Rechnernetze I

SoSe 2025

Roland Wismüller
Universität Siegen
roland.wismueller@uni-siegen.de
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 1. April 2025

Rechnernetze I

SoSe 2025

10 Netzwerksicherheit

Inhalt

- Sicherheitsanforderungen
- Sicherheitsprobleme der Internet-Protokolle
- Kryptographische Grundlagen
- Sicherheitsmechanismen für Protokolle
- Beispiele sicherer Protokolle
- Firewalls

- Peterson, Kap. 8.1, 8.2, 8.3.1, 8.3.3, 8.4
- CCNA, Kap. 11.2

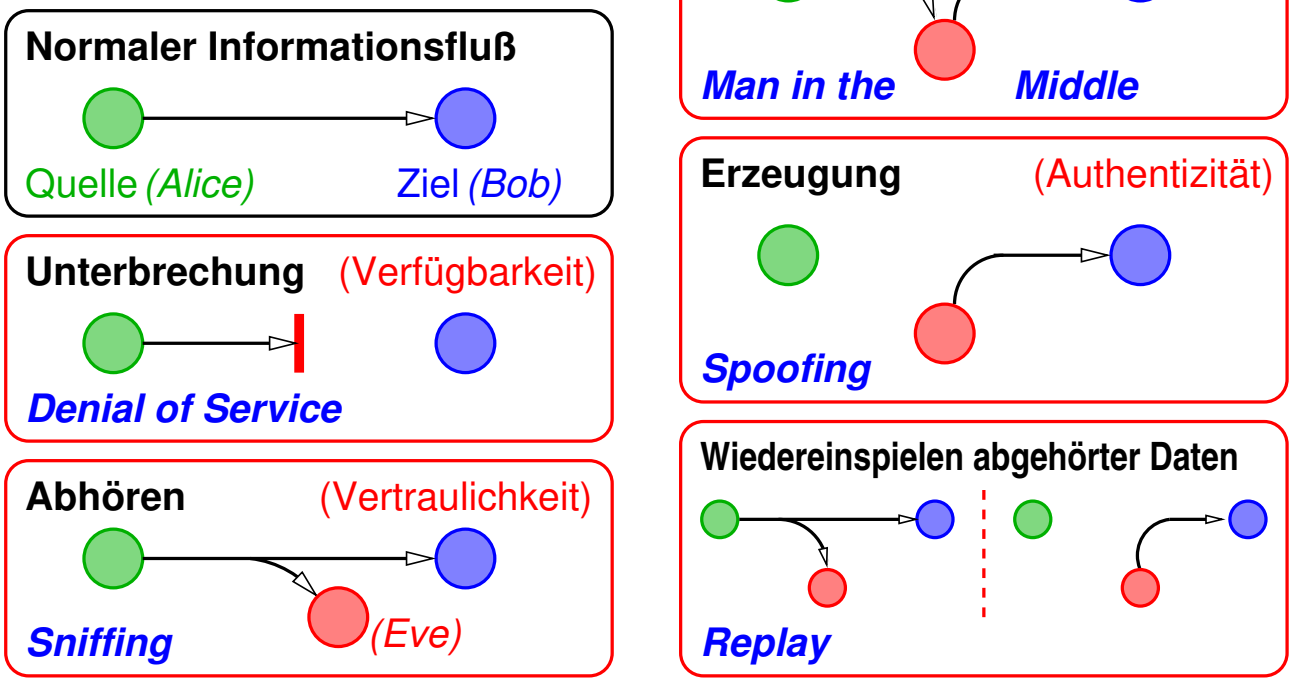
10.1 Sicherheitsanforderungen

- In Netzwerken wird persönliche / wertvolle / vertrauliche Information übermittelt
  - Information sollte nur Berechtigen bekannt werden!
  - Authentizität der Information?
- Wachsende Bedeutung der Netzwerksicherheit wegen
  - steigender Vernetzung
    - höheres Angriffspotential
  - neuer Einsatzgebiete
    - z.B. e-Business: elektronische Zahlung / Verträge

Allgemeine Sicherheitsanforderungen

- **(Informations-)Vertraulichkeit (*confidentiality*)**
  - Schutz vor unautorisierter Informationsgewinnung
- **(Daten-)Integrität (*integrity*)**
  - Schutz vor unautorisierter Veränderung von Daten
- **(Nachrichten-)Authentizität (*message authenticity*)**
  - Urheber der Daten kann korrekt identifiziert werden
- **Verbindlichkeit (*nonrepudiation*)**
  - Handlungen können nicht abgestritten werden
- **Verfügbarkeit (*availability*)** von Diensten
- **Anonymität** der Kommunikationspartner

Angriffe auf die Netzwerksicherheit



★★★

Konkret: Alice sendet eine Nachricht an Bob

- **Vertraulichkeit**: niemand außer Alice und Bob erfahren den Inhalt der Nachricht
- **Integrität**: Bob kann sich (nach entsprechender Prüfung!) sicher sein, daß die Nachricht während der Übertragung nicht (absichtlich) verfälscht wurde
- **Authentizität**: Bob kann sich (nach entsprechender Prüfung!) sicher sein, daß die Nachricht von Alice gesendet wurde
- **Verbindlichkeit**: Alice kann nicht bestreiten, die Nachricht verfaßt zu haben
  D.h. Bob kann Dritten gegenüber **beweisen**, daß die Nachricht von Alice gesendet wurde
- Im Folgenden: Beschränkung auf diese vier Anforderungen

10.2 Sicherheitsprobleme des Internets

Ein Problem des IP-Protokolls: IP-Spoofing

- Viele IP-basierte Protokolle vertrau(t)en der Absenderadresse
  - z.B. UNIX-Dienste rsh, rcp, rlogin*:
    - Festlegung von *Trusted Hosts*
    - Zugriff von *Trusted Host* aus auch ohne Paßwort
- Aber: Angreifer kann IP-Pakete mit beliebiger (falscher) Absenderadresse versenden
  - z.B. um vorzutäuschen, ein *Trusted Host* zu sein
- **Problem**: fehlende Authentifizierung der Pakete in IPv4

* Inzwischen nicht mehr in Verwendung!

Ein Problem des IP-Protokolls: IP-Spoofing ...

- IP-Spoofing ist Basis vieler anderer Angriffe
- Gegenmaßnahmen:
  - nicht auf Senderadresse vertrauen
  - Router-Konfiguration: *Source Address Validation*
    - Prüfen, ob Paket mit angegebener Senderadresse aus dem jeweiligen Subnetz kommen kann
  - IPsec (neuer Internet-Standard):
    sichere Authentifizierung des Senders

Anmerkungen zu Folie 336:

Ein Grundproblem der Internet-(Un)sicherheit ist die Tatsache, daß der Absenderadresse von IP-Paketen nicht vertraut werden kann (der Sender kann die Quell-IP-Adresse Aufwand beliebig angeben, dazu ist praktisch kein Aufwand notwendig).

Daher ist es wichtig, in Protokollen nicht auf die Authentizität der Absenderadresse zu vertrauen.

Eine weitere Maßnahme, um das Problem des *IP-Spoofings* zu adressieren, ist die Verwendung von *Source-Address-Validation* in Routern. Dabei prüft der Router, ob ein IP-Paket, das er über eine bestimmte Schnittstelle S empfangen hat, tatsächlich aus diesem Netz kommen kann. Zur Überprüfung bestimmt der Router über die Routing-Tabelle, über welche Schnittstelle S' er ein Paket an den Rechner mit der angegebenen **Quell**adresse senden würde. Nur wenn S' mit S übereinstimmt, leitet der Router das Paket weiter. Empfängt der Router beispielsweise über eine Schnittstelle, die direkt an ein LAN mit der Netzadresse 141.99.0.0/16 angeschlossen ist, ein Paket mit Quelladresse 131.159.79.3, so weiß er, daß dieses Paket eine falsche Quelladresse besitzt und verwirft es.

Eine sichere Authentifizierung von IP-Paketen ist mit *Secure IP* (IPsec) möglich.

Meist wird heute die Sicherheit aber nicht auf Ebene von IP realisiert, sondern in den darüberliegenden (Anwendungs-)Protokollen.

Ein Problem durch Programmierfehler: *Ping of death*

- Fehler in der Implementierung des `ping`-Kommandos unter Windows 95:
  - `ping -l 65510 my.computer.de` sendet ein fragmentiertes IP-Paket der Länge 65538
- Fehler in (alten Versionen) fast aller Betriebssysteme:
  - Pufferüberlauf im Betriebssystemkern beim Zusammenbau des Pakets
  - Absturz des Systems, *Reboot*, ...
- **Problem**: fehlende Validierung der Eingabe

Ein Problem des DNS: DNS-Spoofing

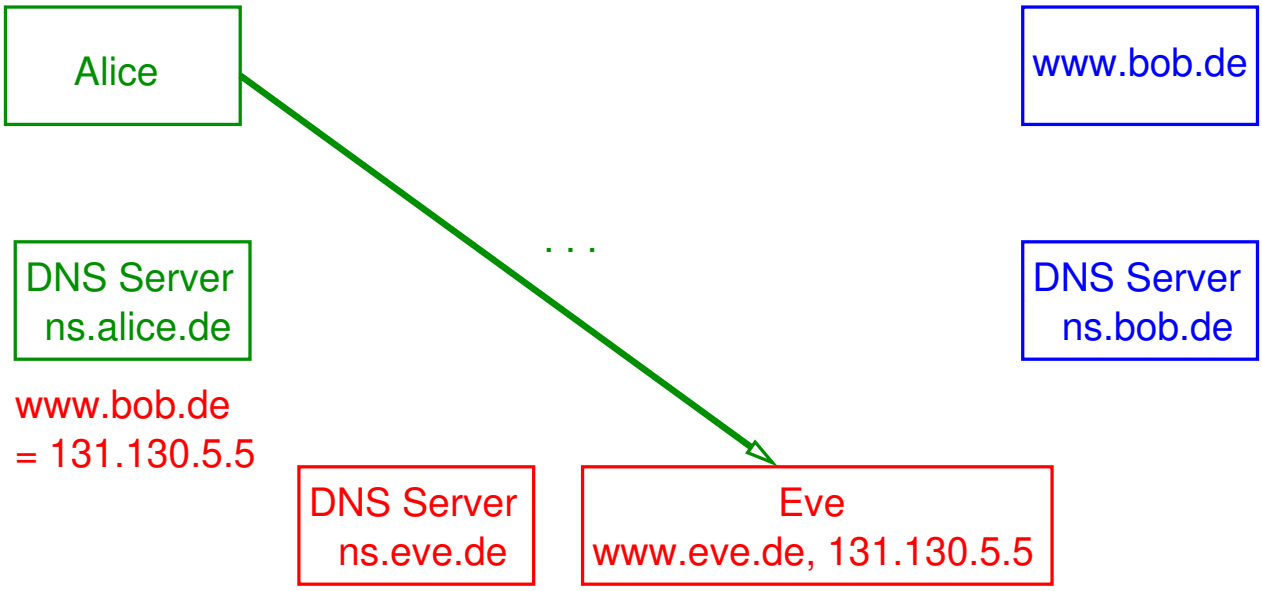
- Angreifer kann falsche Zuordnung zwischen Hostnamen und IP-Adresse in DNS-Servern installieren
  - Zugriffe auf diesen Host werden z.B. auf Rechner des Angreifers umgeleitet (= *Man-in-the-Middle* Attacke)
  - z.B. gefälschte Web-Sites, Ausspionieren von Kreditkarteninfo, Paßworten, ...
- **Problem:** keine Authentizierung
- Schadensbegrenzung: keine *recursive queries* zulassen
  - nur DNS-Cache eines Rechners kann infiziert werden
- Lösungen: TSIG*, DNSSEC (IETF Standards)

* Transaction SIGnature

(Animierte Folie)

Angriff auf DNS: DNS-*Spoofing*

3. Alice kontaktiert Eve und glaubt es wäre Bob!

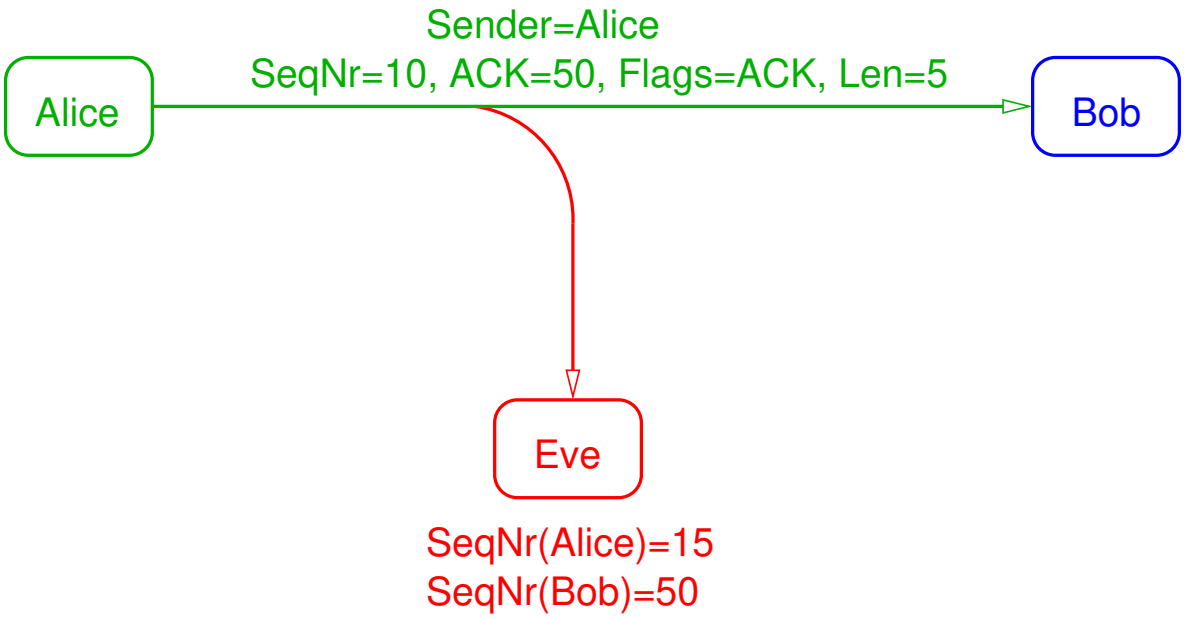


Ein Problem des TCP: Hijacking

- „Feindliche Übernahme“ einer offenen TCP-Verbindung
  - z.B. **nach erfolgter Authentifizierung von Alice**!
- Angriff:
  1. Senden eines gefälschten RST-Pakets (Verbindungsabbruch) an Alice
  2. Senden gefälschter Pakete an Bob, mit geschätzten oder abgehörten Sequenznummern
- Setzt i.d.R. Abhörmöglichkeit (z.B. Zugang zu lokalem Ethernet) voraus
  - Sequenznummern zählen die übertragenen Bytes

(Animierte Folie)

Ein Problem des TCP: Hijacking ...



Weitere problematische Protokolle

- UDP
  - **Problem**: fehlende Authentifizierung
  - „Fälschen“ von UDP-Paketen viel einfacher als bei TCP
  - Einige wichtige Dienste (DNS, NFS) basieren auf UDP
- ICMP
  - **Problem**: fehlende Authentifizierung
  - erlaubt Abbrechen, Behindern und Umleiten von Verbindungen
- ARP
  - **Problem**: keine Authentizierung
  - ARP ist zustandslos, akzeptiert Antwort von beliebigem Sender
  - Angreifer im lokalen Netz kann durch falsche ARP-Antwort Pakete zu sich umleiten (ARP Spoofing)

Weitere problematische Protokolle ...

- NFS (*Network File System*)
  - **Problem**: Authentifizierung nur über Hostname/User-ID
  - **Problem**: ungeschützte Übertragung von Dateiinhalten und -handles
- NIS (*Network Information System*)
  - Zentrale Paßwortdatei für Rechner eines Netzes
  - **Problem**: keine Authentifizierung des Servers
  - **Problem**: Zugriffskontrolle nur durch Domänen-Namen

Weitere problematische Protokolle ...

- `rsh`, `rcp`, `rlogin`, `telnet`, FTP (*File Transfer Protocol*)
  - **Problem**: Authentifizierung z.T. nur über Hostname/User-ID
  - **Problem**: ungeschützte Übertragung (incl. Paßworte!)
- HTTP (*HyperText Transport Protocol*)
  - **Problem**: keine Authentifizierung des Servers
  - **Problem**: ungeschützte Übertragung (auch Paßworte!)
- SMTP (*Simple Mail Transport Protocol*)
  - **Problem**: keine Authentifizierung des Absenders
  - **Problem**: Übertragung / Zwischenspeicherung im Klartext

★★

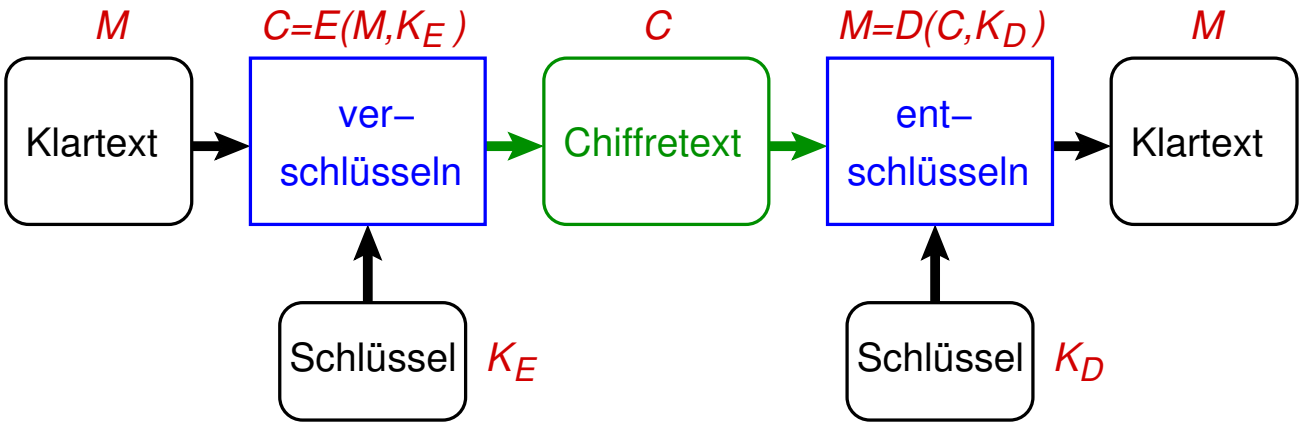
Fazit

- Die Standard-Internet-Protokolle (u.a. IP, TCP, DNS, ARP, NFS, HTTP, SMTP) erfüllen **keine** der in 10.1 genannten Sicherheitsanforderungen
- Hauptprobleme:
  - öffentliche Netze prinzipiell abhörbar
  - fehlende / unzureichende Authentifizierung
- Abhilfe:
  - sichere Protokolle in der Anwendungsschicht:
    - SSL/TLS (HTTPS, FTPS), S/MIME, PGP, SSH, ...
  - sicheres IP-Protokoll (IPsec, siehe Rechnernetze II)
- Basis: kryptographische Verfahren

10.3 Kryptographische Grundlagen

★

Grundprinzip der Verschlüsselung:



- Symmetrische Verschlüsselungsverfahren
  - $K_E = K_D = K$ = gemeinsamer geheimer Schlüssel
- Asymmetrische Verschlüsselungsverfahren
  - K_E = öffentlicher, K_D = privater Schlüssel

★

Anforderungen an Verschlüsselungsverfahren:

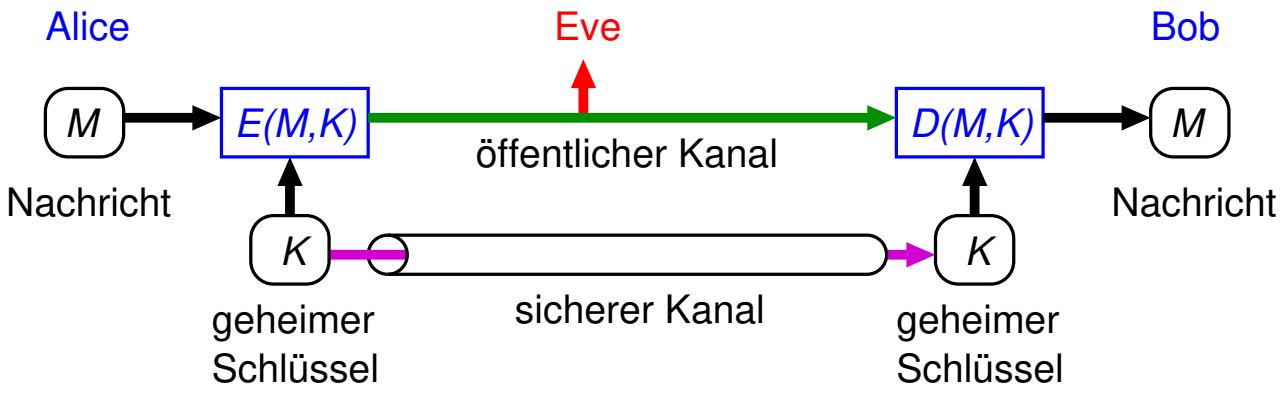
- Nur der Besitzer des geheimen bzw. privaten Schlüssels kann den Chiffretext entschlüsseln
- Sicherheit basiert nicht auf Geheimhaltung der Algorithmen

Mögliche Angriffe:

- Klartext-Angriff: Klartext + Chiffretext $\Rightarrow$ Schlüssel
- Im Idealfall: alle Schlüssel müssen durchprobiert werden
  - Schlüssel müssen lang genug sein!
- Bei asymmetrischen Verfahren auch effizientere Angriffe
  - Berechnung von K_D aus K_E ($\Rightarrow$ längere Schlüssel nötig)

★★★

10.3.1 Symmetrische Verschlüsselung



- Symmetrische Verschlüsselung ist sehr effizient realisierbar
- Schlüssel sind relativ kurz (heute typisch 128-256 Bit)
- Problem: Austausch des Schlüssels K

10.3.1 Symmetrische Verschlüsselung ...

Beispiele symmetrischer Verschlüsselungsverfahren:

- **DES**: veraltet, Schlüssel nur 56 Bit lang
- **Triple-DES**: veraltet, dreifache Anwendung von DES
  - effektive Schlüssellänge: 112 Bit
- **AES**: Nachfolger von DES, 128-256 Bit Schlüssel
  - vom amerikanischen NIST standardisiert
  - in praktisch allen sicheren Protokollen verwendet / unterstützt
- **IDEA**: 128 Bit Schlüssel
  - freies Verfahren, benutzt z.B. in PGP

10.3.1 Symmetrische Verschlüsselung ...

- Zwei verschiedene Arten symmetrischer Verfahren:
  - **Blockchiffren** ver-/entschlüsseln Blöcke von Zeichen mit einer fest vorgegebenen Größe (z.B. 128 Bit bei AES)
    - zu kleine Blöcke müssen aufgefüllt werden (*Padding*)
    - für Nachrichten beliebiger Größe: zusätzliche Betriebsmodi für den Umgang mit einer Folge von Blöcken
  - **Stromchiffren** ver-/entschlüsseln in einem Zeichenstrom jedes Zeichen einzeln
    - typisch: EXOR-Verknüpfung mit Pseudozufallsfolge
- Grundoperationen symmetrischer Blockchiffren:
  - Substitution: ersetze Bitgruppen systematisch durch andere Bitgruppen
  - Permutation: vertausche Bitgruppen nach festgelegtem Schema

Anmerkungen zu Folie 350:

- Für eine Stromchiffre wird ein kryptographischer Pseudozufallszahlengenerator (PRNG) benötigt, der den symmetrischen Schlüssel als Initialisierungswert verwendet. Dieser lässt sich durch spezielle Betriebsarten einer Blockchiffre realisieren. Die Ausgabe des PRNGs wird EXOR mit dem Datenstrom verknüpft.

  Zum Entschlüsseln verknüpft man den verschlüsselten Datenstrom nochmals mit der Ausgabe des PRNGs (mit dem selben symmetrischen Schlüssel als Initialisierungswert). Da $M \oplus X \oplus X = M$ erhält man dabei wieder den Klartext.

- Verwendet man eine **echte** Zufalls-Bitfolge als Schlüssel, kann man mathematisch beweisen, daß ohne Kenntnis des Schlüsselstroms der Chiffre-Text nicht entschlüsselt werden kann, da er nich von einer Zufalls-Bitfolge zu unterscheiden ist. Der Nachteil dieser sog. Vernam-Chiffre (*One Time Pad*) ist, daß der Schlüssel dieselbe Länge hat wie der zu übertragende Klartext, und daß er nur einmal verwendet werden darf.

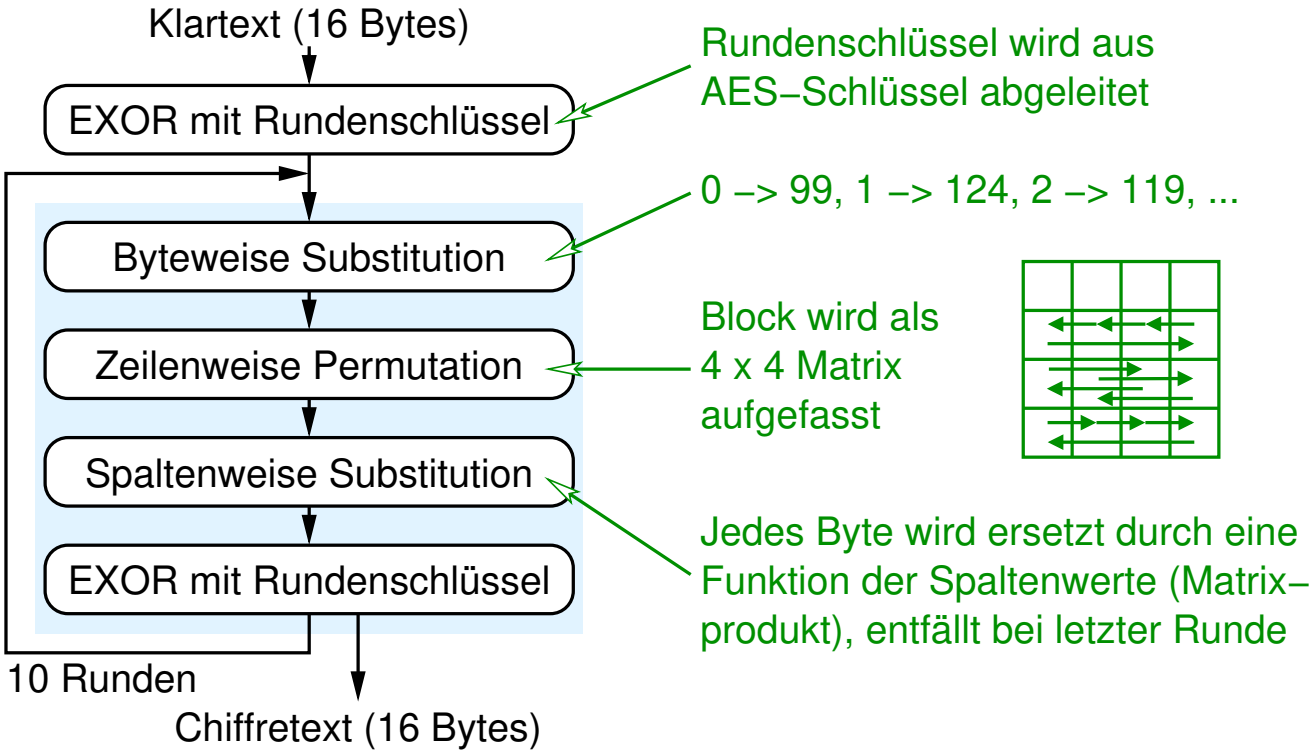
- Ein einfaches Beispiel für die Substitution ist die klassische Caesar-Chiffre, die jeden Buchstaben durch den im Alphabet 3 Plätzen weiter hinten liegenden Buchstaben ersetzt, z.B.:

  Klartext: `JULIUS CAESAR IMPERATOR`
  Chiffretext: `MXOLXV FDHVDU LPSHUDWRU`

- Ein einfaches Beispiel für eine Permutation wäre es, erst die Zeichen an den geraden Positionen und dann die an den ungeraden Positionen zu nehmen, z.B.:

  Klartext: `JULIUS CAESAR IMPERATOR`
  `J L U _ A S R I P R T R`
  `U I S C E A _ M E A O`
  Chiffretext: `JLU ASRIPRTRUISCEA MEAO`

Grobstrutkur von AES (Verschlüsselung)

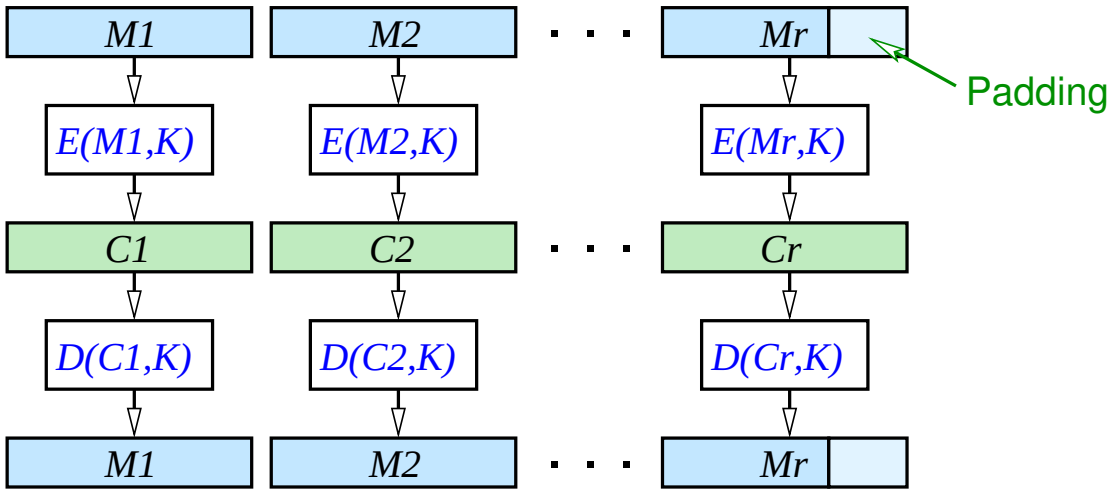


Anmerkungen zu Folie 351:

- Zur Entschlüsselung werden jeweils die inversen Substitutionen und Permutationen in umgekehrter Reihenfolge angewendet.
- Eine detaillierte Beschreibung von AES finden Sie in William Stallings: „*Cryptography and Network Security - Principles and Practices*“, Prentice Hall, 2003.

Betriebsarten von Blockchiffren: *Electronic Code Book* (ECB)

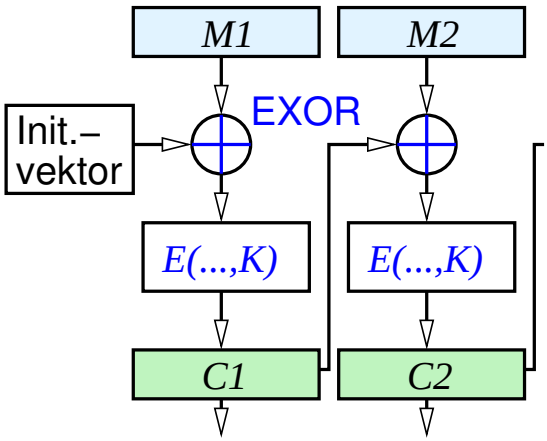
- Klartext wird in Blöcke (z.B. 128 Bit) aufgeteilt, ggf. mit Padding
- Blöcke werden unabhängig voneinander ver-/entschlüsselt



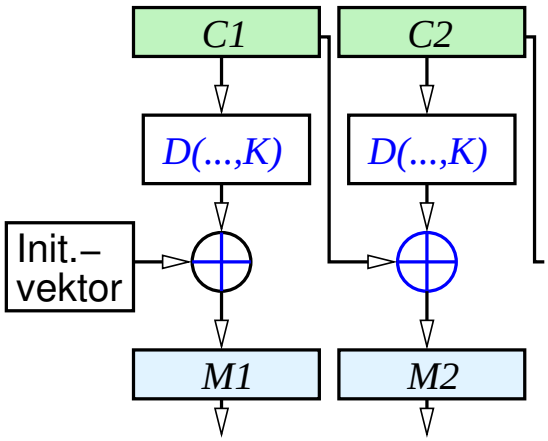
- Kein Schutz vor Löschung / Wiedereinspielung von Blöcken

Betriebsarten von Blockchiffren: *Cipher Block Chaining* (CBC)

Verschlüsselung



Entschlüsselung



- Gleiche Klartextblöcke $\Rightarrow$ verschiedene Chiffretextblöcke
- Fehlerfortpflanzung: Vorteil und Nachteil

Anmerkungen zu Folie 353:

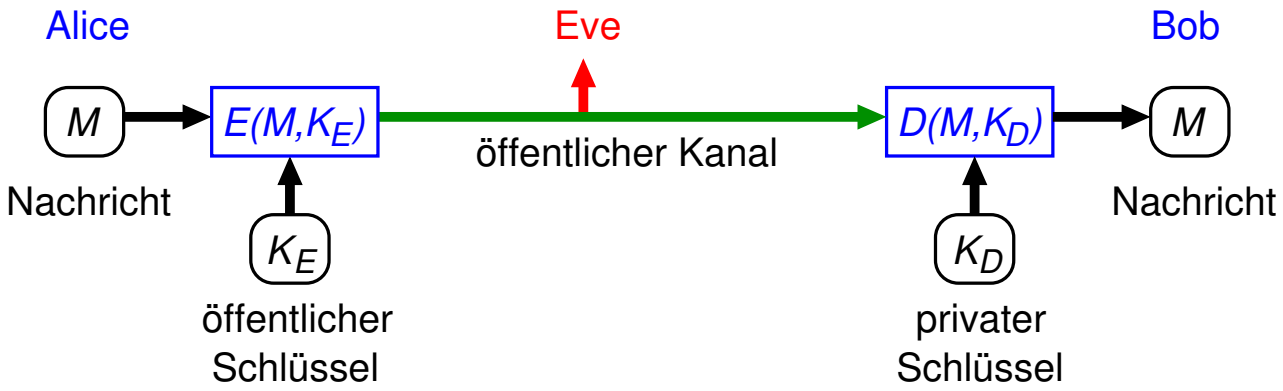
Es gibt weitere Betriebsmodi, insbesondere *Cipher Feedback Mode* (CFB), *Output Feedback Mode* (OFB) und *Counter Mode* (CTR), mit deren Hilfe aus einer Blockchiffre eine Stromchiffre gemacht werden kann.

WPA-2 beim WLAN verwendet z.B. AES im *Counter Mode* zur Verschlüsselung.

10.3 Kryptographische Grundlagen ...

★★★

10.3.2 Asymmetrische Verschlüsselung



- Bob berechnet K_E aus K_D und veröffentlicht K_E
  - Problem: Authentizität von K_E
- Weniger effizient als symmetrische Verfahren
- Längere Schlüssel nötig (heute typisch 2048-4096 Bit)

Basis asymmetrischer Verfahren:

- Einwegfunktionen (*on-way functions*):
  - Berechnung von $y = f(x)$ einfach
  - Berechnung von $x = f^{-1}(y)$ praktisch unmöglich
- Beispiele:
  - diskreter Logarithmus: $f(x) = a^x \bmod p$, p prim
    - Verwendung z.B. im Diffie-Hellman-Schlüsselaustausch
    - *Elliptic Curve Cryptography*: verwendet „Elliptische Kurven über endlichen Körpern“ als algebraische Struktur
      - erlaubt deutlich kürzere Schlüssel
  - Multiplikation großer Primzahlen vs. Faktorisierung
    - Verwendung z.B. in RSA
- Schwierigkeit der Berechnung der Umkehrfunktion nicht bewiesen

Diffie-Hellman-Schlüsselaustausch

- Frage: Wie können Alice und Bob über einen öffentlichen Kanal einen gemeinsamen geheimen Schlüssel K aushandeln?
- Gegeben sind öffentliche Elemente
  - q: Primzahl
  - a: primitive Wurzel von q ($a^n \bmod q$ durchläuft $1 \ldots q-1$)

Alice		Bob
wähle $X_A < q$		
berechne $Y_A = a^{X_A} \bmod q$	$Y_A \rightarrow$	
		wähle $X_B < q$
	$\leftarrow Y_B$	berechne $Y_B = a^{X_B} \bmod q$
berechne $K = (Y_B)^{X_A} \bmod q$		berechne $K = (Y_A)^{X_B} \bmod q$

- Problem: keine Authentifizierung!

RSA (Rivest, Shamir, Adleman)

- Schlüsselgenerierung
  - wähle große Primzahlen p und q, berechne **Modul** $n = p \cdot q$
    - Euler'sche Zahl $\varphi(n) = (p-1) \cdot (q-1)$
  - wähle e mit $1 < e < n$ und $ggT(\varphi(n), e) = 1$
  - wähle d so, daß $e \cdot d \bmod \varphi(n) = 1$
  - öffentlicher Schlüssel $K_E = (e, n)$
  - privater Schlüssel $K_D = (d, n)$
- Verschlüsseln und Entschlüsseln
  - Klartextblock M als binärcodierte Zahl auffassen: $M < n$
  - Verschlüsseln: $C = E(M, K_E) = M^e \bmod n$
  - Entschlüsseln: $M = D(C, K_D) = C^d \bmod n$

Anmerkungen zu Folie 357:

- Die Wahl der Primzahlen erfolgt typischerweise so, daß zunächst eine Zufallszahl bestimmt wird, die anschließend durch (probabilistische) Primzahltests, z.B. Rabin-Milner, überprüft wird. Ggf. wird eine neue Zufallszahl gewählt.
- Die Euler'sche Zahl $\varphi(n)$ ist die Anzahl der zu n teilerfremden Zahlen im Intervall $[1, n[$
  - für eine Primzahl p gilt $\varphi(p) = p - 1$
  - für das Produkt zweier Primzahlen p und q gilt $\varphi(p \cdot q) = (p-1) \cdot (q-1)$
- Die Berechnung von d (als multiplikative Inverse von e) erfolgt über einen erweiterten Euklidischen Algorithmus. Dieser testet gleichzeitig, ob $ggT(\varphi(n), e) = 1$.

RSA: Beispiel zum Nachrechnen

- Schlüsselerzeugung
  - $p = 3,\ q = 11$
  - $n = p \cdot q = 33,\ \varphi(n) = (p-1) \cdot (q-1) = 2 \cdot 10 = 20$
  - $e = 3$, damit $ggT(\varphi(n), e) = ggT(20, 3) = 1$
  - Wähle d so, daß $e \cdot d \equiv 1 \bmod \varphi(n),\ 3 \cdot d \equiv 1 \bmod 20,\ d = 7$
  - Öffentlicher Schlüssel $K_E = (3, 33)$, privater $K_D = (7, 33)$
- Verschlüsseln und Entschlüsseln
  - Klartextnachricht $M = 5$
  - $C = E(M, K_E) = 5^3 \bmod 33 = 125 \bmod 33 = 26$
  - $D(C, K_E) = 26^7 \bmod 33 = 8031810176 \bmod 33 = 5 = M$

Anmerkungen zu Folie 358 (zur Korrektheit des RSA-Verfahrens):

Mathematische Grundlagen:

- Definition der Restklassenarithmetik:
  - $a \equiv b \bmod n \ \Leftrightarrow\ a \bmod n = b \ \Leftrightarrow\ \exists k : a = b + k \cdot n$
- Satz von Euler:
  - $ggT(M, n) = 1 \ \Rightarrow\ M^{\varphi(n)} \equiv 1 \bmod n$

Zu zeigen: $D(E(M, K_E), K_D) = M$

Beweis

- Nach Definition von E und D: $D(E(M, K_E), K_D) = M^{e \cdot d} \bmod n$
- Da $e \cdot d \equiv 1 \bmod \varphi(n)$, gilt $\exists k : e \cdot d = k \cdot \varphi(n) + 1$
  - damit: $M^{e \cdot d} \bmod n = M^{k \cdot \varphi(n)+1} \bmod n = M \cdot M^{k \cdot \varphi(n)} \bmod n$
- Falls $ggT(M, n) = 1$ gilt mit dem Satz von Euler:
  - $M \cdot M^{k \cdot \varphi(n)} \bmod n = M$

- Sonst: M ist entweder ein Vielfaches von p oder ein Vielfaches von q (beides zusammen geht nicht, da $M < n = p \cdot q$)
  - falls M ein Vielfaches von p ist:
    - dann $ggT(M, q) = 1$ und $\exists i : M = i \cdot p$
    - mit Satz von Euler: $M^{\varphi(q)} \equiv 1 \bmod q$
    - damit auch $M^{k \cdot \varphi(p) \cdot \varphi(q)} = M^{k \cdot \varphi(n)} \equiv 1 \bmod q$
    - also $\exists k : M^{k \cdot \varphi(n)} = 1 + l \cdot q$
    - damit ist $M \cdot M^{k \cdot \varphi(n)} = M \cdot (1 + l \cdot q) = M + M \cdot l \cdot q = M + i \cdot p \cdot l \cdot q = M + i \cdot l \cdot n$
    - das bedeutet $M \cdot M^{k \cdot \varphi(n)} \bmod n = M$
  - falls M ein Vielfaches von q ist: analog

10.3 Kryptographische Grundlagen ...

★★★

10.3.3 Kryptographische Hashfunktionen (*Message Digest*)

- Analog einer normalen Hashfunktion:
  - Nachricht wird auf einen Wert fester Größe abgebildet
- Zusätzliche Eigenschaft: **Kollisionsresistenz**
  - zu Nachricht *x* kann (in vernünftiger Zeit) keine andere Nachricht *y* mit gleichem Hashwert gefunden werden
- Einsatz zur Sicherung der Integrität
  - „kryptographische Prüfsumme“
- Beispiele
  - MD5 (*Message Digest, Version 5*): 128 Bit Hashwert, unsicher
  - SHA-1 (*Secure Hash Algorithm 1*): 160 Bit Hashwert, unsicher
  - SHA-2 / SHA-3: 224 - 512 Bit Hashwert

Anmerkungen zu Folie 359:

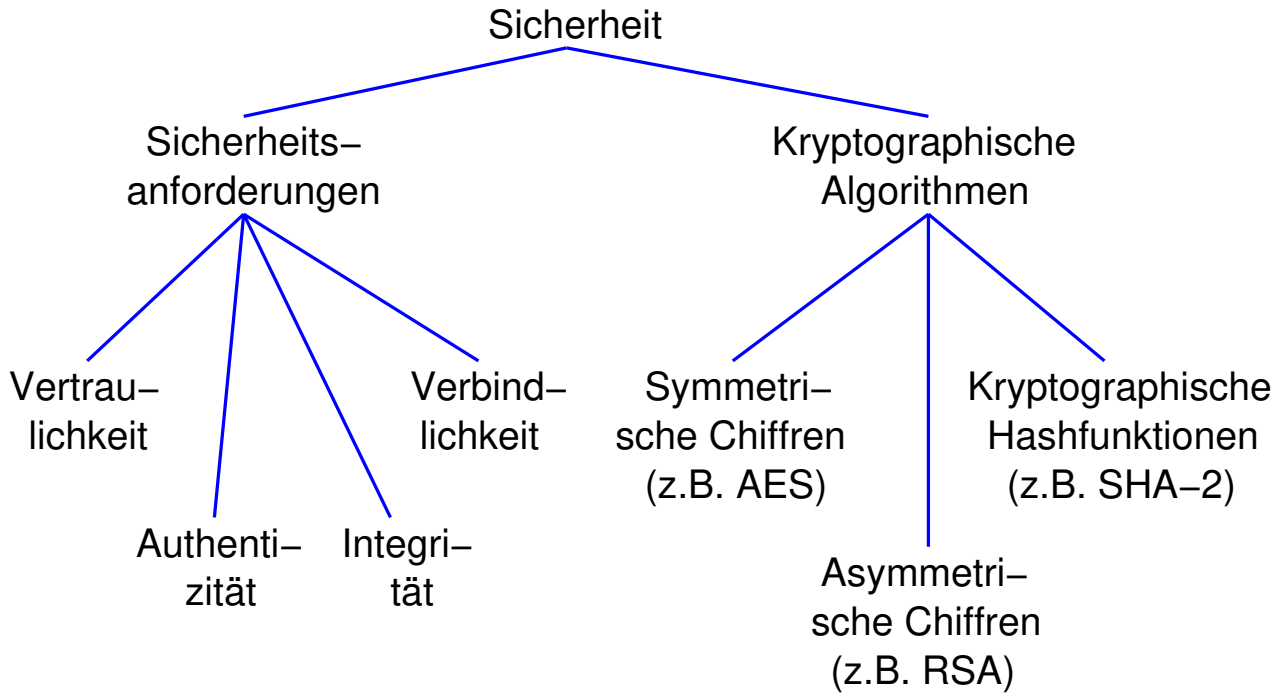
Genauer unterscheidet man zwischen „schwacher“ und „starker“ Kollisionsresistenz.

- Die hier gegebene Definition ist die für die schwache Kollisionsresistenz.
- Starke Kollisionsresistenz ist gegeben, wenn es (praktisch) unmöglich ist, zwei Nachrichten x und y zu finden, die denselben Hashwert produzieren.
- Um starke Kollisionsresistenz zu erreichen, sind in der Regel doppelt so lange Hashwerte erforderlich (siehe „Geburtstagsparadoxon“).

10.3 Kryptographische Grundlagen ...

★★★

10.3.4 Zusammenfassung



10.3.4 Zusammenfassung ...

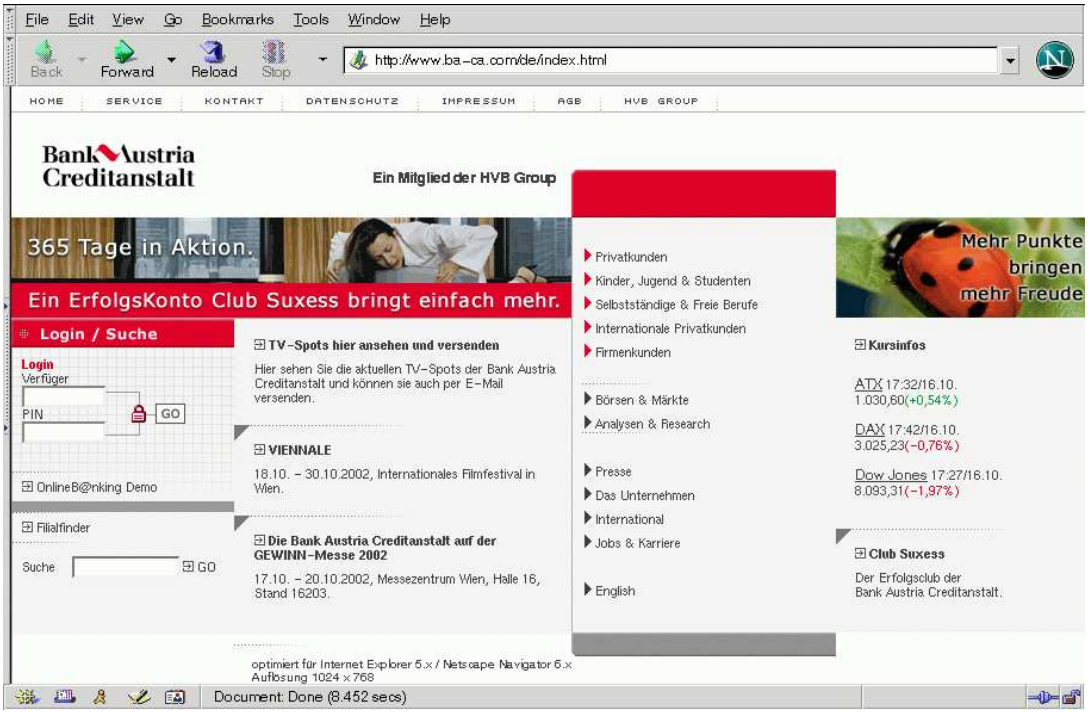
★★★

Was leistet die reine Verschlüsselung von Nachrichten?

- Vertraulichkeit: **ja**
- Integrität: **bedingt**
  - nur, wenn Klartext genügend Redundanz aufweist
  - ⇒ Verwendung von *Message Digests*
- Nachrichtenauthentizität:
  - **nein** bei asymmetrischen Verfahren: K_E öffentlich!
  - **bedingt** bei symmetrischer Verschlüsselung
    - nur mit gesicherter Integrität und Schutz vor *Replay*
- Verbindlichkeit: **nein**
- Schutz vor Replay: **nein**
  - ⇒ Transaktionszähler im Klartext + Integrität sichern

10.3.4 Zusammenfassung ...

Verschlüsselung muss auch richtig angewendet werden!



Die Seite ist von 2003 und wurde nach ca 3 Jahren(!) geändert

10.4 Sicherheitsmechanismen

- Kryptographische Algorithmen sind nur Bausteine für die Netzwerksicherheit
- Zusätzlich benötigt: Mechanismen und Protokolle
- Einige Sicherheitsaufgaben:
  - Authentifizierung
    - von Kommunikationspartnern
      - „wer ist mein Gegenüber?“
    - von Nachrichten
      - „stammt die Nachricht wirklich vom Absender?“
  - Sicherung der Integrität von Nachrichten
  - Verbindlichkeit
  - Verteilung öffentlicher Schlüssel

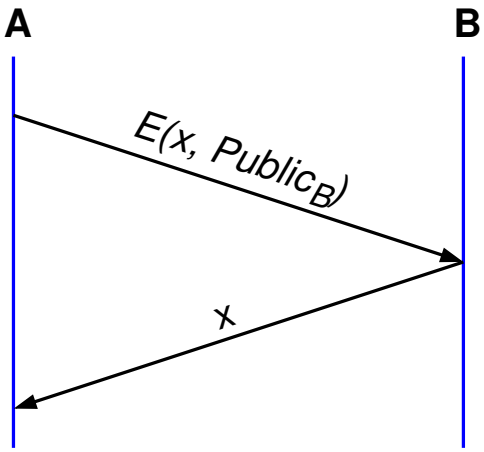
10.4 Sicherheitsmechanismen ...

Partner-Authentifizierung

- Kommunikationspartner authentifizieren sich gegenseitig
  - Beispiel: File-Server
    - Server authentifiziert den Client
      zur Prüfung der Schreib-/Leserechte
    - Client authentifiziert den Server
      zum Lesen/Schreiben sensibler Daten
- Manchmal auch nur einseitige Authentifizierung
  - Beispiel: WWW-Server
    - Client authentifiziert den Server
      zur Übertragung wichtiger / vertraulicher Daten

★★

Partner-Authentifizierung über asymmetrische Chiffre



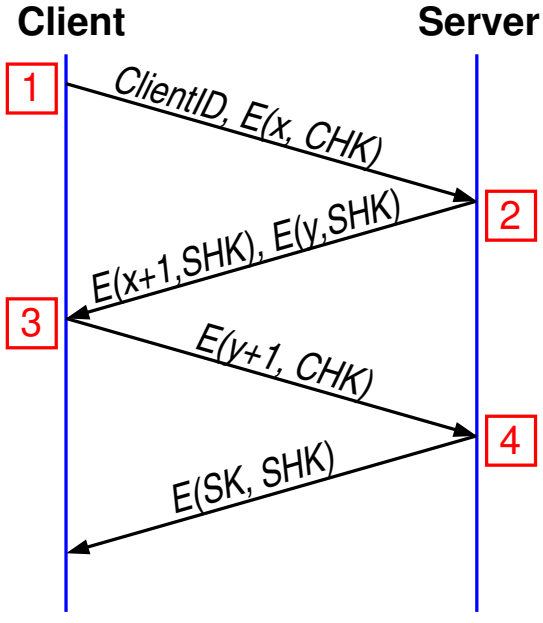
- Einseitige Authentifizierung von *B*
  - ggf. authentifiziert sich *A* ebenso (≈ 3-Wege-Handshake)
- $Public_B$ nicht zum Verschlüsseln verwenden!

★★

Partner-Authentifizierung mit Drei-Wege-Handshake

- Server kennt Schlüssel des Clients (z.B. Paßwort bei login)

1. Client sendet *ClientID* und verschlüsselte Zufallszahl x (CHK: *Client Handshake Key*)
2. Server sucht den zu *ClientID* gehörigen Schlüssel *SHK*, sendet *x+1* und Zufallszahl *y*
3. Server ist authentifiziert (*x+1*)
4. Client ist authentifiziert (*y+1*), Server sendet *Session Key SK* für weitere Kommunikation



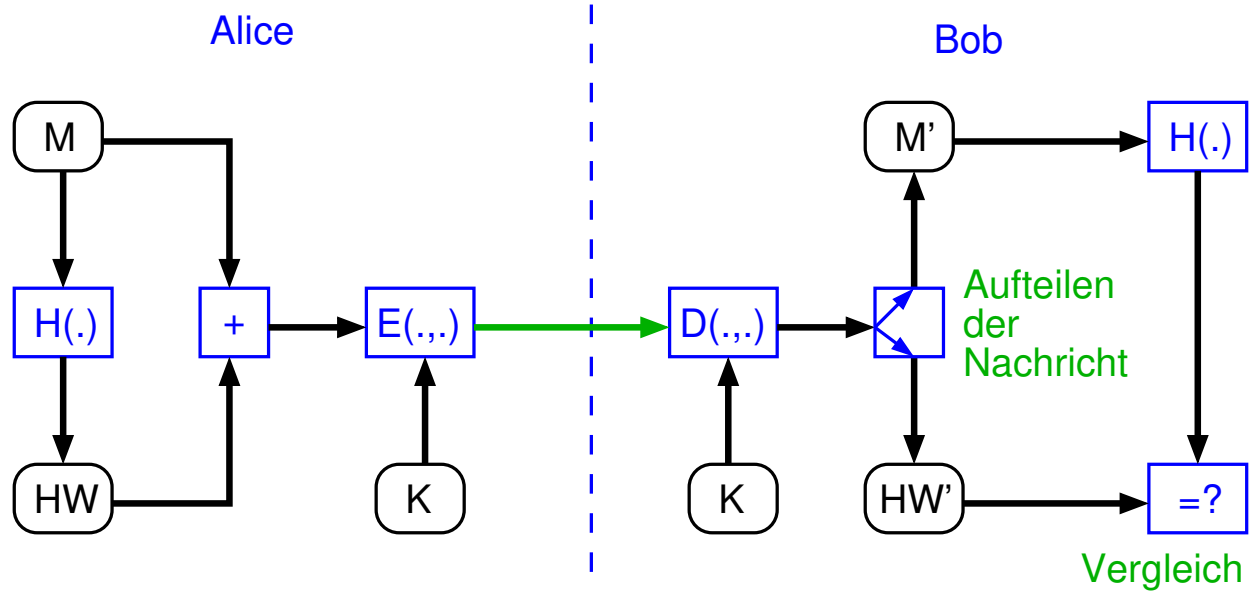
★★

Sicherung der Nachrichtenintegrität und -authentizität

- Integrität: Kein Dritter soll Nachricht verfälschen können
  - setzt sinnvollerweise Nachrichten-Authentizität voraus
- Bei Übertragung mit symmetrischer Verschlüsselung:
  - kryptographischen Hashwert *H(M)* an Klartext *M* anfügen und verschlüsseln
  - bei Modifikation des Chiffretexts paßt die Nachricht nicht mehr zum Hashwert
  - kein Angreifer kann neuen Hashwert berechnen / verschlüsseln
  - Nachrichten-Authentizität (bis auf Replay) durch symmetrische Chiffre sichergestellt
    - Replay-Schutz: Transaktionszähler / Zeitstempel in *M*

Anmerkungen zu Folie 367:

Der Ablauf einer Kommunikation (einschließlich der Prüfung, die der Empfänger durchführen muß) sieht dann so aus:



★★

Sicherung der Nachrichtenintegrität und -authentizität ...

- Bei asymmetrischer Verschlüsselung:
  - Hash-Wert allein nützt nichts, da Nachrichten-Authentizität nicht sichergestellt ist
- Bei unverschlüsselter Übertragung (oft sind Daten nicht vertraulich, aber ihre Integrität wichtig):
  - Hash-Wert stellt Integrität nicht sicher, da jeder nach einer Modifikation der Nachricht den neuen Hash-Wert berechnen kann
- Lösungen:
  - kryptographischer Hashwert mit geheimem Schlüssel
  - digitale Signatur

★★★

Hashwert mit geheimem Schlüssel

- Einbeziehen eines (gemeinsamen) geheimen Schlüssels *K* in den Hashwert:
  - füge *H(M + K)* an Nachricht M an (*+* = Konkatenation)
- Sichert auch Nachrichten-Authentizität (bis auf Replay)
  - kein Dritter kann *H(M + K)* korrekt berechnen
  - Replay-Schutz: Transaktionszähler / Zeitstempel in *M*
- Sichert nicht Verbindlichkeit
  - Empfänger kann *H(M + K)* berechnen
- Beispiel: *HMAC-SHA-256*

Anmerkungen zu Folie 369:

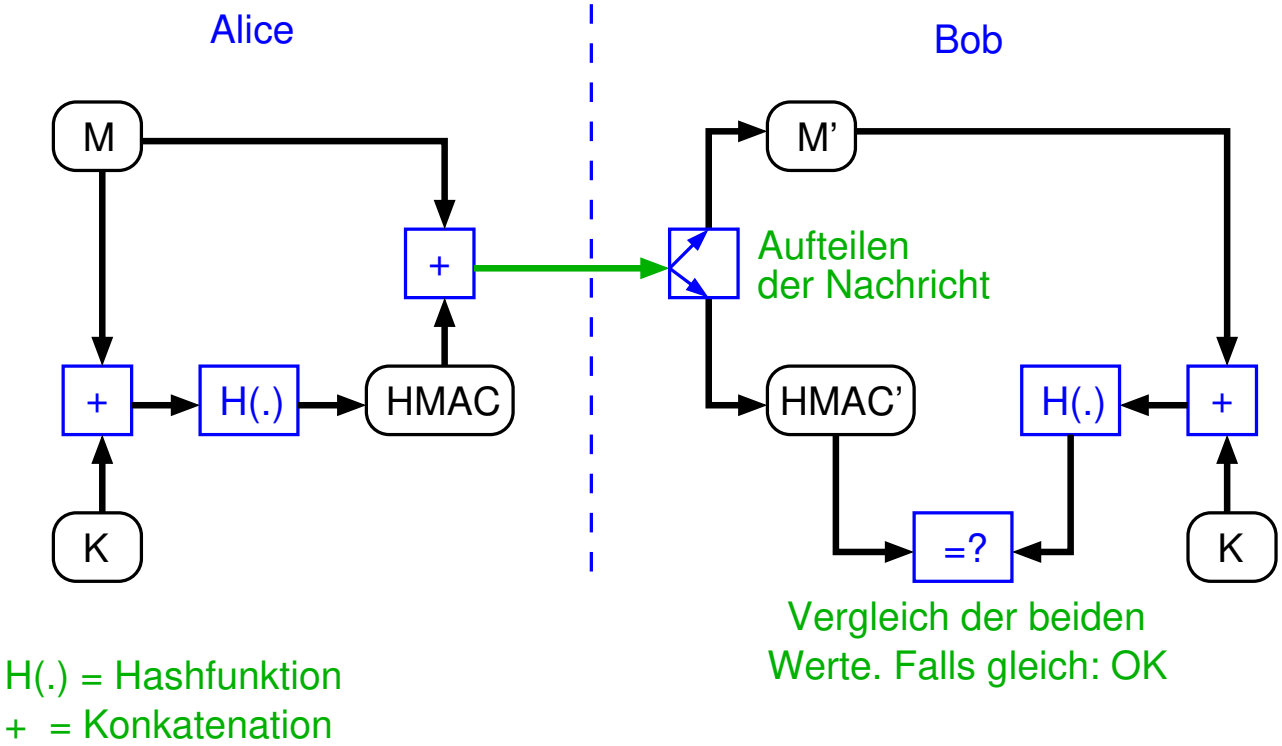
Tatsächlich wird der Schlüssel nicht einfach an die Nachricht angefügt, sondern komplexer mit dieser verknüpft, um Kryptoanalyse zu erschweren:

$$HMAC(M, K) = H(K \oplus \texttt{opad} + H(K \oplus \texttt{ipad} + M))$$

ipad = $\mathbf{0011\ 0110\ \ldots\ 0011\ 0110_2}$
opad = $\mathbf{0101\ 1010\ \ldots\ 0101\ 1010_2}$

Dabei ist $\oplus$ die bitweise Exklusiv-Oder-Verknüpfung und $+$ die Konkatenation.

Der Ablauf einer Kommunikation (einschließlich der Prüfung, die der Empfänger durchführen muß) sieht dann so aus:



★★★

Digitale Signatur mit asymmetrischer Chiffre

- Sender *A* sendet *M* und *E(M, Private$_A$)* an Empfänger *B*
- *B* entschlüsselt mit *Public$_A$* und prüft, ob Ergebnis gleich *M* ist
- Problem: asymmetrische Verschlüsselung ist langsam
- Daher: Kombination mit kryptographisher Hashfunktion
  - digitale Signatur von *A* auf *M* dann: *E(H(M), Private$_A$)*
- Digitale Signatur sichert Integrität, Nachrichten-Authentizität (bis auf Replay) und Verbindlichkeit
  - nur *A* besitzt *Private$_A$*
  - Replay-Schutz: Transaktionszähler in *M*

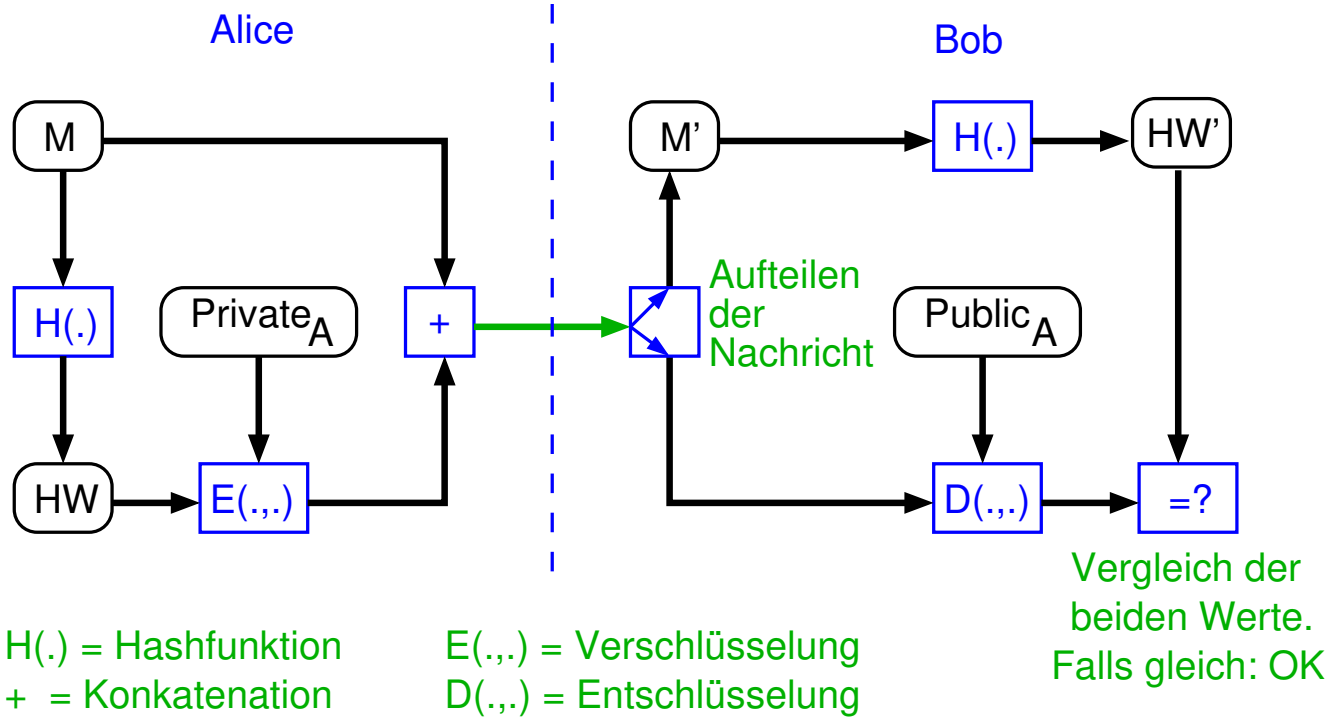
Anmerkungen zu Folie 370:

Bei der Signatur mit Hilfe einer kryptographishen Hashfunktion ist die exakte Vorgehensweise wie folgt:

- Alice sendet *M* und *S* := *E(H(M), Private$_A$)* an Bob
- Bob entschlüsselt *S* und erhält einen „Soll"-Hashwert *H'* := *D(S, Public$_A$)*
- Bob berechnet nun selbst den Hashwert über die Nachricht *M*:
  H := *H(M)*
- Nun vergleicht Bob *H'* mit *H*. Sind beide gleich, akzeptiert er die Signatur.

Dieses Signaturschema ist nur sicher, wenn die verwendete Hashfunktion kollisionsresistent ist. Sonst könnte ein Angreifer zu dem Hashwert
H' := *D(S, Public$_A$)* einer Signatur *S* eine neue Nachricht *M'* berechnen, die dieselbe Signatur hat, und somit behaupten, Alice hätte *M'* signiert.

Der Ablauf der Erstellung und Prüfung einer digitalen Signatur sieht dann so aus:



10.4 Sicherheitsmechanismen ...

Verteilung öffentlicher Schlüssel

- Problem: Übertragung des öffentlichen Schlüssels $Public_A$ von *A* zu *B*
- Woher weiß *B*, daß $Public_A$ authentisch ist?
  - zur Authentifizierung bräuchte *B* den Schlüssel von *A* ...
- Lösungen:
  - Übertragung über andere Medien (persönlich, Post, ...)
  - Zertifikate (*Certificates*)

★★★

Zertifikat

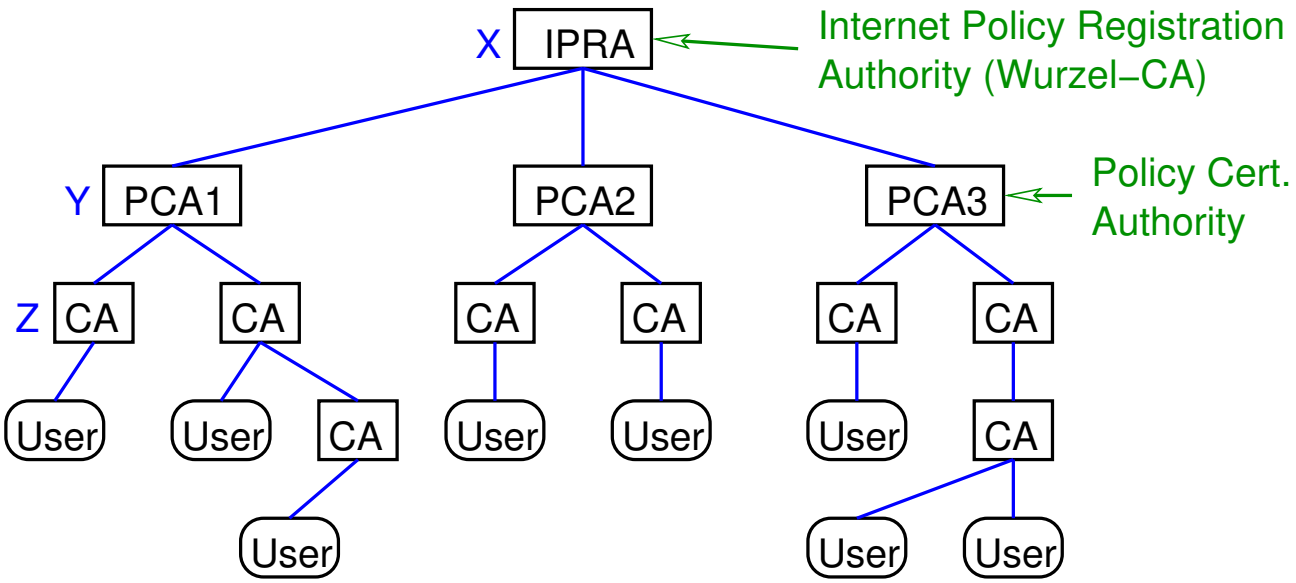
> *Ich bestätige, daß der in diesem Dokument stehende öffentliche Schlüssel dem angegebenen Eigentümer gehört.*
>
> Gezeichnet: $\mathcal{CA}$

- Die Zertifizierungsstelle (CA, *Certification Authority*) beglaubigt die Zuordnung zwischen einem öffentlichem Schlüssel und seinem Besitzer
  - durch digitale Signatur
- Nur noch der öffentliche Schlüssel der CA muß separat veröffentlicht werden

★

Zertifizierungshierarchie (z.B. bei HTTPS)

- Vertrauenskette: *X* zertifiziert, daß Schlüssel von *Y* authentisch ist, *Y* zertifiziert Schlüssel von *Z*, ...



X.509 Zertifikate

- X.509: wichtiger Standard für Zertifikate
- Komponenten des Zertifikats:
  - Name der Person/Institution oder eines Rechners
    - ggf. auch Email-Adresse oder Domain-Name
  - öffentlicher Schlüssel der Person/Institution bzw. des Rechners
  - Name der CA
  - Ablaufdatum des Zertifikats (optional)
  - digitale Signatur der CA
    - über alle obigen Felder

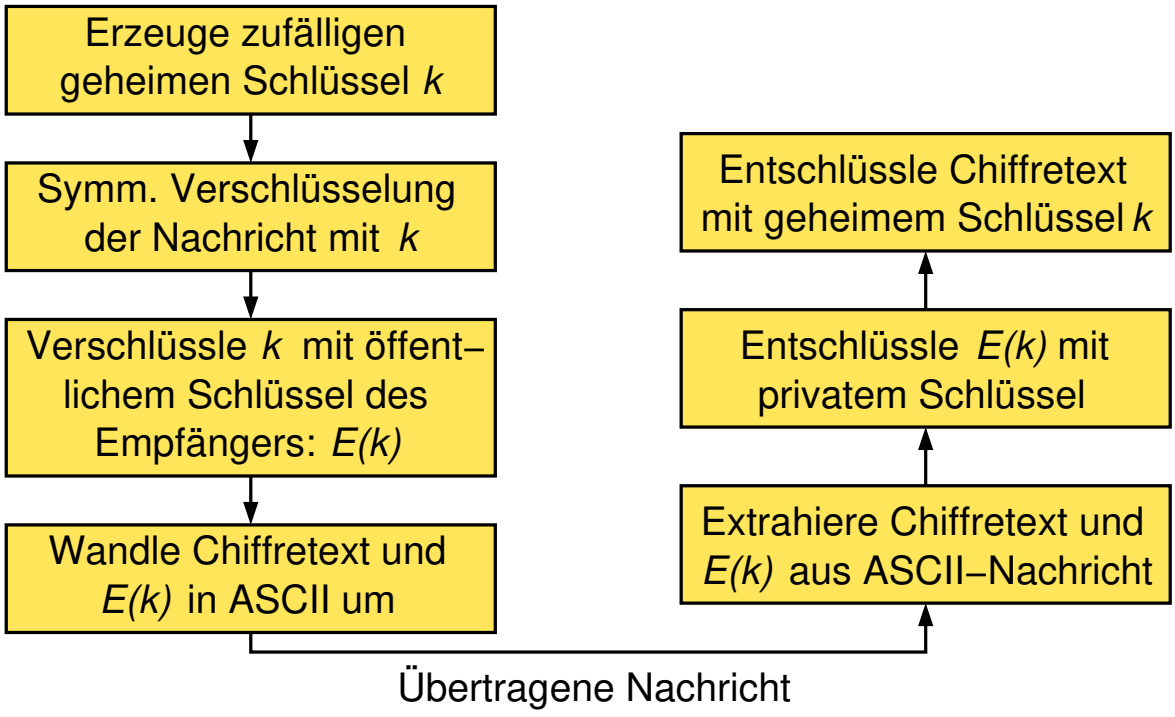
Invalidierung von Zertifikaten

- Zertifikate können beliebig kopiert und verbreitet werden
- Authentizität wird durch ein Zertifikat nur in Verbindung mit dem Besitz des privaten Schlüssels belegt
- Falls privater Schlüssel ausgespäht wurde:
  - Widerruf des Zertifikats nötig
- Einfache Möglichkeit:
  - *Certificate Revocation List* (CRL)
    Liste widerrufener Zertifikate, signiert von CA
  - Ablaufdatum begrenzt Länge der Liste

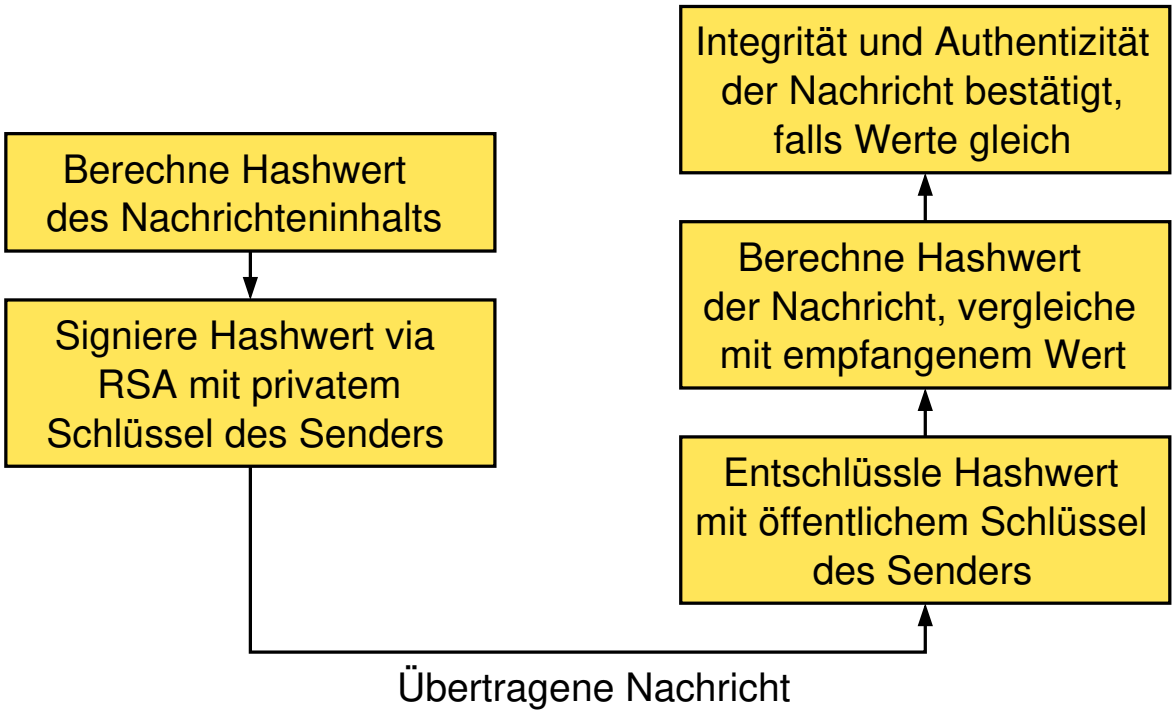
PGP (Pretty Good Privacy)

- Realisiert Vertraulichkeit, Integrität, Authentifizierung und Verbindlichkeit für Email
- Mechanismen: Verschlüsselung und digitale Signatur
  - einzeln oder kombiniert verwendbar
- Keine Zertifizierungsstellen bzw. –hierarchie
  - PGP-Benutzer zertifizieren die öffentlichen Schlüssel gegenseitig
    - mehrere Zertifikate möglich (höheres Vertrauen)
  - Vertrauensstufe des Schlüssels wird bei Email-Empfang angezeigt

PGP: Verschlüsselte Übertragung von Emails



PGP: Signierung von Emails



★★

TLS (*Transport Layer Security*)

- Motivation: Sicherheit im WWW, z.B. für Kreditkartenzahlung
  - Vertraulichkeit (der Kreditkarteninformation)
  - Authentizität (des WWW-Servers)
  - Integrität (der Bestelldaten)
  - (Verbindlichkeit wird von TLS nicht gewährleistet)
- TLS ist ein Internet-Standard der IETF
  - Basis: ältere Realisierung SSL (*Secure Socket Layer*)
- TLS ist die Grundlage vieler sicherer Protokolle im WWW:
  - z.B. HTTPS, FTPS, ...
  - realisiert durch eine zusätzliche Schicht

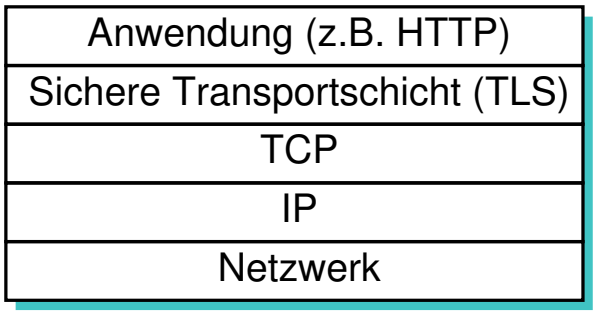
Anmerkungen zu Folie 379:

SFTP ist eine weitere sichere Variante des Dateitransfer-Protokolls FTP, das nicht TLS, sondern das SSH-Protokoll als Basis für die Sicherheit verwendet.

379-1

10.5 Beispiele sicherer Protokolle ...

TLS: sichere Transportschicht



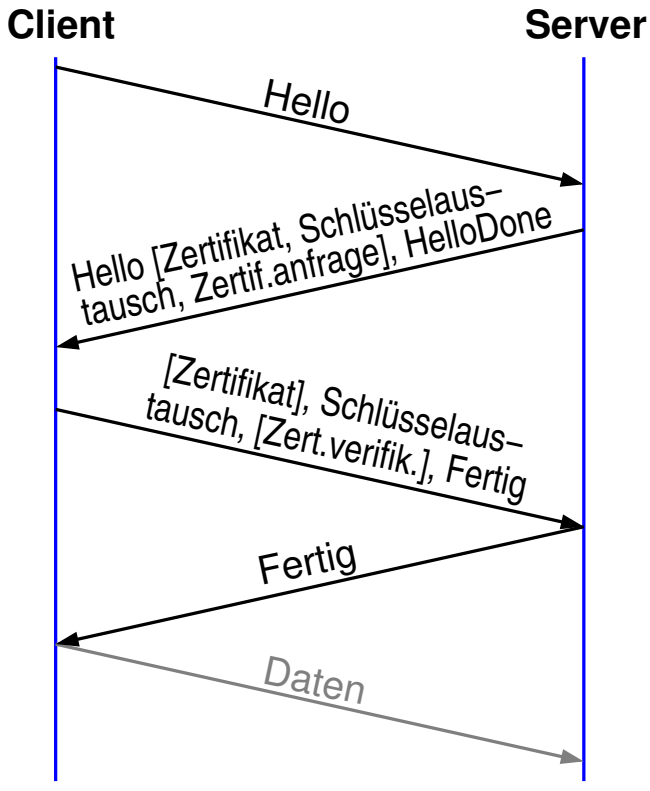
- Vorteil: unveränderte Anwendungsprotokolle
- Spezielle Ports, z.B. 443 für HTTPS
  - TLS gibt Daten von TCP an HTTP-Protokoll weiter (bzw. umgekehrt)

Wichtige TLS Teil-Protokolle:

- Handshake-Protokoll
  - beim Verbindungsaufbau
  - Aushandeln der kryptographischen Parameter:
    - Verfahren, Schlüssellänge, Sitzungsschlüssel, Zertifikate, Kompression
- Record-Protokoll
  - für die eigentlichen Daten
  - Fragmentierung, Kompression, Message Digests, Verschlüsselung, Transport (TCP)

TLS Handshake-Protokoll

- Bis zu 12 Nachrichten
- Aushandeln der kryptographischen Parameter notwendigerweise unverschlüsselt
- Man-in-the-Middle kann schwache Verschlüsselung aushandeln
- Anwendungen müssen auf Mindestanforderungen bestehen, ggf. Verbindungsabbruch

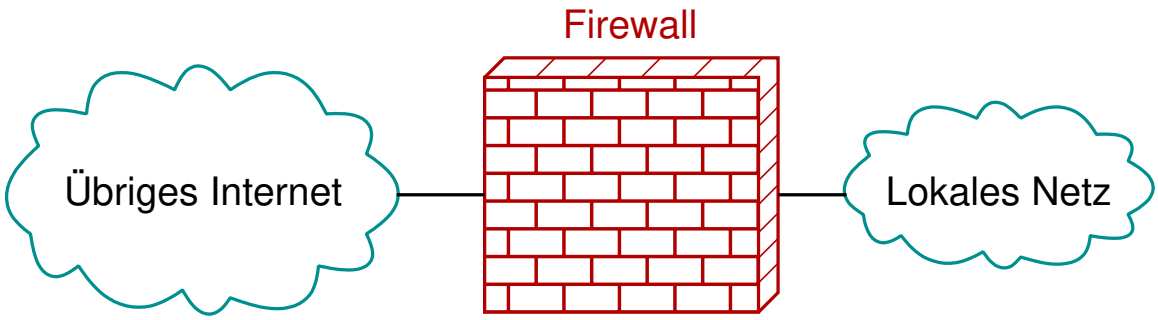


Anmerkungen zu Folie 382:

- Die Hello-Nachrichten dienen zum Aushandeln von Parametern:
  - Version, Zufallszahl, SessionID, Verfahren für Schlüsselaustausch, Verschlüsselung und Kompression, ...
- Für den Schlüsselaustausch können unterschiedliche Verfahren verwendet werden, die dann auch Anzahl und Inhalt der Nachrichten bestimmen.
- Ein Client-Zertifikat wird nur auf Anfrage durch Server übertragen
- Die Zertifikatsverifikations-Nachricht enthält die digitale Signatur aller bisher gesendeten und empfangenen Nachrichten
- Der Server verifiziert sein Zertifikat beim Schlüsselaustausch durch eine Signatur über die Zufallszahlen aus den Hello-Nachrichten und die Server-Schlüsselaustausch-Parameter

10.6 Firewalls

★★



- **Firewall**: Router mit Filterfunktion
  - kann bestimmte Pakete ausfiltern (verwerfen) und somit Zugriff auf bestimmte Hosts / Dienste unterbinden
  - wäre i.W. überflüssig, wenn alle Dienste sicher wären!
- Zwei Typen:
  - Filter-basierte Firewalls
  - Proxy-basierte Firewalls

★★

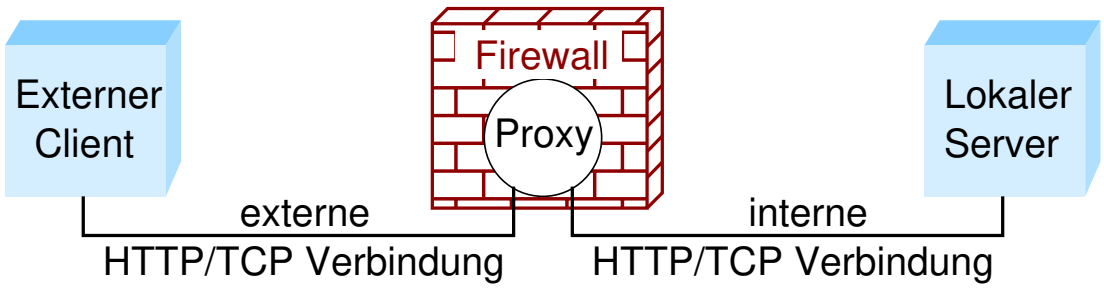
Filter-basierte Firewalls

- Filtern nur aufgrund von Quell- und Ziel-IP-Adressen, Quell- und Ziel-Ports, übertragenem Protokoll, sowie ggf. der TCP-Flags
- Filterregeln z.B.
  - `deny tcp 192.12.0.0/16 host 128.7.6.5 eq 80`
  - `permit tcp any host 128.7.6.5 eq 25`
- Frage: alles erlaubt, was nicht verboten ist, oder umgekehrt?
- Statische oder dynamische Regeln
  - z.B. FTP: neue Ports für jede übertragene Datei
- „Level-4-Switch“: Firewall kennt Transport-Protokolle

★

Proxy-basierte Firewalls

- Proxy: Mittler zwischen Client und Server
  - für Client: Proxy ist Server, für Server: Proxy ist Client



- Proxy arbeitet auf Anwendungsschicht
  - kann auf der Basis des Nachrichteninhalts filtern
  - z.B. HTTP-Anfragen nach bestimmten Seiten nur von speziellen Hosts akzeptieren

Grenzen von Firewalls

- Kein Schutz interner Benutzer untereinander
- Nur begrenzter Schutz gegen mobilen Code (z.B. Email Wurm)
- Schutz von Teilen eines Netzes schwierig
- Angreifer kann sich in privilegiertes Netz „einschleichen“
  - z.B. bei drahtlosen Netzen
- Filterung über Sender-IP-Adresse/Port ist unsicher

Vorteil von Firewalls

- Umsetzung einer Sicherheitsstrategie an zentraler Stelle

10.7 Zusammenfassung

- Sicherheitsanforderungen:
  - Vertraulichkeit, Integrität, Authentizität,Verbindlichkeit
  - Verfügbarkeit, Anonymität, ...
- IP, TCP, UDP erfüllen keine Sicherheitsanforderungen
  - Vertraulichkeit, Integrität, Authentizität
- Kryptographische Verfahren:
  - symmetrische und asymmetrische Chiffren
  - Kryptographische Hashes (Message Digest)
- Sicherheitsmechanismen
  - Authentifizierung (Kommunikationspartner, Nachrichten)
  - Integrität: Hashwerte mit Schlüssel, digitale Signatur
  - Verteilung öffentlicher Schlüssel: Zertifikate

10.7 Zusammenfassung ...

- Sichere Protokolle, z.B. PGP, TLS (HTTPS), IPsec
- Firewalls

Fortsetzung:

- Rechnernetze-Praktikum (WiSe)
  - Aufbau von Netzen, Routing und Switching
  - PO 2012: B.Sc., Vertiefungspraktikum, 5 LP
  - FPO 2021: B.Sc., Grundlagenpraktikum, 6 LP
- Rechnernetze II (SoSe)
  - weitere Netzwerktechnologien (Fast Ethernet, WLAN, ...)
  - Vertiefung (Routing, QoS, IPsec, ...)
  - PO 2012: B.Sc., Wahlmodul, 5 LP
  - FPO 2021: M.Sc., Kernmodul, 6 LP