Kommunikationssicherheit

Inhalt: Sicherungsprotokolle auf verschiedenen Schichten

- Data Link Layer: IEEE 802.1x – portbasierte Netzzugangskontrolle; Point-to-Point Protocol
- Netzwerkschicht: IPsec => skalierbare Sicherungsfunktionen
- Transportschicht: TLS => Handshake-Dialog (Schlüsselaushandlung, Authentisierung)

<table>
<tr><td rowspan="2">5. Anwendungs-schicht</td><td>S/MIME | PGP</td><td rowspan="2">HTTP</td><td rowspan="2"></td></tr>
<tr><td>SMTP</td></tr>
<tr><td rowspan="2">4. Transport-schicht</td><td colspan="2">SSL, TLS, SSH</td><td rowspan="2">UDP, ICMP</td></tr>
<tr><td colspan="2">TCP</td></tr>
<tr><td rowspan="2">3. Netzwerk-schicht</td><td colspan="3">IPSec</td></tr>
<tr><td colspan="3">IP</td></tr>
<tr><td>2. Data Link Layer</td><td colspan="3">IEEE 802.1x, EAP, PPP, PPTP, …</td></tr>
<tr><td colspan="4">1. Bitübertragungsschicht</td></tr>
</table>

Schicht 1: Bitübertragung

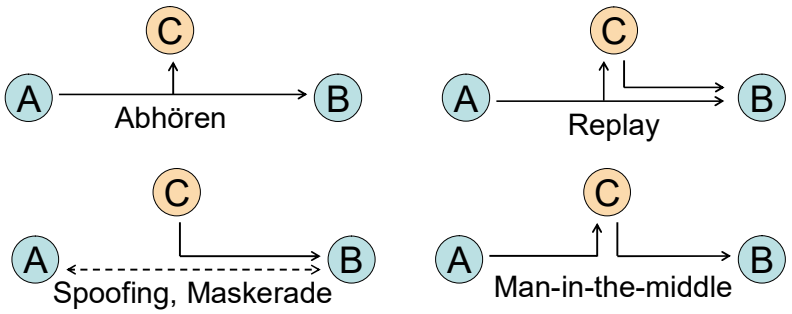
Schicht 2: Data Link Layer: Kontrollierte Nutzung eines Datenübertragungsmediums

Schicht 3: Netzwerkschicht: Kommunikation zwischen Endsystemen

Schicht 4: Transportschicht: Kommunikation zwischen Anwendungsprozessen

1 Angriffe auf die Kommunikation

- Abhören (Sniffing)
- Spoofing: Vortäuschen einer falschen Identität, C versucht sich als A zu maskieren
- Replay: Wiedereinspielen aufgezeichneter Nachrichten
- Man-in-the-middle



2 Data Link Layer

Aufgaben Data Link Layer:

- Übertragung von Frames zwischen direkt benachbarten Stationen
- Fehlererkennung und -korrektur
- Kontrollierte Nutzung eines gemeinsamen Datenübertragungsmediums (Medium access control)

IEEE 802 LAN/MAN-Standardisierungskomitee:

- Standards für LAN und MAN
- IEEE 802.3: Ethernet-Famile
- IEEE 802.5: Token-Ring
- IEEE 802.11: Wireless LAN

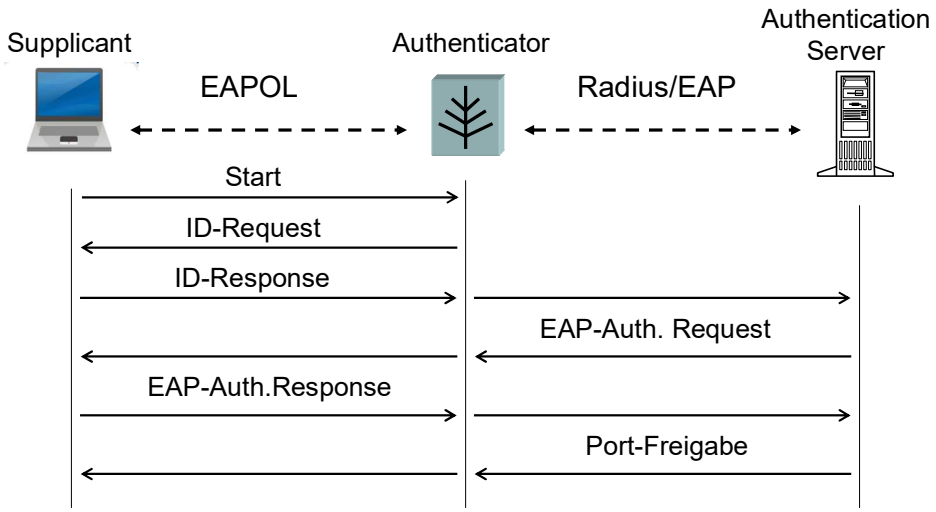
Sicherheitsprotokolle erfüllen Aufgaben, die über die originären Aufgaben der Schicht 2 gemäß des ISO/OSI-Schichtenmodells hinausgehen.

IEEE 802.1x: Authentication Framework

- Ergänzt andere IEEE 802 Standards durch optionale Zugriffskontrolle
- Portbasierte Zugriffskontrolle (Portbased Network Access Control, PNAC) für LANs und WLANs
- Zwei Arten logischer Kommunikationsendpunkte für Ports:
  - unkontrollierter Endpunkt (Port): ermöglicht Authentisierungsdialog, keine Nutzdatenübertragung
  - kontrollierter Endpunkt (Port): wird nach erfolgreicher Authentisierung freigeschaltet, ermöglicht Nutzdatenübertragung
- Port: Zugangspunkt einer Punkt-zu-Punkt Verbindung
  - Zugangsport eines Ethernet-Switches
  - Zugangspunkt eines WLAN-AccessPoints
  - an einem Port ist genau ein Gerät angeschlossen

- IEEE 802.1x definiert keine eigenen Authentisierungsprotokolle, sondern empfiehlt Einsatz bestimmter Protokolle:
  - Extensible Authentication Protocol (EAP, RFC 5247)
    IEEE 802.1x definiert Kapselung von EAP over LAN (EAPOL)
  - Einsatz von RADIUS (Remote Authentication Dial In User Service, RFC 2865/2866)
- EAP unterstützt zahlreiche Authentisierungsweisen, z.B.:
  - EAP Pre-Shared Key (EAP-PSK)
  - EAP Protected One-Time Password (EAP-POTP)
  - EAP Transport Layer Security (EAP-TLS)

IEEE 802.1x Authentifizierungsszenario



3 Virtual Private Networks (VPNs)

- Ein VPN ist ein logisches privates Netz innerhalb einer öffentlichen Netzinfrastruktur
- es ist „virtuell“:
  - das VPN abstrahiert von den vorhandenen Netzen, es kann sich über Netze mehrerer Anbieter (network provider) erstrecken.
- es ist „privat“:
  - das VPN als solches bildet ein geschlossenes Netz. Nur eine wohldefinierte Gruppe von Instanzen kann auf das VPN zugreifen. Instanzen außerhalb des VPN können nicht auf das VPN und in diesem übertragene Daten zugreifen.
- es ist ein Netz:
  - obwohl es virtuell ist, muss das VPN von allen berechtigten Instanzen wie ein „normales“ Netz nutzbar sein.

Möglichkeiten zur Bildung von VPNs

- Routenfilterung und kontrollierte Routenpropagierung:
  - Einschränkung der Routing-Informationsverteilung, so dass nur Knoten innerhalb des VPNs Routen zueinander kennen.
  - Sicherheit basiert darauf, dass das Routing entsprechend eingeschränkt ist.
- Tunneling durch das Internet
  - Nutzung von IPSec, SSH, TLS o.a.

Typischer Einsatzzweck von VPNs: Erweiterung von Firmennetzen über das Internet hinweg zu

- Filialen (Branch Offices): **Intranet VPN**
- Geschäftspartnern (Business Partners): **Extranet VPN**
- mobilen Nutzern und Telearbeitsplätzen (Remote User): **Remote Access VPN**

Kostenaspekte von VPNs über das Internet: (Infonetics Research 1997)

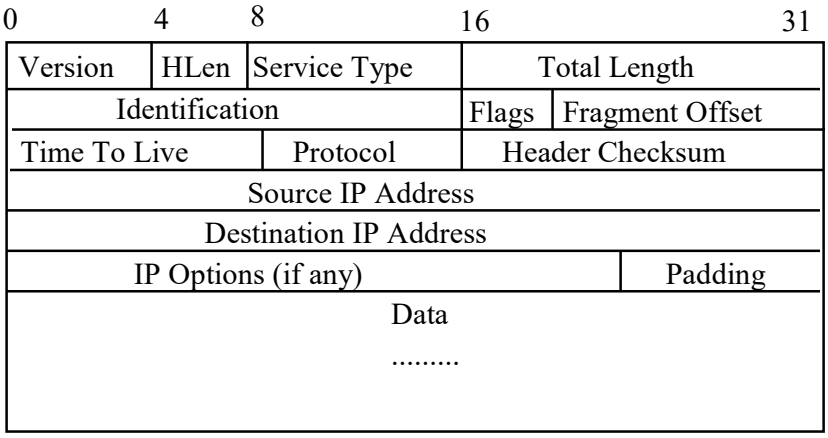
- Nutzung von VPNs ist 20% bis 47% günstiger gegenüber Anmietung dedizierter WAN-Standleitungen.
- Nutzung von VPN für Remote Access ist 60%-80% günstiger als Dial-In Zugriffe

Im Folgenden werden zwei Ansätze zur Realisierung von VPNs genauer betrachtet:

IPSec: Mehrere Varianten des Schutzes bei der Übertragung (AH/ESP, Transport/Tunnel)

TLS: Varianten der Schlüsselaushandlung stehen im Vordergrund

Wiederholung: IP-Paketformat



- Version (IPv4, IPv6)
- HLen (Header Length): in 4-Byte-Worten (mindestens 5, höchstens 15)
- Service Type: für Quality of Service Einstellungen
- Total Length: Gesamtlänge des IP-Pakets (maximal 65.535 Bytes)
- Time To Live: Lebenszeit eines Pakets
- Protocol: Kennung des höheren Protokolls ("Typfeld")
- Data: Nutzdaten (von höherer Schicht)

Beim Entwurf von IPv4 wurden Sicherheitsaspekte nicht in die Entwicklung einbezogen:

- IP bietet keine Nachrichtenauthentisierung und Datenintegrität
- IP bietet keinen Schutz gegen ein Wiedereinspielen von Paketen
- IP bietet keinen Schutz gegen Abhören

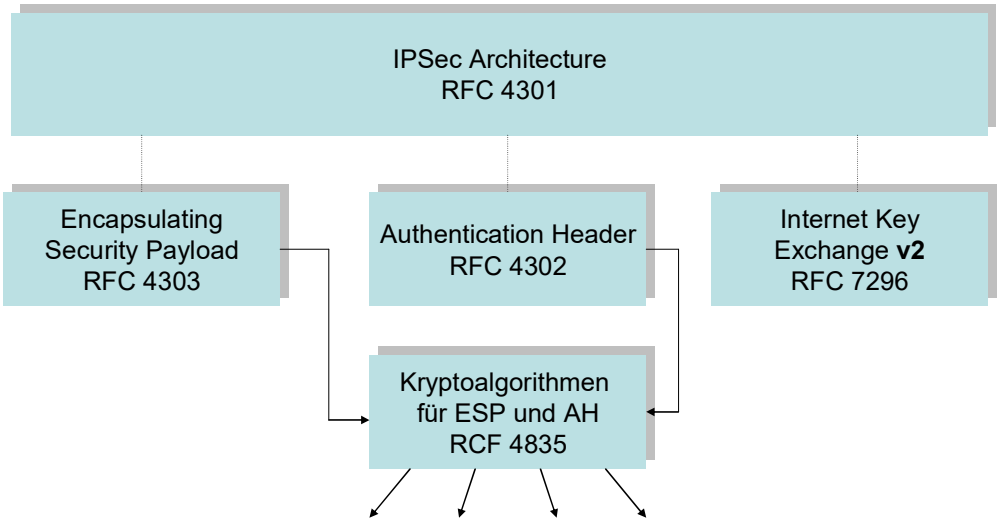
Bei IPv6 wurden Sicherheitsaspekte berücksichtigt. Diese werden über IPv6 Extension-Header bereitgestellt (AH, ESP im Next Header Feld).

Absicherung auf IP-Ebene (Verschlüsselung, Authentisierung):

- Vorteil: Absicherung des gesamten IP-Netzverkehrs (TCP, UDP, alle Anwendungen) möglich
- Nachteil: Authentisierung nur auf IP-Ebene, keine Nutzer-Authentifizierung

4 Sicherungsprotokolle auf der Netzwerkschicht am Beispiel IPSec

- IPSec wurde 1998 in den RCFs 2401 und folgende standardisiert
- Überarbeitung 2005: RFCs 4301 und folgende
- Überarbeitung Algorithmenvorgaben in 2007: RFC 4835



RFCs der Kryptoalgorithmen

IPSec bietet folgende Sicherheitsfunktionen:

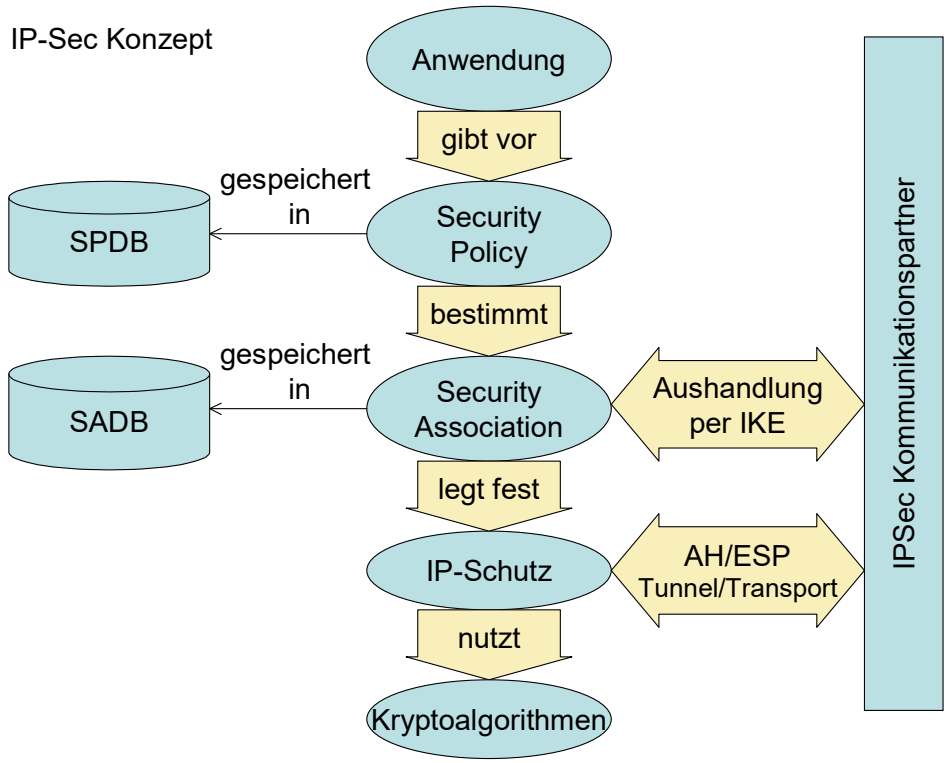
- Authentizität der IP-Quell- und IP-Zieladresse sowie der Nutzdaten
- Schutz vor Wiedereinspielen (Replay) von Paketen
- Schutz gegen Abhören übertragener Nutzdaten

IPSec stellt hierzu zwei Sicherheitsprotokolle bereit:

- Authentication Header (AH): Authentizität und Replay-Schutz
- Encapsulating Security Payload (ESP): Vertraulichkeit der Nutzdaten und/oder Authentizität und Replay-Schutz (mindestens eine Funktion muss aktiviert sein)

IPSec Management:

- Anwendungsspezifische Security Policies werden in der „Security Policy Database“ (SPDB) verwaltet.
- Tunnelspezifische Security Associations (SAs) werden hieraus abgeleitet und in SADB verwaltet.
- SAs werden über IKE ausgehandelt.



Security Associations:

- IP ist verbindungsloses Protokoll. Für IPSec ist jedoch die Speicherung von Zuständen (verwendete Kryptoalgorithmen, Schlüssel, etc.) erforderlich.
- Die Speicherung der erforderlichen Informationen erfolgt in Security Associations (SA)
- Beide Kommunikationspartner müssen SA speichern.
- Jede SA gilt nur für eine Kommunikationsrichtung.
- Jede SA wird eindeutig durch ein Tripel identifiziert:
  - Security Parameter Index (Teil des IPSec-Headers)
  - IP-Zieladresse
  - Sicherheitsprotokoll (AH oder ESP)
- SAs können zwischen folgenden Instanzen eingerichtet werden:
  - zwischen zwei Endsystemen
  - zwischen zwei Zwischensystemen (z. B. Routern)
  - zwischen einem Endsystem und einem Zwischensystem
- aktive SAs werden in der Security Association Database (SADB) verwaltet

Wie werden SAs zwischen zwei Instanzen eingerichtet?

- IKEv2 definiert Protokolle für Authentisierung und Schlüsselaustausch

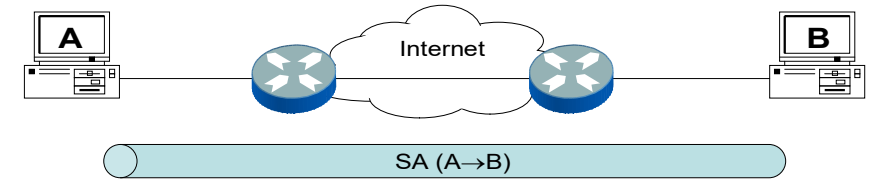
Security Policy Database (SPDB):

- Was soll bei der Übertragung wie gesichert werden?
- Selektoren für SPDB Einträge:
  - IP-Quelladresse, IP-Zieladresse
  - Name: DNS Name oder X.500 Name
  - Protokoll: Identifizierung des Transportprotokolls inkl. Port
- Inhalt von SPDB Einträgen:
  - Sicherheitseinstellungen:
    - » AH oder ESP,
    - » Tunnel- oder Transportmode,
    - » anzuwendende Algorithmen,
    - » Lebensdauer zugehöriger SAs
  - Vorgaben für Schlüsselaushandlung

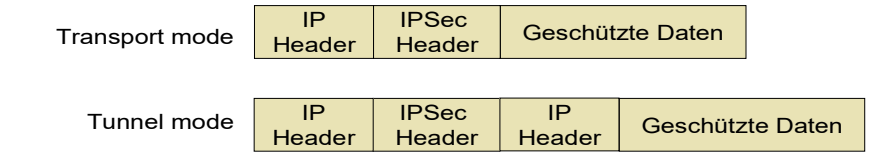
Transportmodus vs. Tunnelmodus

Sicherheitsassoziationen werden in einer der folgender Modi betrieben:

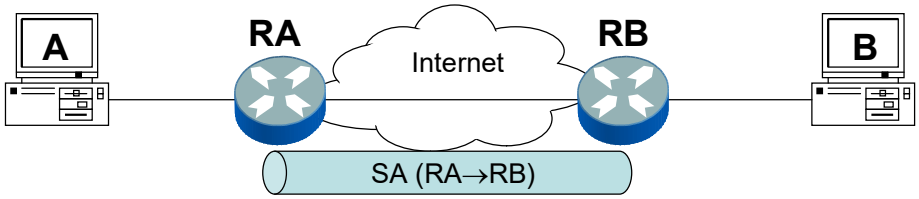
- Transportmodus (Transport mode):
  - Original IP-Header wird durch IPSec-Header ersetzt
  - Kann nur gewählt werden, wenn Endpunkte der IPSec-Kommunikation mit den IP-Kommunikationsendpunkten übereinstimmen.
- Tunnelmodus (Tunnel mode):
  - Original IP-Header ist Bestandteil der geschützten Daten
  - Kann zwischen beliebigen Kommunikationspunkten genutzt werden.



Paketstruktur:

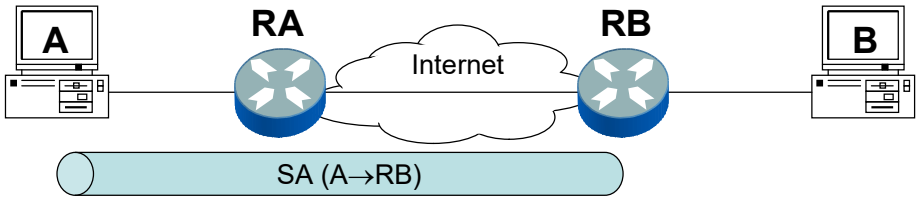


Szenario: Verbindung von Intranets



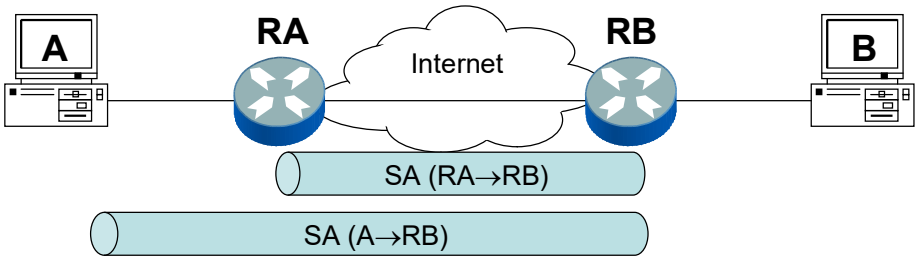
Paketstruktur:	IP Header	IPSec Header	IP Header	Geschützte Daten
	Src=RA Dst=RB		Src=A Dst=B	

Szenario: Telearbeit



Paketstruktur:	IP Header	IPSec Header	IP Header	Geschützte Daten
	Src=A Dst=RB		Src=A Dst=B	

Verschachtelung von Sicherheitsassoziationen



Paketstruktur:

IP Header	IPSec Header	IP Header	IPSec Header	IP Header	Geschützte Daten
Src=RA Dst=RB		Src=A Dst=RB		Src=A Dst=B	

ursprüngliches Paket

mit SA(A →RB) geschütztes Paket

zusätzlich SA(RA →RB) geschütztes Paket

Schutz vor Wiedereinspielen

Schutz vor Wiedereinspielen von Paketen durch Sequenznummern, die in die Berechnung des Nachrichtenauthentisierungswerts einbezogen werden:

- IPSec-Header enthält 32 Bit Sequenznummer (SN)
- SN wird bei Einrichtung der SA mit 0 initialisiert
- SN wird mit jedem gesendeten Paket um 1 erhöht
- Vor dem Überlauf ist ein neuer Schlüssel zu vereinbaren.

IP-Pakete können in falscher Reihenfolge beim Empfänger ankommen

Sequenznummernverwaltung beim Empfänger

- Sequenznummernverwaltung erfolgt durch ein gleitendes Fenster (Sliding Window)
- Fensterbreite gemäß RFC 2401 mindestens 32 Bit, empfohlen sind 64 Bit.
- Fenster wird immer so verschoben, dass das empfangene Paket mit der höchsten Sequenznummer „vorne“ ist.
- Für empfangene Pakete wird das Fensterbit auf 1 gesetzt, für nicht empfangene Pakete ist es 0.
- Ein empfangenes Paket, dessen Sequenznummer kleiner ist als die letzte Sequenznummer im Fenster, wird als zu alt verworfen.

IPSec Protokollverarbeitung

IPSec Protokollverarbeitung beim Senden eines IP-Pakets:

1. Anhand der SPDB prüfen, ob das Paket geschützt werden muss.
2. Bestimmung der zugehörigen SA. Falls noch keine zugehörige SA besteht, sind zunächst die erforderlichen Parameter (Schlüssel, etc.) über das Schlüsselmanagement auszuhandeln.
3. Lesen der SA aus der SADB (bzw. Speichern der SA)
4. Sicherung des IP-Pakets gemäß Angaben der SA (AH oder ESP, Transport oder Tunnel, etc.)
5. Weiterverarbeitung des Pakets: Fortsetzung bei IPSec-Verarbeitung (aufgrund möglicher IPSec Schachtelung)

IPSec Protokollverarbeitung beim Empfang eines IP-Pakets:

1. Prüfung, ob das Paket einen IPSec-Header aufweist. Nur in diesem Falle ist eine IPSec Bearbeitung erforderlich.
2. Verarbeitung des äußeren IPSec-Headers
   - Prüfung der Sequenznummer
   - Prüfung des Nachrichtenauthentisierungswerts
   - Ggf. Entschlüsselung
3. Anhand der SPDB prüfen, ob das Paket korrekt gesichert war.
   - Speicherung von Zustandsinformationen über die bisherigen Sicherungsmaßnahmen, damit später mit der SPDB verglichen werden kann.
   - Ggf. Bearbeitung weiterer IPSec-Header

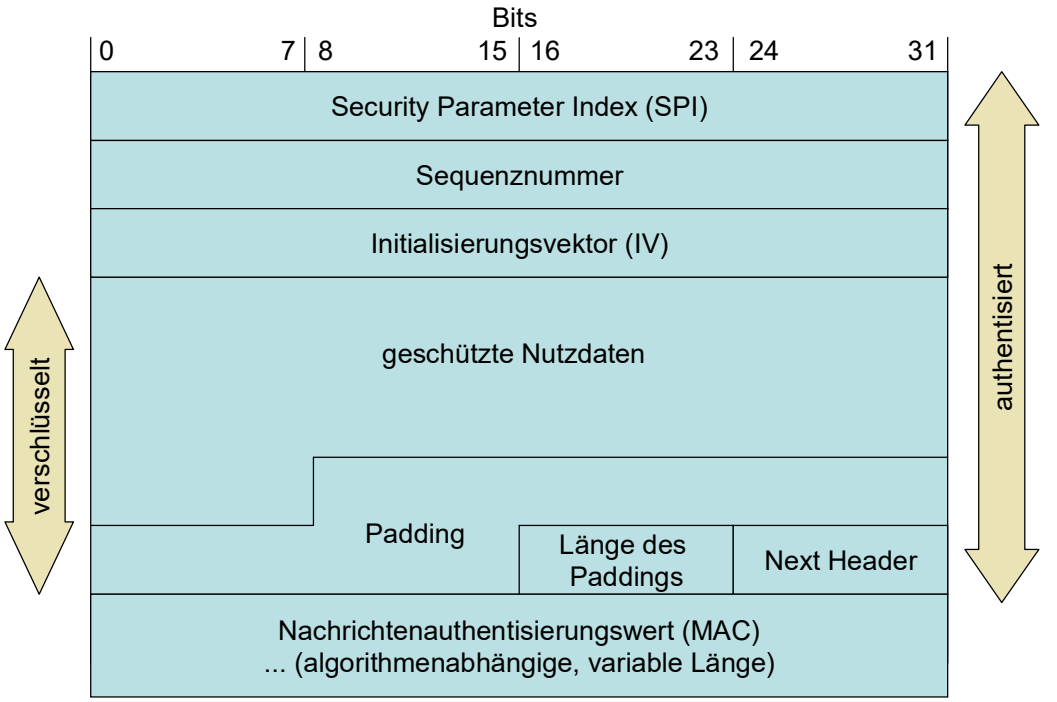
Encapsulating Security Payload (ESP)

RFC 4303: Generische Beschreibung des Protokolls

RFC 4835 spezifiziert zu unterstützende Kryptoverfahren für ESP und AH

- Vertraulichkeit: Ohne (RFC2410), AES-CBC-128 (RFC3602), TripleDES-CBC (RFC2451)
- Authentisierung: HMAC-SHA1-96 (RFC2404)
- Für Authentisierung und Verschlüsselung können verschiedenen Schlüssel genutzt werden.

ESP-Paketformat



IPSec ESP Paketformat

- SPI: Kennzeichnet zusammen mit der IP-Zieladresse eindeutig die SA für dieses Paket. Wird vom Empfänger vorgegeben, da dieser anhand der SPI die zugehörige SA identifizieren können muss.
- IV: für den kryptographischen Algorithmus
- Padding: Auffüllen der zu sichernden Nachricht auf ein Vielfaches der Blockgröße der Blockchiffre.
- Next Header: Protokolltyp der enthaltenen Nutzlast
  (IP, TCP, UDP)

Ausgehende Bearbeitung eines IP-Pakets bei ESP:

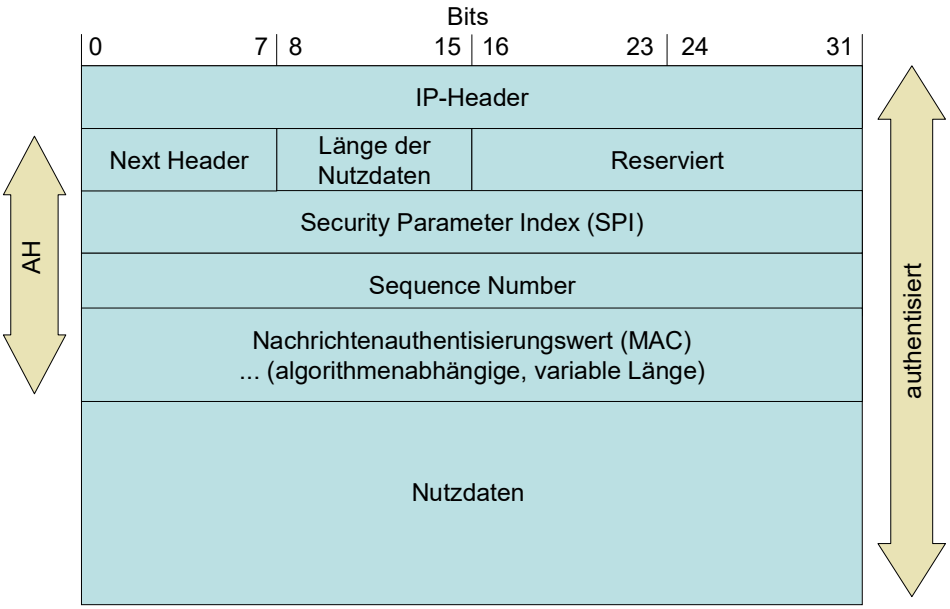
1. Falls Transportmodus zu verwenden: Transportmodus Header erstellen
   - Füge ESP-Header hinter IP-Header ein
   - ESP.nextHeader := IP.nextHeader
   - IP.nextHeader := ESP
   - Belege andere ESP-Header Felder
2. Falls Tunnelmodus zu verwenden: Tunnelmodus Header erstellen
   - Setze ESP-Header vor IP-Header: ESP.nextHeader := IP
   - Belege andere ESP-Header Felder
   - Setze neuen IP-Header vor ESP-Header
   - » NewIP.nextHeader := ESP
   - » NewIP.Src := this.IP-Address
   - » NewIP.Dst := tunnelEnd.IP-Address
3. Falls chiffriert werden soll: Verschlüsseln der Nutzlast
4. Falls authentisiert werden soll: Berechnung des MAC
5. Berechne Prüfsumme des äußeren IP-Headers

Eingehende Bearbeitung eines IP-Pakets bei ESP:

1. Warte solange, bis alle IP-Segmente des Pakets angekommen sind.
2. Bestimme zum SPI gehörendes SA. Kein SA zugeordnet
   -> ERROR
3. Prüfe die Sequenznummer des Pakets. SN außerhalb des zulässigen Fensterbereichs
   -> ERROR
4. Falls Authentisierungswert gegeben: Prüfe MAC auf der Basis im SA angegebener Algorithmen und Schlüssel. MAC inkorrekt -> ERROR
5. Verschiebe Sequenznummernfenster
6. Falls Nutzlast verschlüsselt: Entschlüssele Nutzlast auf der Basis im SA angegebener Algorithmen und Schlüssel.
7. Falls Tunnelmode: Verwerfe äußeren IP-Header und ESP-Header
8. Falls Transportmode:
   - IP.nextHeader:=ESP.nextHeader
   - Verwerfe ESP-Header und berechne IP-Prüfsumme neu
9. Prüfe, ob Paket und SA der Policy (SPD) entsprechen. Falls nicht -> ERROR
10. Falls ERROR verwerfe das Paket, ansonsten leite es weiter:
    - Neuer IPSec-Header -> IPSec-Verarbeitung
    - IP-Paket für bearbeitendes System bestimmt -> an höhere Protokollinstanzen weiterreichen (z. B. TCP, UDP)
    - IP-Paket nicht für bearbeitendes System bestimmt -> weiterleiten gemäß Routing

Authentication Header (AH)

Paketformat des AH



IPSec AH Paketformat

- SPI: Kennzeichnet zusammen mit der IP-Zieladresse eindeutig die SA für dieses Paket. Wird vom Empfänger vorgegeben, da dieser anhand der SPI die zugehörige SA identifizieren können muss.
- Next Header: Protokolltyp der enthaltenen Nutzlast
  (IP, TCP, UDP)
- Länge der Nutzdaten: „Payload length“ = Länge der Felder Sequence Number und MAC
- Reserviert = für zukünftige Anwendungen, derzeit 0.
- MAC-Algorithmen: HMAC-MD5-96, HMAC-SHA-1-96, HMAC-RIPEMD160-96
  - „96“: Hashwert wird auf die höherwertigen 96 Bit gekürzt.
- MAC: wird über gesamtes IP-Paket inkl. der sich nicht verändernden Felder des IP-Headers gebildet. Möglicherweise Änderungen unterworfene Felder werden für die MAC-Berechnung als 0 angenommen.
  - Nicht mit einbezogen werden: Type of Service, Flags, Fragment Offset, TTL, Header Checksum

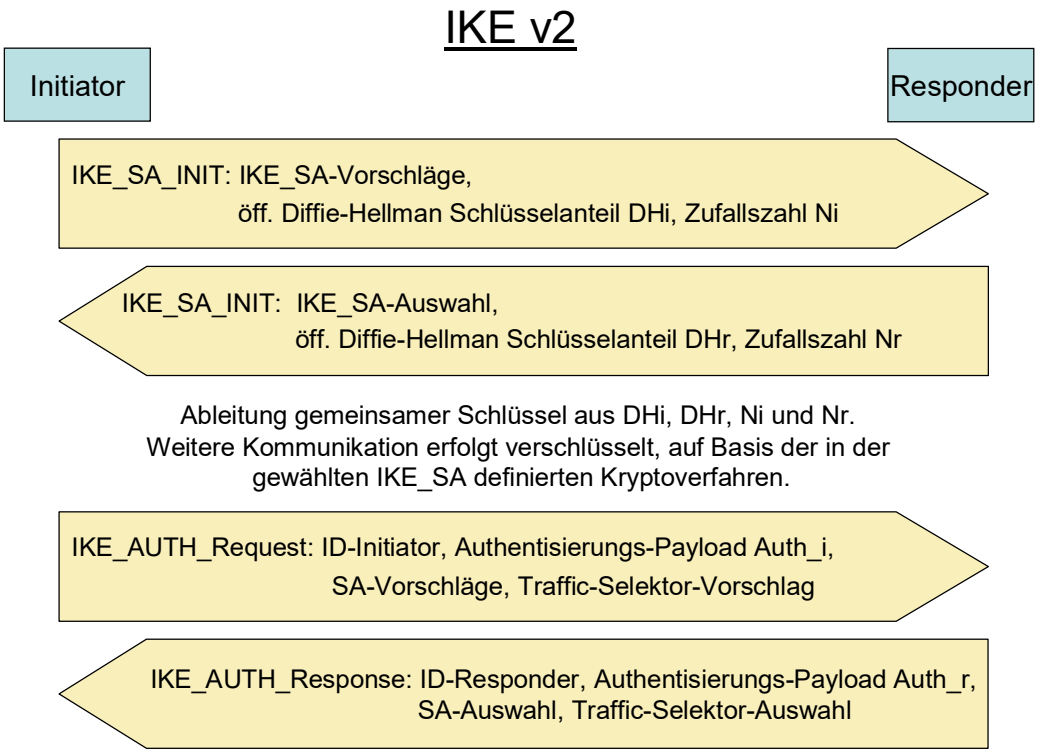
Gleichzeitige Nutzung von ESP und AH ist möglich!

- Werden AH und ESP gleichzeitig für eine Verbindung genutzt, so muss spezifikationsgemäß ESP vor AH angewendet werden!

IKE v2

Aushandlung von SAs und Authentisierung in 4 Nachrichten

- Nachrichten 1+2: Aushandlung von Algorithmen und Schlüsseln für weitere Nachrichten
- Nachrichten 3+4: Authentisierung und Aushandlung von SAs für den IPSec Datenaustausch



Authentisierung, Inhalte der Authentisierungs-Payloads:

Mit Pre-shared-Keys (symmetrische Verfahren, MAC)

- Über einen fest definierten Datenblock, der auch die Zufallszahl des Kommunikationspartners und die eigene Identität enthält, wird mit vorher verteilten Keys (pre-shared keys) ein Message-Authentication-Code gebildet.

Mit Public-Key Verfahren auf Basis von Zertifikaten

- Über den o.g. Datenblock wird eine Signatur berechnet. Zertifikat und Signatur werden mitgeschickt.

Traffic-Selektor Vorschlag und Auswahl

- Initiator kann einen IP-Adressbereich und TCP Portbereich für die IPSec Datenübertragung vorschlagen.
- Der Responder kann diesen Bereich bei Bedarf einschränken.

Schutz vor DoS-Angriffen:

- Bei der Schlüsselaushandlung werden aufwändige Diffie-Hellman Berechnungen durchgeführt. Viele gleichzeitige Anfragen lasten den Server schnell aus. Es besteht Gefahr eines DoS-Angriffs.
- Basisschutz: Bei vielen gleichzeitigen Anfragen (DoS-Verdacht) schickt Responder ein zufällig gewähltes „Cookie“ und der Initiator muss seine Anfrage inkl. Cookie wiederholen.
  - Rechenintensive DH-Operationen werden nur ausgeführt, wenn das „Cookie“ korrekt ist.
- Schutzwirkung: DoS-Angriff ist zwar weiterhin möglich jedoch aufwändiger!
  - » Zwei Nachrichten wären notwendig
  - » Cookie muss vom Angreifer empfangen werden
  - » Zweite Nachricht muss vom Angreifer mit Cookie generiert werden

IPSec Randprobleme:

- Firewalls müssen getunnelte Protokolle passieren lassen
  - Enable IP Forwarding
  - Permit UDP port 500 für IKE
  - Permit IP protocols 50 und 51 für ESP und AH
  - Permit UDP port 1701 für L2TP und L2F
  - Permit IP protocol 47 (GRE) und TCP port 1723 für PPTP
- Bei Ende zu Ende Verschlüsselung ist den Firewalls keine Inspektion der IP-Header und IP-Daten möglich, z. B. zur Prüfung auf Virenbefall.
- Probleme bei Network Address Translation:
  - NAT erfordert Änderung der Adressen von IP-Paketen in Abhängigkeit von Schicht 4 Protokollinformationen (z. B. Ports)
  - Bei Verwendung von AH verwirft Empfänger geänderten IP-Header
  - Bei getunnelten, verschlüsselten Paketen ist der IP-Header mit geschützt
  - Bei verschlüsseltem Transportmodus können IP-Header zwar eingesehen und verändert werden, jedoch der Schicht 4 Protokollkopf ist verschlüsselt.
  - Lösung: NAT-Traversal for IPSec (in IKEv2 enthalten): Tunnelung der IPSec-Pakete in UDP-Datagrammen

5 Sicherheitsprotokolle der Transportschicht am Beispiel SSL/TLS

Secure Socket Layer (SSL):

- Client Server Protokoll
- 1994 von Netscape Communications entwickelt: v1 (1994), v2 (1995), v3 (1996)
- SSL Version 2.0 enthielt einige Schwachstellen, SSL Version 3.0 Grundaufbau sicher

Transport Layer Security (TLS):

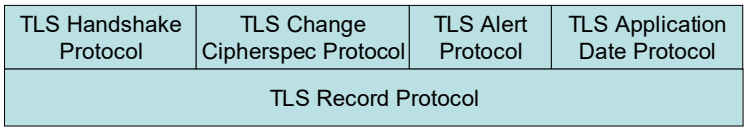
- TLS 1.0 (IETF RFC 2246, 1999, auf SSL v3 basierend).
- TLS 1.1 (RFC4346, 2006, Beast Angriff, CBC-IV-Schwachstelle): jetzt CBC-IV zufällig
- TLS 1.2 (RFC5246, 2008), Änderungen u.a.
  - MD5/SHA1 ersetzt durch SHA256 (PseudoRandomFunction)
  - TLS_RSA_WITH_AES_128_CBC_SHA muss unterstützt werden (mandatory)
  - RFC 5288 (2009) AES GCM for TLS1.2 (gegen Padding-Oracle-Angriffe, Poodle)
  - RFC 6176 (2011) backward compatibility to SSLv2 for TLS prohibited
  - RFC 6797 (2012) HTTP Strict Transport Security (HSTS)
  - RFC 6844 (2013) Certification Authority Authorization (CCA) im DNS
  - RFC 7465 (2015) Verbot von RC4,
  - RFC 7469 (2015) HTTP Public Key Pinning
  - RFC 7525 Recommendations for Secure Use of TLS (Heartbleed, etc.)
- TLS 1.3 (RFC 8446, 2018)
- (gemäß OSI-Schichtenmodell gehört SSL zur Sitzungsschicht)
- TLS ist heute das Standardprotokoll zur Absicherung von Web-Verbindungen
- Ursprünglich wurde SSL nur zur Absicherung von HTTP entwickelt, es ist jedoch auch zur Absicherung von FTP, SMTP, etc. geeignet. => OpenVPN

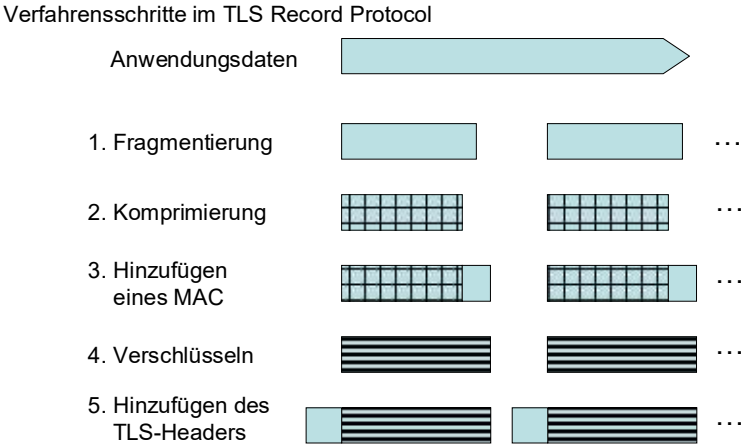
TLS Sicherungsfunktionen:

- Serverauthentisierung auf Basis eines X.509-Zertifikats
- Optionale Clientauthentisierung auf Basis eines X.509-Zertifikats
  - sofern vom Server gefordert
- Verschlüsselung ausgetauschter Nutzdaten
  - Optional. Kryptoverfahren und Schlüssellänge werden im Rahmen des Handshake festgelegt.
- Nachrichtenauthentisierung (Datenintegrität):
  - Jede Nachricht wird mit einem Message Authentication Code (MAC) gesichert.

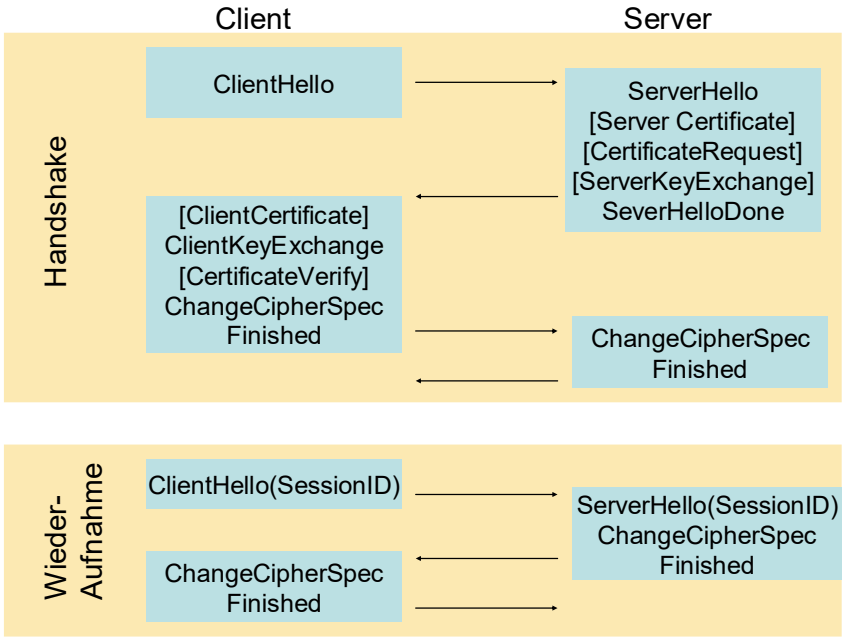
SSL Protokollarchitektur

- SSL Handschake: Aushandlung der eingesetzten Kryptoverfahren, Authentifizierung und Schlüsselaustausch
- SSL Change Cipherspec Protocol: Umschaltung auf geschützte Übertragung
- SSL Alert Protocol: SSL Fehlermeldungen
- SSL Application Data Protocol: gesicherte Datenübertragung





Allgemeiner Handshake Dialog (bis TLS 1.2):



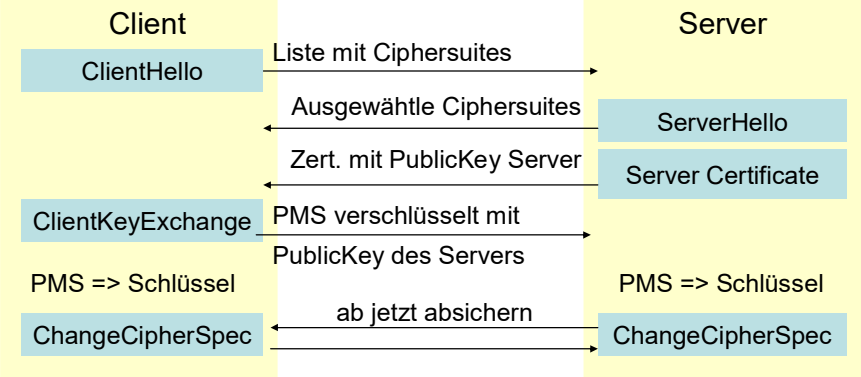
- ClientHello:
  - Verwendete Protokollversion, Zufallszahl, SessionID (0=neue Sitzung)
  - Liste unterstützter CipherSuites (nach Präferenz geordnet)
  - Liste unterstützter Kompressionsverfahren
- ServerHello:
  - Verwendete Protokollversion, Zufallszahl, ggf. SessionID,
  - ausgewählte CipherSuite
  - ausgewähltes Kompressionsverfahren
- ServerCertificate (optional)
  - Übertragung eines oder mehrerer X.509 Serverzertifikate des Servers an den Client
- CertificateRequest (optional)
  - Anforderung einer Zertifikatsauthentisierung beim Client
  - nur möglich, falls Server sich selbst durch ein Zertifikat ausgewiesen hat
- ServerKeyExchange (optional)
  - Daten zur Schlüsselaushandlung (z. B. Diffie Hellman Schlüsselteil, signiert durch den Server)
  - nur, falls kein ServerCert gegeben oder ServerCert lediglich öff. Signaturschlüssel (DSS) enthält
- ServerHelloDone
  - Signalisierung des Nachrichtenendes
- ClientCertificate (optional)
  - Übertragung des Clientzertifikats, falls der Server eines angefordert hat
- CertificateVerify (optional)
  - Client weist durch digitale Signierung nach, dass er im Besitz des geheimen Schlüssels ist, der zum in seinem Zertifikat abgelegten öffentlichen Schlüssel gehört
  - nur erforderlich, falls Client sich über ein Zertifikat ausweist
- ClientKeyExchange
  - RSA: 48 Byte Wert: PreMasterSecret (z. B. mit dem öffentlichen RSA-Schlüssel des Servers verschlüsselt)
  - DH: Diffie Hellman Schlüsselanteil
- ChangeCipherSpec
  - Der Client bzw. der Server signalisiert, dass alle nachfolgenden Nachrichten gemäß der ausgehandelten Art und Weise (CipherSuite) geschützt werden.
- Finished
  - Anzeige des Endes der Handshake-Phase
  - Finished Nachricht wird bereits verschlüsselt übertragen
  - Die SSL-Sitzung ist etabliert, sobald Client und Server finished Meldung erhalten haben.

Sitzung „resumable“

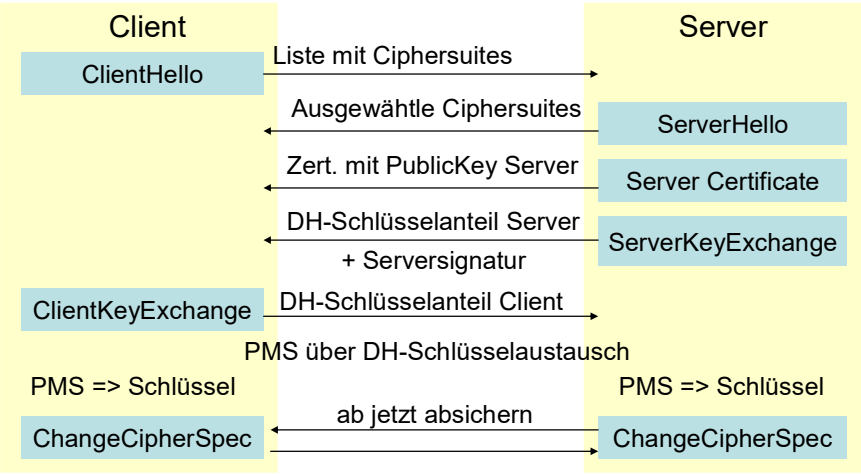
- Sitzung kann mehrfach verwendet oder zu einem späteren Zeitpunkt wieder aufgenommen werden. Hierzu werden neue Randoms in den Hello-Nachrichten ausgetauscht => verändertes MasterSecret
- Für HTTP wichtig, da HTML Dokumente häufig aus mehreren Elementen (HTML-Dokument, Grafiken, Klänge) bestehen, die über separate TCP-Verbindungen transportiert werden.

Zwei wesentlich unterschiedliche Handshake Varianten:

- Variante 1: Hybride Verschlüsselung mit RSA-Schlüsselaustausch
  - Client wählt PreMasterSecret (PMS) und schickt es verschlüsselt an den Server
  - verschlüsselt mit dem öffentlichen Schlüssel aus dem Serverzertifikat
  - (Voraussetzung: KeyUsage im Zertifikat erlaubt Einsatz des öff. Schlüssels zur Verschlüsselung)
  - Implizite Authentisierung des Servers: Nur der Server kann das verschlüsselte PMS mit seinem zum Zertifikat gehörigen privaten Schlüssel entschlüsseln.
  - TLS1.2 CipherSuite Beispiel: TLS RSA WITH …



- Variante 2: Schlüsselaustausch per Diffie-Hellman
  - Server signiert seinen DH-Schlüsselanteil und schickt Signatur an den Client
  - Client prüft Serversignatur mit öff. Schlüssel aus dem Serverzertifikat
  - TLS 1.2 CipherSuite Bsp.: TLS DHE RSA WITH…, TLS ECDHE ECDSA WITH…



- Zur Server-Authentisierung und weitere Prüfungen
  - Der Server schickt in der Server Certificate PDU die Zertifikate seines Zertifizierungspfads.
  - Der Browser sucht das Root-Zertifikat in der Liste seiner gespeicherten Zertifikate und prüft dessen eingestellte Vertrauenswürdigkeit.
  - Der Browser prüft des Weiteren, ob das Server-Zertifikat für diese Web-Domäne ausgestellt wurde (Namensvergleich mit dem Subject-Name im Zertifikat (i.d.R. der CommonName (CN))

Beispiele für ChipherSuites bis TLS1.2:

- Format:
  TLS KeyExchangeAlg With BulkCipherAlg MACAlg (=> MAC then Encrypt)
  TLS KeyExchangeAlg With AEAD-Alg PRF (=> authenticated encryption)
- TLS NULL With NULL NULL: keine Verschlüsselung, Authentisierung oder Integritätsprüfung
- TLS RSA With NULL SHA256: RSA-Schlüsselaustausch und Integritätssicherung (SHA256)
- TLS DH anon With RC4 128 SHA
  - Keine Serverauthentisierung, Schlüsselaushandlung per Diffie-Hellman (anonym), Verschlüsselung (RC4 128 Bit) und Integritätssicherung (SHA)
- TLS DHE DSS With AES 256 CBC SHA256
  - DH-Schlüsselaustausch (E = ephemeral = flüchtig), Serverauth. per DSS-Signatur Verschlüsselung (AES 256 CBC) und Integritätssicherung (SHA256)
- TLS ECDHE RSA With AES 256 GCM SHA256
  - ECDH-Schlüsselaustausch, Serverauth. per RSA-Signatur, Verschlüsselung und Integritätssicherung verbunden (AES 256 GCM), SHA256 nur noch zu Schlüsselableitung

Schlüsselableitung:

- MasterSecret wird vom Client und Server aus den ausgetauschten Werten berechnet:
- MasterSecret := PRF(PreMasterSecret, "master secret", ClientHello.random + ServerHello.random) [0..47];

Aus MasterSecret wird ein KeyBlock abgeleitet:

- KeyBlock:= PRF(SecurityParameters.MasterSecret, "key expansion", ClientHello.random + ServerHello.random);
- Aus dem KeyBlock werden dann eingesetzte Schlüssel nacheinander extrahiert

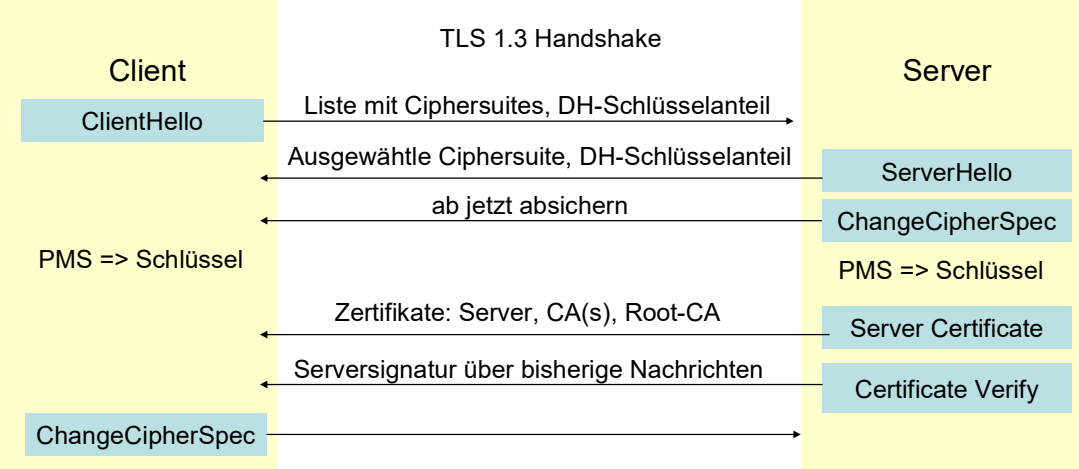
client_write_MAC_key,	server_write_MAC_key
client_write_key,	server_write_key
client_write_IV,	server_write_IV

TLS 1.3 (IETF RFC 8446, August 2018, 160 Seiten)

- nur noch DH-Schlüsselaustausch (DHE, ECDHE), Forward Secrecy,
  - kein RSA-Schlüsselaustausch mehr, kein statisches DH mehr
- dadurch vereinfachter Handshake-Dialog (nur ein Roundtrip)
  - schnellerer Verbindungsaufbau
  - Client macht Annahmen über vom Server unterstützte Verfahren und sendet schon mal DH-Schlüsselanteil mit
  - DH-Schlüsselaustausch erfolgt bereits in Client Hello und Server Hello Nachrichten
  - Server auhentisiert sich danach durch eine „Certificate Verify“ Nachricht mit einer Signatur über die bisherigen Handshake Nachrichten.
- nur noch Authenticated Encryption (GCM, CCM)
  - kein CBC mehr, keine MAC-then-Encrypt Problematik mehr
- Unterstützung neuer elliptischer Kurven (Curve 25519, Goldilock)
  - Alte Verfahren (MD5, SHA1 und RC4) werden nicht länger unterstützt
- Neuer Schlüsselableitungsmechanismus basierend auf HKDF (RFC5869)
  - HKDF: Hashed Key Derivation Function (durch iterierte Anwendung eines HMAC)

TLS 1.3 Ciphersuites

- Schlüsselaustausch-Verfahren sind nicht mehr Bestandteil der Ciphersuites
  - genutzte Verfahren in Client/Server HELLO Extensions, nicht mehr in Ciphersuite
- Namensaufbau der Ciphersuites: "TLS" AEAD Algorithmus, Hashfunktion für Schlüsselableitung
- Beispiele für Ciphersuites aus RFC8446
  - TLS_AES_128_GCM_SHA256 , TLS_AES_256_GCM_SHA384(RFC5116)
  - TLS_CHACHA20_POLY1305_SHA256 (RFC8439)
  - TLS_AES_128_CCM_SHA256 (RFC5116)



Typische SSL/TLS-Ports:

- HTTP via SSL (HTTPS) 443
- Telnet via SSL (TELNETS) 992
- DNS-over-TLS (2016, RFC7858) 853
- SMTP via SSL (SMTPS) 465
- POP3 via SSL (SPOP3) 995

OpenSSL

- Als SSL-Client den Verbindungsaufbau und –ablauf im Detail mit verfolgen
- Schlüsselpaare generieren und zugehörige Zertifikate erzeugen
- Verschlüsseln und Signieren zahlreichen Kryptoalgorithmen

OpenVPN

- Nutzung von TLS zur Kanalabsicherung
- Einfacher einsetzbar und konfigurierbar als IPsec
- Geringere Skalierbarkeit hinsichtlich verschiedener Einsatzszenarien

Wireguard

- Einfaches, auf das wesentliche beschränktes Linux-basiertes VPN
- Festgelegte Kryptoalgorithmen, nicht konfigurierbar.
- Dafür einfach, schlank, übersichtlich und performant.

Secure Shell (SSH)

- sichere Alternative zu Telnet
- insbesondere genutzt für Remote-Administration
- ermöglichst wie auch SSL verschiedene Authentisierungsarten
  - Kennwörter
  - Zertifikate
- SSH-Tunneling möglich

=> Nachbarübung: KomSec Abkürzungen

Problem der Zertifizierung / Internet PKI:

- Sehr viele (>100) Zertifizierungsstellen (CAs) und Zwischen-und Root-CAs im Browser
- auch aus China, Iran, ... etc. Vertrauenswürdigkeit teilweise zweifelhaft
- Komprimittierte CA ermöglicht Man-in-the-Middle Angriffe auf TLS Verbindungen
  - Gefälschte Webseite wird aufgrund eines gültigen Zertifikats als echt erkannt
  - Z.B. www.hs-osnabrueck.de hat das Zertifikat von der DFN-CA (DFN = Deutsches Forschungsnetz) und keiner anderen CA. Das prüft der Browser aber nicht.

Idee des Certificate-Pinning (RFC 6844)

- Domain-Inhaber kann festlegen, welche CA Zertifikate für seine Domäne ausstellen darf (CA-Pinning) oder welcher öffentliche CA-Schlüssel hierzu dienen soll (Key-Pinning)
- CA/Key-Pin wird *im Browser* gespeichert => Problem: nur wenige Pins in aktuellen Browsern

Idee HTTP Public Key Pinning (HPKP, RFC 7469, 2015)

- Server sendet PIN im HTTP Header mit
- PIN korrespondiert zu öffentlichem Schlüssel, z.B. Hashwert
- Browser speichert PIN und akzeptiert nur Verbindungen, wenn Schlüssel im Zertifikat zu PIN passt.
- Probleme: Trust on first use, Neuinitialisierung bei Zertifikatswechsel

Drei Stufen von Zertifikaten:

- EV = Extended Validation, OV = Organization Validation, DV = Domain Validation

EV-Zertifikate

- Zertifikate zeigen exklusive Rechte der Organisation zur Benutzung einer Domain sowie die rechtliche, betriebliche und physische Existenz der Organisation
- Organisationen haben weltweit standardisiertes Identitätsüberprüfungsprogramm durchlaufen
- EV-CAs werden durch das CA/Browser Forums jährlich durch ein WebTrust-Audit zertifiziert
- EV-Zertifikate werden nur mit dedizierten privaten Schlüsseln besonders vertrauenswürdiger Zertifizierungsstellen ausgestellt => EV-Zertifikate sind am Aussteller erkennbar
- EV-Zertifikate nutzen besonders sichere Kryptoverfahren

Sicherheit von Let's Encrypt Zertifikaten (Domain Validation)?