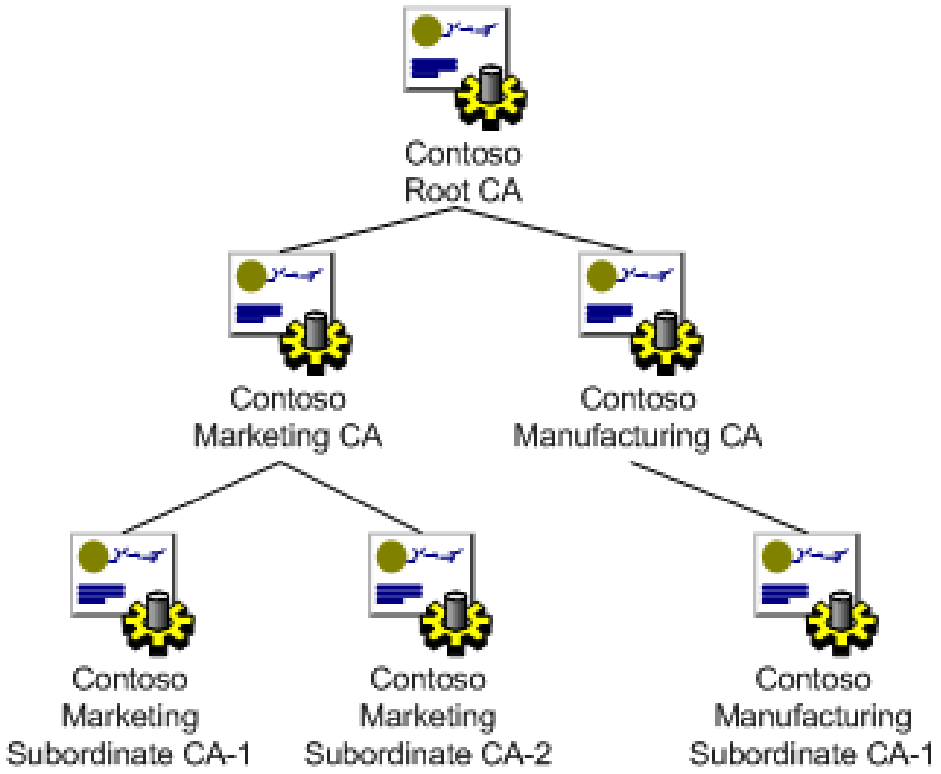
PKI: Certificate Authorities

- Verantwortlich für das Verifizieren von Antragsstellern und Signieren aller Zertifikate
- Signatur besagt, dass die CA sichergestellt hat, die Person bzw. Firma ist auch wirklich diejenige, die sie behauptet zu sein.
- Geht meistens gut - aber nicht immer
  - VeriSign stellt 2001 zwei Microsoft Zertifikate aus
  - Antragsteller war aber nicht von Microsoft...

PKI: Root Certificates

- Zertifikate, die nur von sich selbst signiert sind
- Werden generell mit Software (z.B. Browser) ausgeliefert
- Diese Softwareentwickler müssen sehr vorsichtig sein, welche *Root Certificates* sie mit aufnehmen!
- Sehr hohe Einstiegsschwelle für neue Certificate Authorities

Zertifikationshierarchie



Certificate Hierarchy[2]

- Root CA:
  - Besitzt *Root Certificate*
  - Benötigt für Validierung aller Contonso Zertifikate
- Marketing CA:
  - Benötigt für die Validierung von Marketing CA-1 und CA-2 Zertifikaten und deren Unterzertifikate
- Manufacturing CA:
  - Nötig für Validierung eigener Unterzertifikate

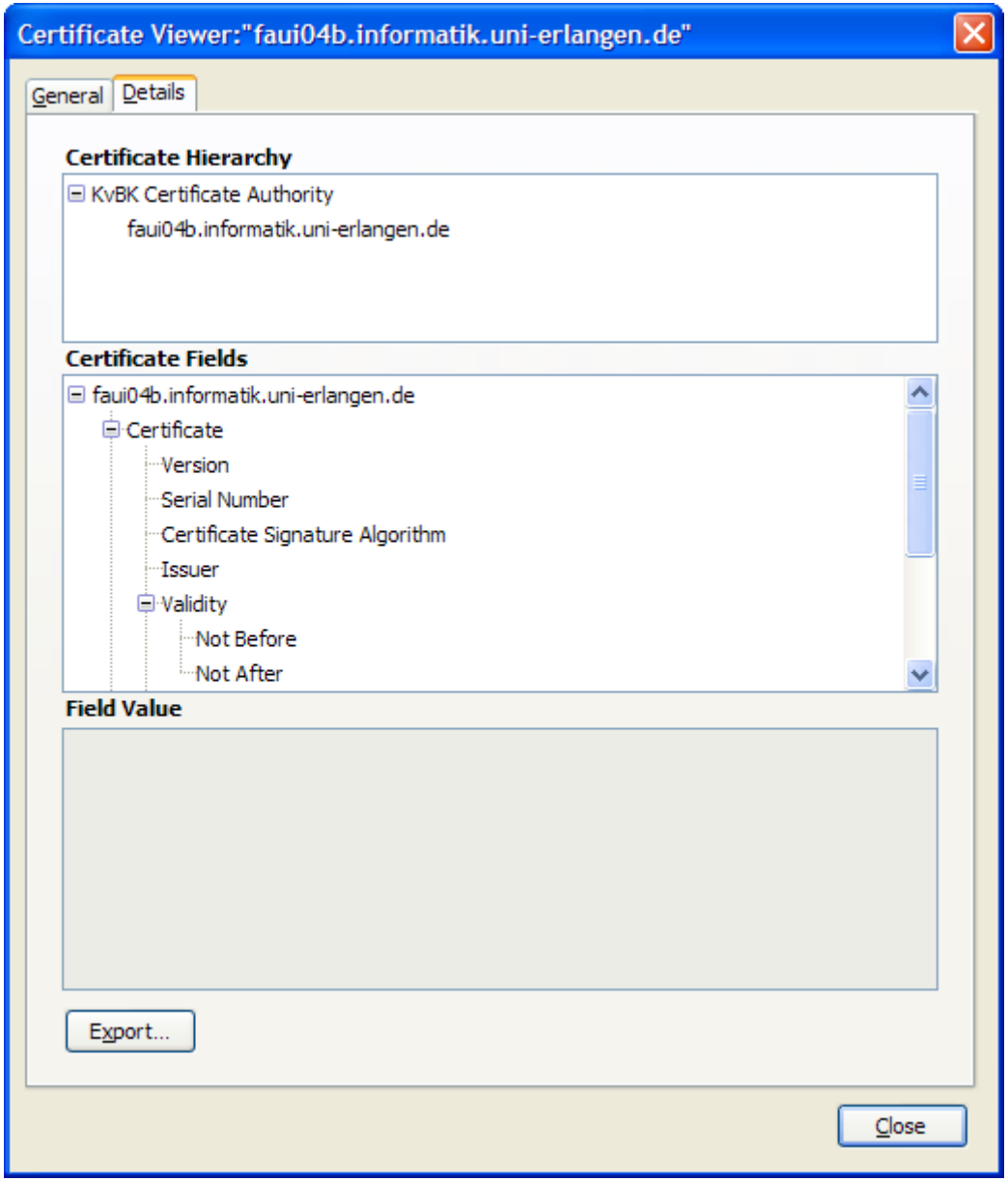
Überblick

- Motivation
- Public Key Infrastructure
  - Certificate Authorities
  - Root Certificates
  - Hierarchie
- SSL-Zertifikate
  - Aufbau
  - Signieren und Validieren
  - Vor- und Nachteile von Zertifikatshierarchien
- SSL/TLS
  - Konzept und Entwicklung
  - Der SSL Handshake

SSL-Zertifikate

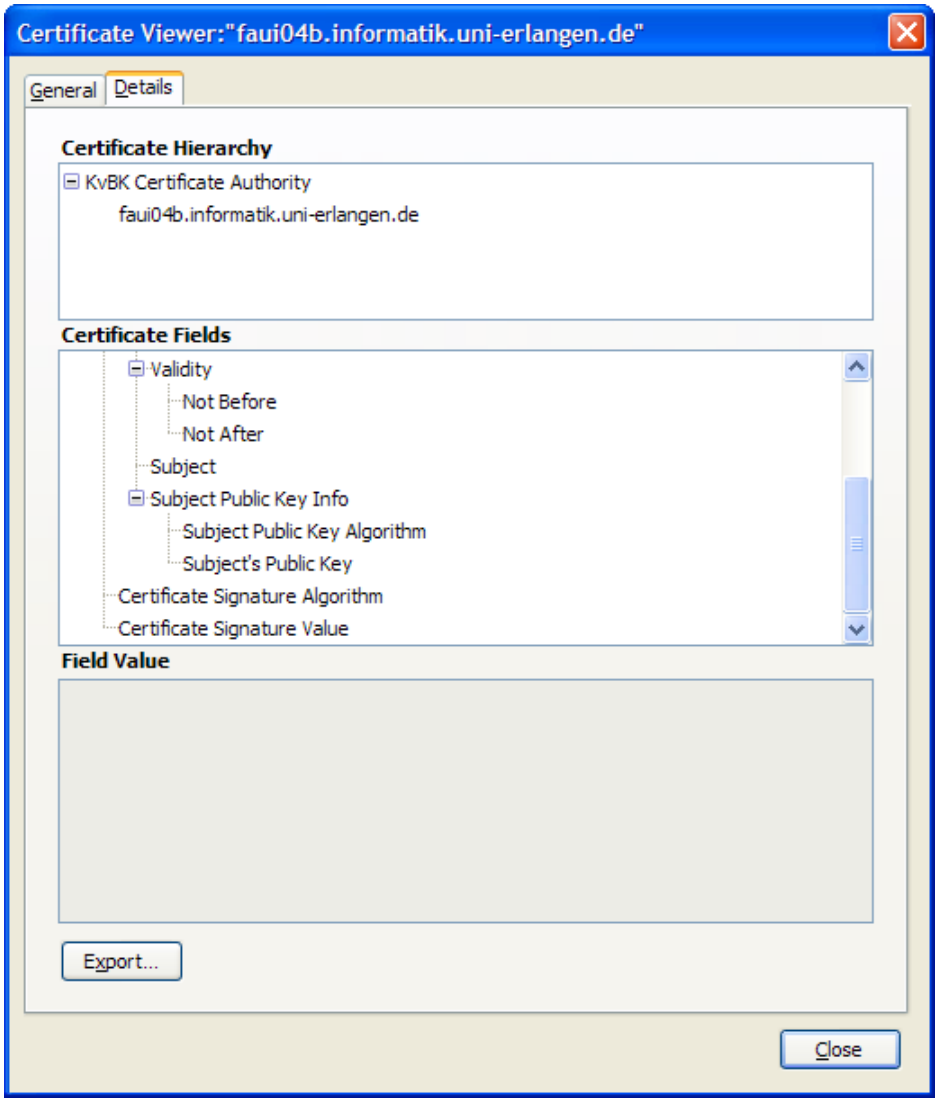
- Verwendet für Authentifizierung des Partners
- Garantiert nur, dass Nachrichten von diesem Partner kommen und nicht unterwegs verändert werden - nicht mehr!
  - Amazon.de → Amazone.de
  - Können beide SSL-Zertifikate haben - nur nicht das selbe
- Authentität von Zertifikaten sichergestellt durch Signatur von *Certificate Authority*
- Alle Zertifikate hängen von *Root Certificates* ab

Zertifikatsaufbau



- Standard: X.509
  - Signatur-Algorithmus der CA
  - Issuer: Signierende CA
  - Validity: Ausstellungsdatum und Gültigkeitsdauer

Zertifikatsaufbau



- Standard: X.509
  - Subject: Gültigkeitsbereich des Zertifikats
  - Subject Public Key: Algorithmus und *public key* des Zertifikatsinhabers
  - Signatur-Algorithmus der CA (Wiederholt)
  - Signatur des Zertifikats

Signieren von Zertifikaten

- Vorraussetzungen:
  - Bestehendes Zertifikat für die CA (evtl. *Root Certificate*)
  - Private key der CA
  - CSR (*Certificate Signing Request*) des Antragsstellers
  - Identität des Antragsstellers bestätigt

Signieren von Zertifikaten

- Signiervorgang:
  - Hash- und Verschlüsselungsalgorithmus wählen
  - In Certificate Signature Algorithm eintragen
  - Bilden eines Hashes über den Rest des Zertifikats
  - Signieren des Hashes mit dem *private key* der CA

Validieren von Zertifikaten

- Vorraussetzungen:
  - Zu validierendes Zertifikat
  - Alle Zertifikate zwischen diesem und einem *Root Certificate*
  - Bereits installiertes *Root Certificate*

Validieren von Zertifikaten

- Validierungsvorgang:
  - Entschlüsselung der Signatur mit angegebenem Algorithmus und *public key* des darüberliegendem Zertifikats
  - Bilden des Hashes über den Rest des Zertifikats und vergleichen mit dem Hash in der Signatur
  - Überprüfen der Gültigkeitsdauer und Subjekt des Zertifikats

Vor- und Nachteile

- Vorteile:
  - Relative starke Kontrolle über Zertifikatsvergabe
  - Sehr gut skalierbar
  - Für Endbenutzer sehr transparent und unaufwendig
- Nachteile:
  - Hoher Aufwand zu Beginn
  - Zertifikatsnachschub vollkommen von CAs abhängig
  - Zertifikatswiderruf möglich, aber umständlich
    - Mehrere Ansätze, aber es hat sich noch keine Methode durchgesetzt

Überblick

- Motivation
- Public Key Infrastructure
  - Certificate Authorities
  - Root Certificates
  - Hierarchie
- SSL-Zertifikate
  - Aufbau
  - Signieren und Validieren
  - Vor- und Nachteile von Zertifikatshierarchien
- SSL/TLS
  - Konzept und Entwicklung
  - Der SSL Handshake

Konzept von SSL/TLS

- Sitzt zwischen Application Layer und Transport Layer
- Verwendung hauptsächlich mit TCP
- Unterstützt eine Vielzahl an Verschlüsselungs- und Komprimierungsmethoden
- Verwendet sowohl symmetrische als auch asymmetrische Verschlüsselung

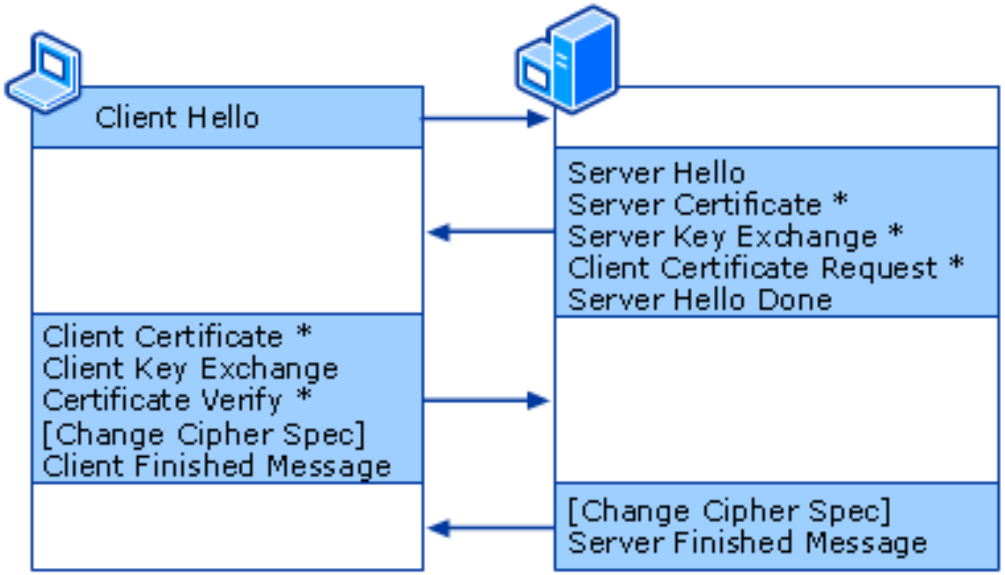
Entwicklung von TLS

- SSL (Secure Socket Layer) in den 90er Jahren von Netscape entwickelt
- Version 1 nie öffentlich herausgegeben
- Version 2 in 1995 fertiggestellt, aber unsicher
  - Veschlüsselung unterlag in den US dem Munitionsgesetz
  - Zwei Editionen: “US” Edition hatte volle 128-bit Verschlüsselung, “International” Edition nur effektiv 40-bit
  - US Edition so umständlich zu erhalten, dass sich kaum jemand die Mühe machte

Entwicklung von TLS

- 1996 wurde Verschlüsselungstechnologie von der Munitionsliste genommen
  - Im selben Jahr gibt Netscape SSL Version 3 heraus
- IETF baut auf SSL v3 auf und definiert 1999 TLS (Transport Layer Security) als inoffiziellen Nachfolger von SSL
  - SSL und TLS sind unterschiedlich genug um nicht kompatibel zu sein, aber sind zu großen Teilen identisch
  - TLS im Protokoll benannt als “SSL 3.1”

SSL Handshake

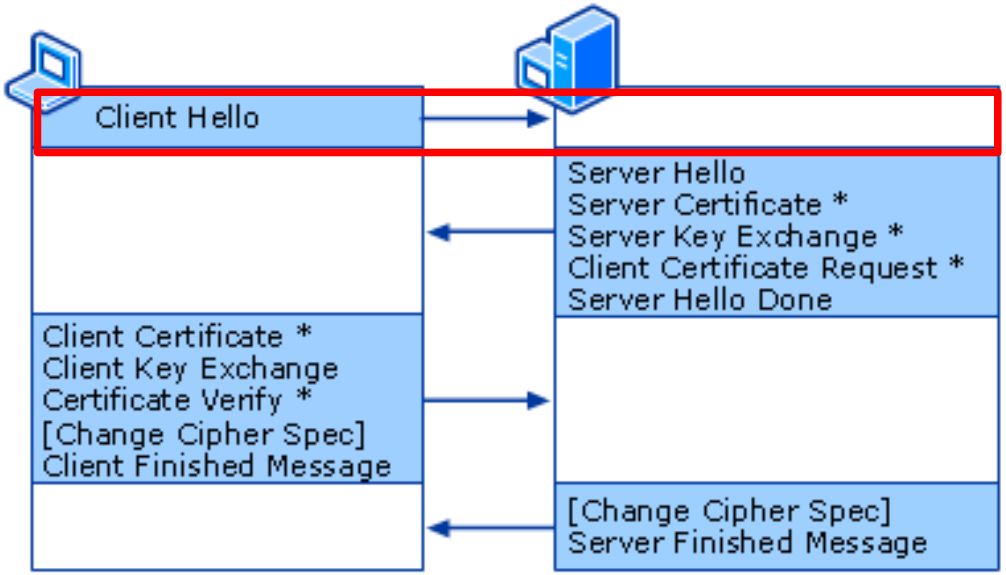


SSL Handshake Protocol[1]

1) Einigung auf Verschlüsselungs- und Kompressions-Algorithmus
2) Erstellen eines gemeinsamen Geheimnises
3) Austausch von Authentifizierungs-Informationen
4) Beginn der Datenübertragung

* Optional

SSL Handshake: ClientHello

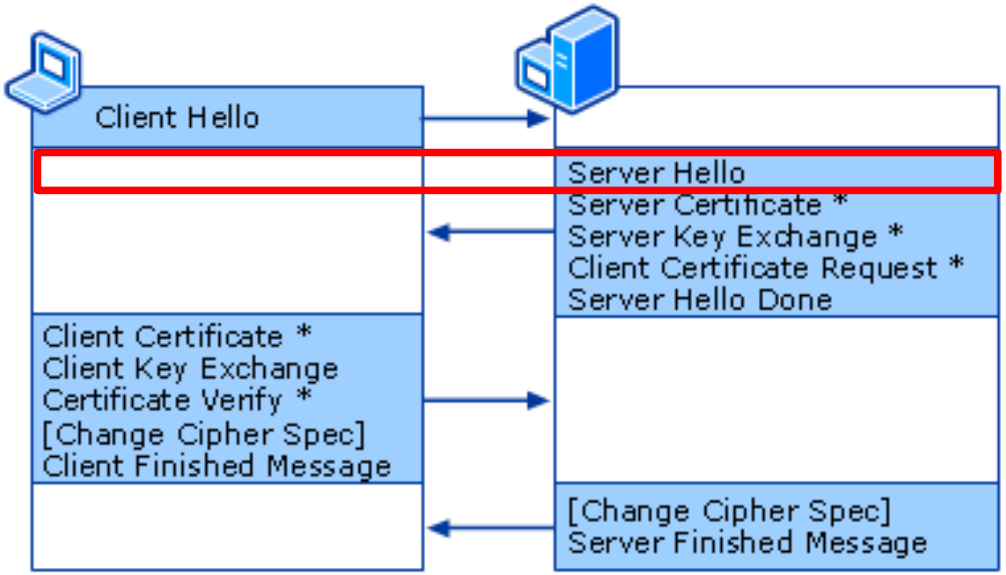


SSL Handshake Protocol

- Version
- Zufallsnummer
- Unterstützte Verschlüsselungs-Algorithmen
  - Key Exchange, Cypher, Hash
- Unterstützte Kompressions-Algorithmen

* Optional

SSL Handshake: ServerHello

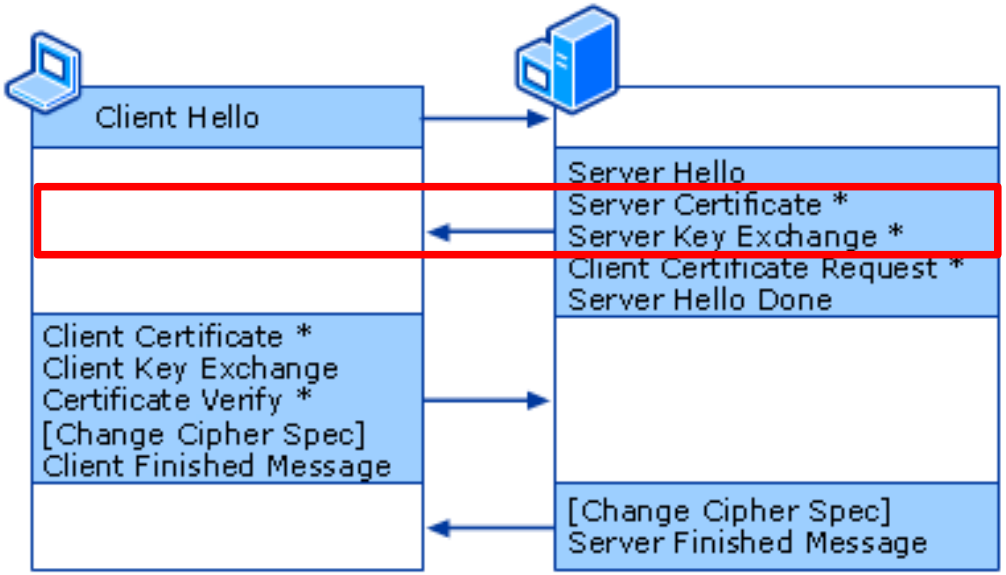


SSL Handshake Protocol

- Verwendete Version
- Zufallsnummer
- Verwendeter Verschlüsselungs-Algorithmen
  - Key Exchange, Cypher, Hash
- Verwendeter Kompressions-Algorithmen

* Optional

SSL Handshake: Server Certificate

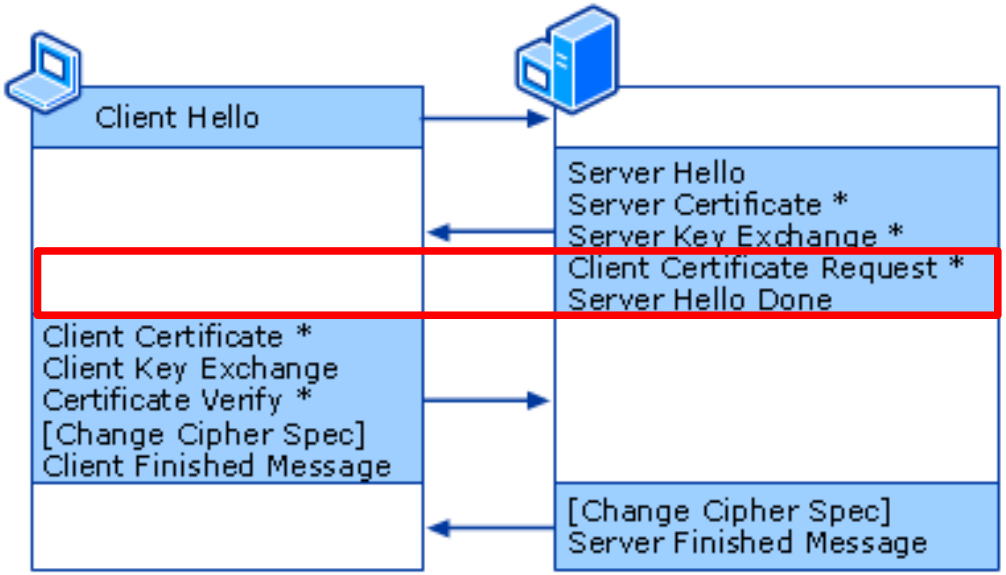


SSL Handshake Protocol

* Optional

- Falls Authentifizierung verwendet wird:
  - Server Certificate mit Liste von SSL-Zertifikaten
  - Originales Zertifikat enthält *public key*
- Anonym:
  - Server Key Exchange direkt mit *public key*
- Eines von beiden muss geschickt werden

SSL Handshake: Server Hello Done

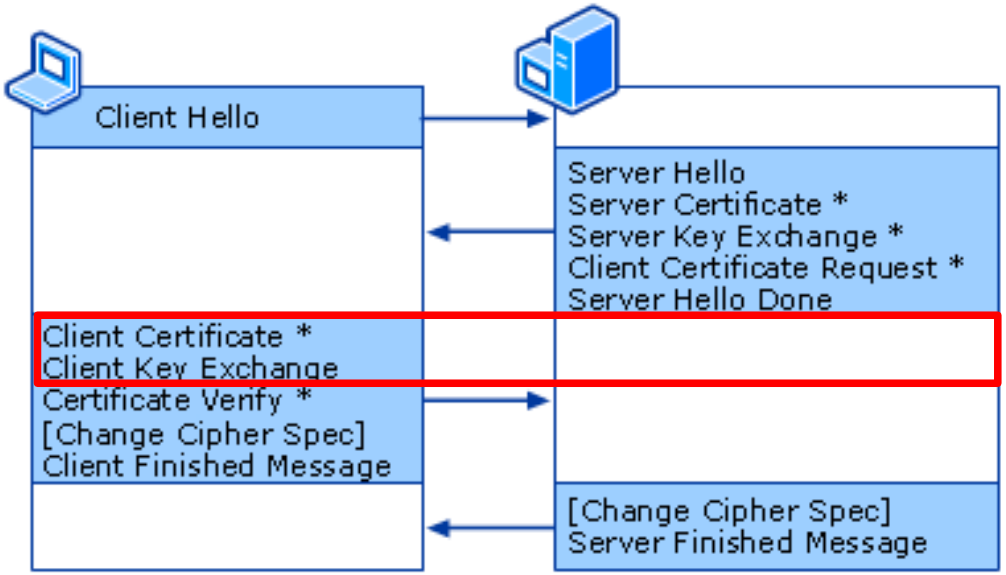


SSL Handshake Protocol

- Client Certificate Request *
  - Nur bei bilateraler Authentifizierung
- Server Hello Done
  - Kein Inhalt
  - Signalisiert Ende der Antwort

* Optional

SSL Handshake: Client Key Exchange

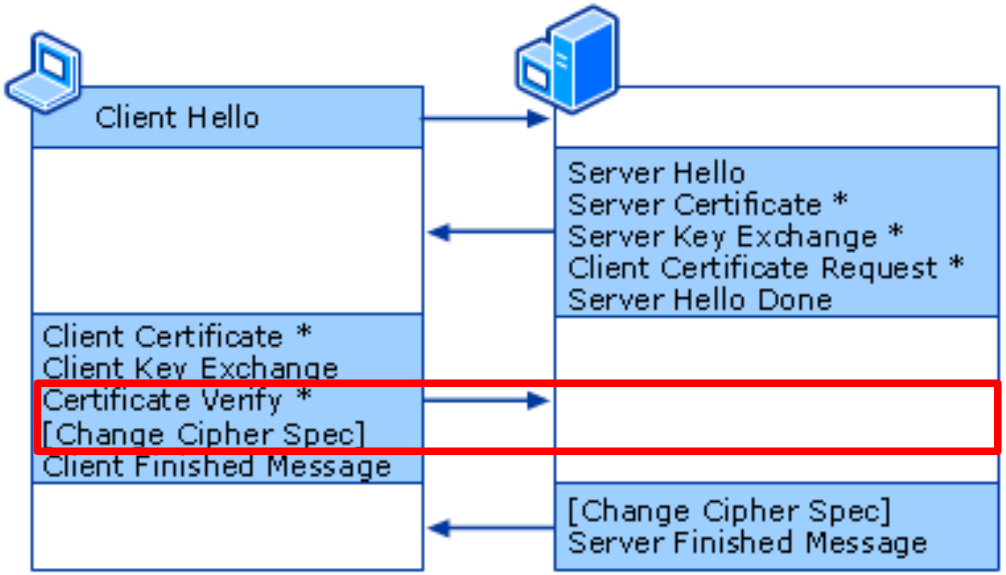


SSL Handshake Protocol

- Client Certificate *
  - Nur bei vorherigem Client Certificate Request
- Client Key Exchange
  - Client Protokoll Version
  - *Premaster secret* aus Zufallsnummern
  - Mit *public key* des Servers verschlüsselt

* Optional

SSL Handshake: Change Cypher Spec

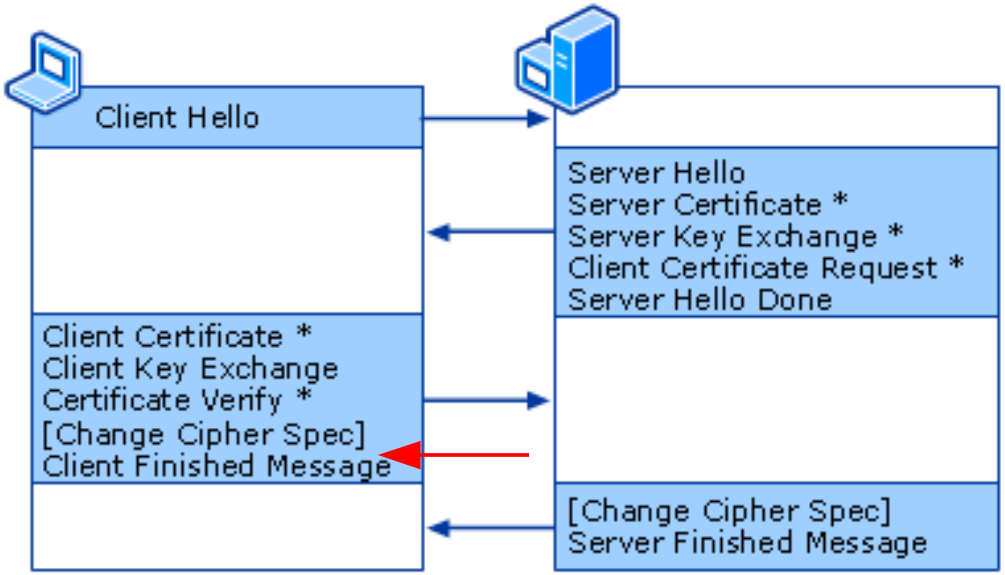


SSL Handshake Protocol

- Certificate Verify *
  - Nur bei vorherigem Client Certificate
- Change Cypher Spec
  - Client ist bereit
  - Alle zukünftigen Nachrichten sind mit gemeinsamen key verschlüsselt

* Optional

SSL Handshake: Master Secret

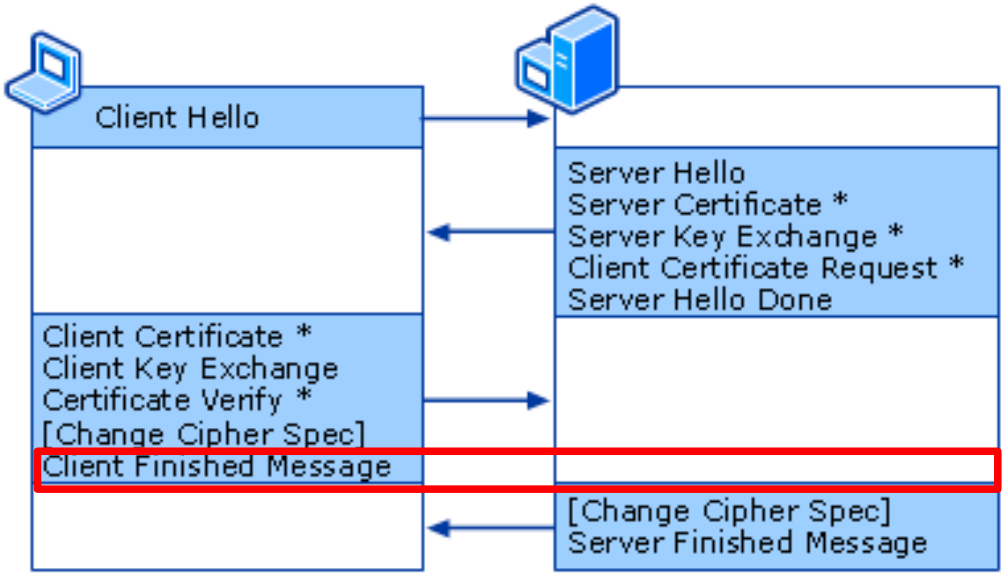


SSL Handshake Protocol

* Optional

- Erzeugung des *master secrets* aus *premaster secret* und Zufallsnummern
- Erzeugung von Client/Server *write keys* und Client/Server *MAC (Message Authentication Code) write keys* aus *master secret* jeweils lokal auf Client/Server

SSL Handshake: Client Finished

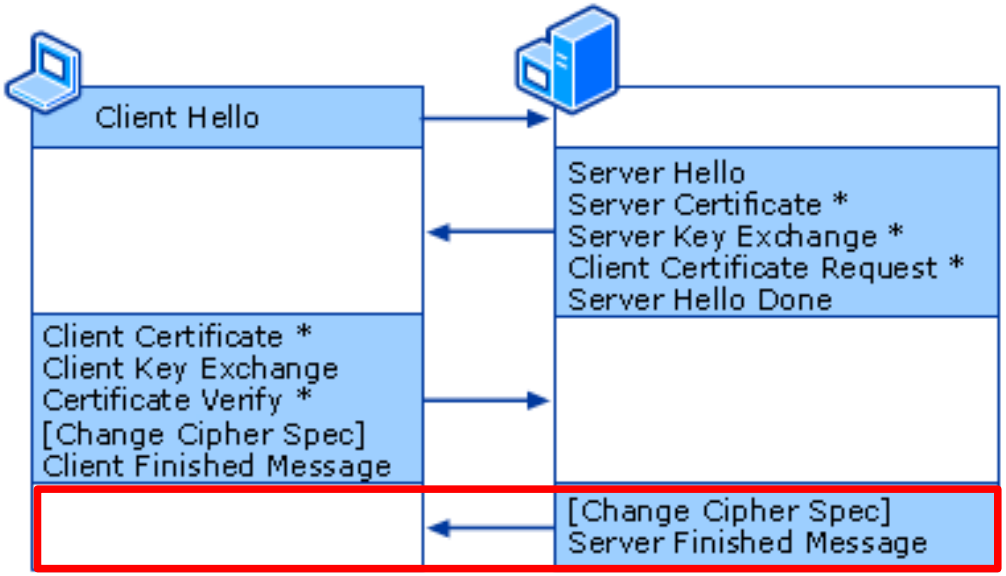


SSL Handshake Protocol

- Hash aller bisherigen Nachrichten
- Extra geschützt mit MAC, bestehend aus Hash der Nachricht und *client MAC write key*
- Bereits mit *client write key* verschlüsselt

* Optional

SSL Handshake: Server Finished

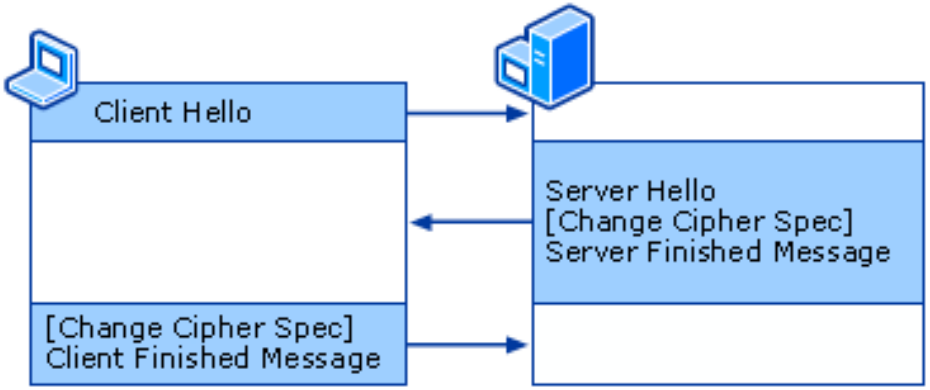


SSL Handshake Protocol

- Change Cypher Spec
  - Analog zu Client
- Server Finished
  - Ebenfalls mit MAC geschützt, diesmal aus Hash + *server MAC write key*
  - Ebenfalls bereits verschlüsselt, diesmal mit *server write key*

* Optional

SSL Handshake: Resume



SSL Resume Protocol[1]

- Zur schnelleren Wiederaufnahme vorheriger Verbindungen
- Hauptsächlich für Secure HTTP nützlich
- Normale SSL Handshake: viel asymmetrische Verschlüsselung
- Resume Handshake: Nur symmetrische

Literaturliste

[1] http://technet.microsoft.com/en-us/library/cc783349(WS.10).aspx

[1] http://msdn.microsoft.com/en-us/library/bb931353(VS.85).aspx

[1] John Viega, Matt Messier, Pravir Chandra: "Network Security with OpenSSL", 2002, O'Reilly, Sebastopol

[1] http://technet.microsoft.com/en-us/library/cc758686(WS.10).aspx

[1] http://computing.ece.vt.edu/~jkh/Understanding_SSL_TLS.pdf

[2] http://www7.informatik.uni-erlangen.de/~dressler/lectures/netzwerksicherheit-ws0708/12_TransportLayerSecurity.pdf