Netzwerktechnologien
3 VO

Univ.-Prof. Dr. Helmut Hlavacs
helmut.hlavacs@univie.ac.at

Dr. Ivan Gojmerac
gojmerac@ftw.at

Bachelorstudium Medieninformatik
SS 2012

Kapitel 8 - Netzwerksicherheit

8.1 Sicherheitsanforderungen in Netzen

Vertraulichkeit: nur der Sender und der korrekte Adressat sollen den Inhalt der Nachricht lesen können

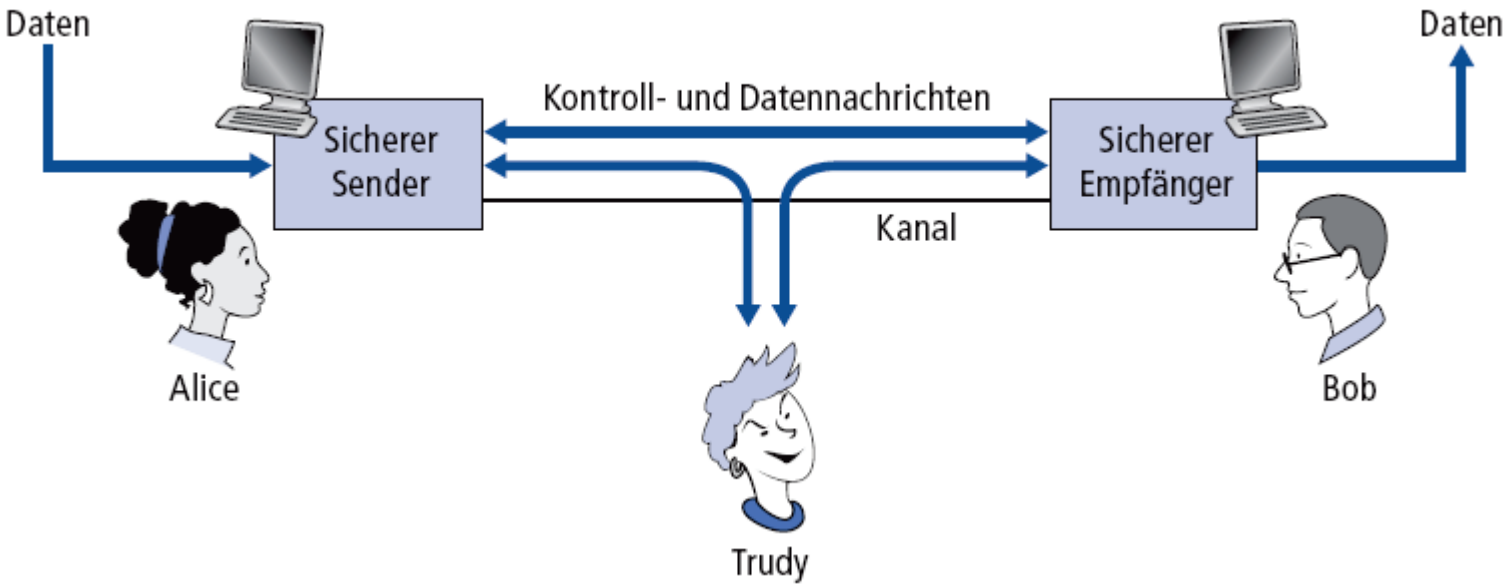
Authentifizierung: Sender und Empfänger wollen gegenseitig ihre Identität sicherstellen

Nachrichtenintegrität: Sender und Empfänger wollen sicherstellen, dass die Nachricht nicht unbemerkt verändert wurde (während der Übertragung oder danach)

Zugriff und Verfügbarkeit: Dienste müssen für Benutzer zugreifbar und verfügbar sein

8.1 Freunde und Feinde – Alice, Bob, Trudy

- Alice, Bob und Trudy sind „bekannte Gestalten“ in der Welt der Netzwerksicherheit
  - Alice und Bob möchten “sicher” kommunizieren
  - Trudy (ein Eindringling) kann Nachrichten abfangen, löschen, einfügen



8.1 Wer könnten Bob und Alice sonst noch sein?

- Webbrowser/-server für elektronische Transaktionen (z.B. Online-Einkäufe)
- Client und Server für Online-Banking
- DNS-Server
- Router, die Routingtabellen-Updates austauschen
- Weitere Beispiele?

8.1 Typen von Angriffen

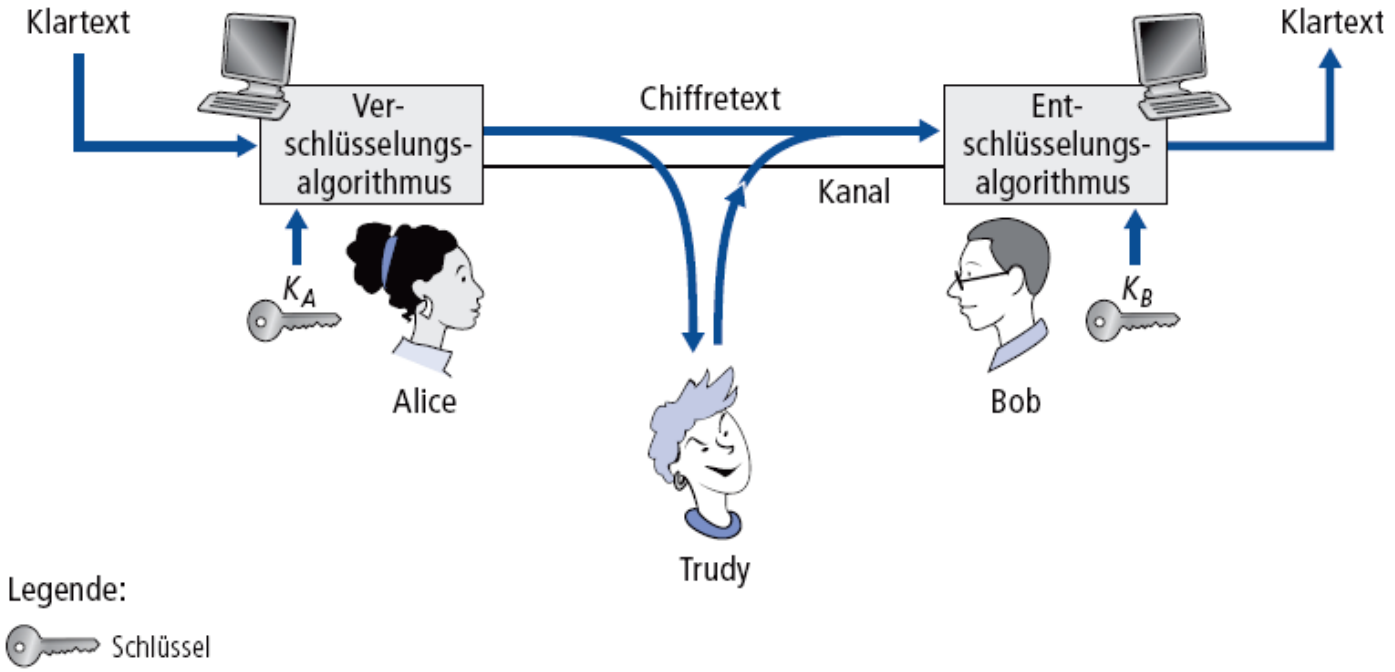
Q: Was können Angreifer tun?

A: Eine ganze Menge!

- **Lauschen:** Nachrichten mitlesen
- Aktiv Nachrichten in die Verbindung **einspeisen**
- **Fremde Identitäten annehmen und Quelladressen** (oder andere Felder im Paket) **fälschen**
- **Denial of Service:** verhindern, dass andere einen Dienst nutzen können (z.B. durch Überlasten von Ressourcen)
- Usw.

Kapitel 8 - Netzwerksicherheit

8.2 Terminologie der Kryptographie



Symmetrische Kryptographie: Sender- und Empfängerschlüssel sind *identisch*

Public-Key-Kryptographie: Schlüssel zur Verschlüsselung ist *öffentlich bekannt*, zur Entschlüsselung *geheim*

8.2 Kryptographie mit symmetrischen Schlüsseln

Ersetzungschiffre: eine Sache durch eine andere ersetzen

– Monoalphabetische Chiffre: einen Buchstaben durch einen anderen ersetzen

```
Klartext:   abcdefghijklmnopqrstuvwxyz
```

```
Chiffretext:  mnbvcxzasdfghjklpoiuytrewq
```

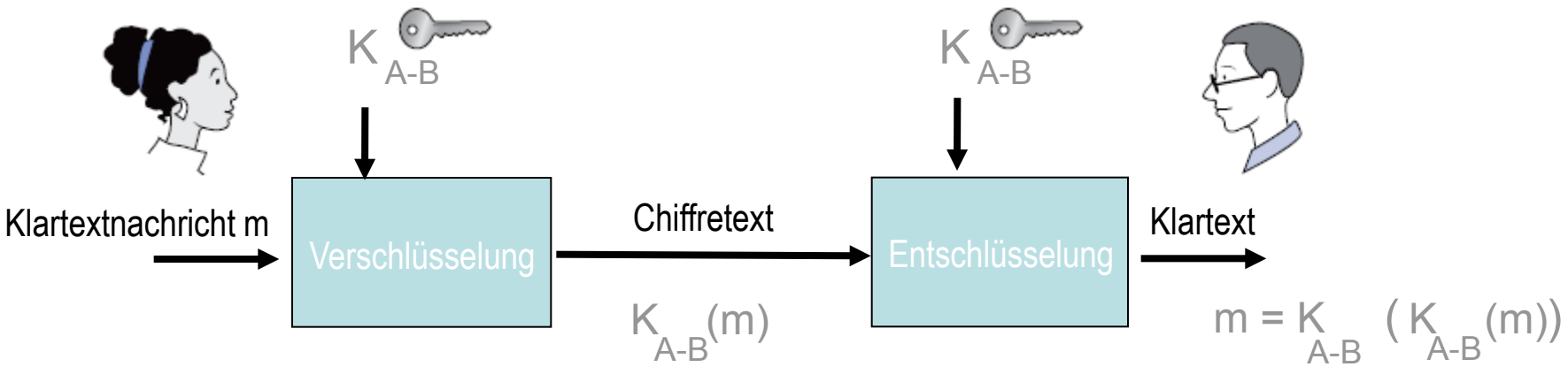
z.B.:

```
Klartext: bob, ich liebe dich. alice
Chiffretext: nkn, sba pscbc vsba. mgsbc
```

8.2 Kryptographie mit symmetrischen Schlüsseln

Kryptographie mit symmetrischen Schlüsseln: Bob and Alice kennen denselben (symmetrischen) Schlüssel **K**

- Der Schlüssel könnte zum Beispiel das Ersetzungsmuster der monoalphabetischen Chiffre sein



8.2 Symmetrische Kryptographie: DES Algorithmus

DES: Data Encryption Standard

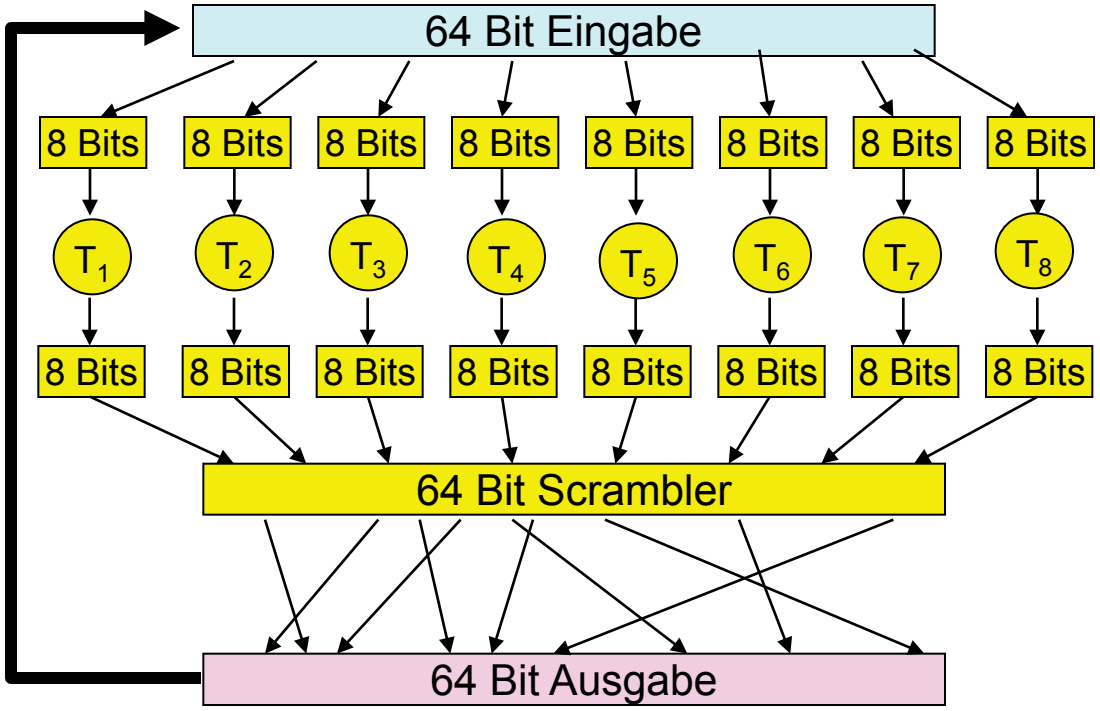
- US-Verschlüsselungsstandard [NIST 1993]
- Symmetrische 56-Bit-Schlüssel, 64 Bit lange Klartext-Eingaben
- Problem: DES ist nicht ausreichend sicher → [RFC 4772]

8.2 AES: Advanced Encryption Standard

- Neuer symmetrischer NIST-Standard (Nov. 2001), der DES ersetzen soll.
- Verarbeitet Daten in 128-Bit-Blöcken
- 128, 192, oder 256 Bit lange Schlüssel
- Wenn Brute-Force-Entschlüsselung (alle Schlüssel ausprobieren) für DES eine Sekunde dauert, braucht sie für AES-128 149 Billionen Jahre

8.2 Blockchiffre

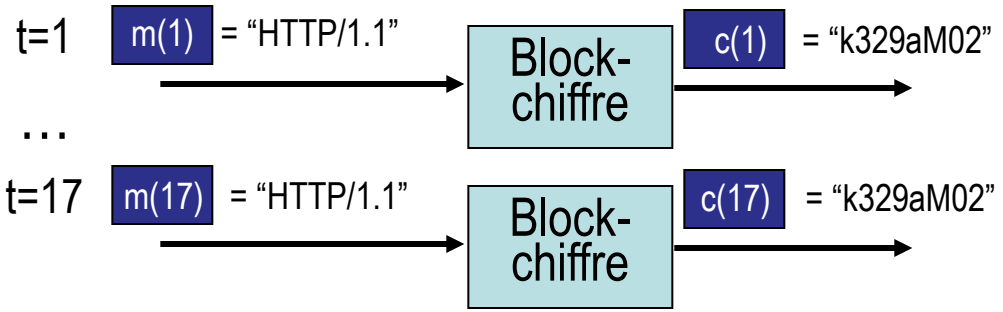
Schleife über n Runden



- Ein Durchlauf: ein Eingabebit beeinflusst acht Ausgabebits
- Mehrere Durchläufe: jedes Eingabebit hat Auswirkungen auf alle Ausgabebits
- Blockchiffren: DES, 3DES, AES

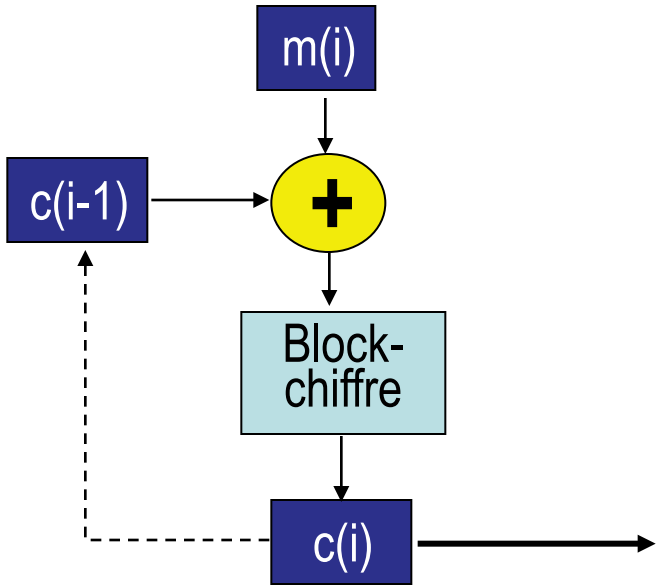
Cipher Block Chaining

Wenn ein Eingabeblock sich wiederholt, wird dieselbe Chiffre eine identische Ausgabe erzeugen.



Abhilfe:

- **Cipher Block Chaining:** XOR des i-ten Eingabeblocks m(i) mit dem vorangegangenen verschlüsselten Block c(i-1)
  - **Initialisierungsvektor** c(0) wird im Klartext an den Empfänger übertragen



8.2 Public Key Kryptographie

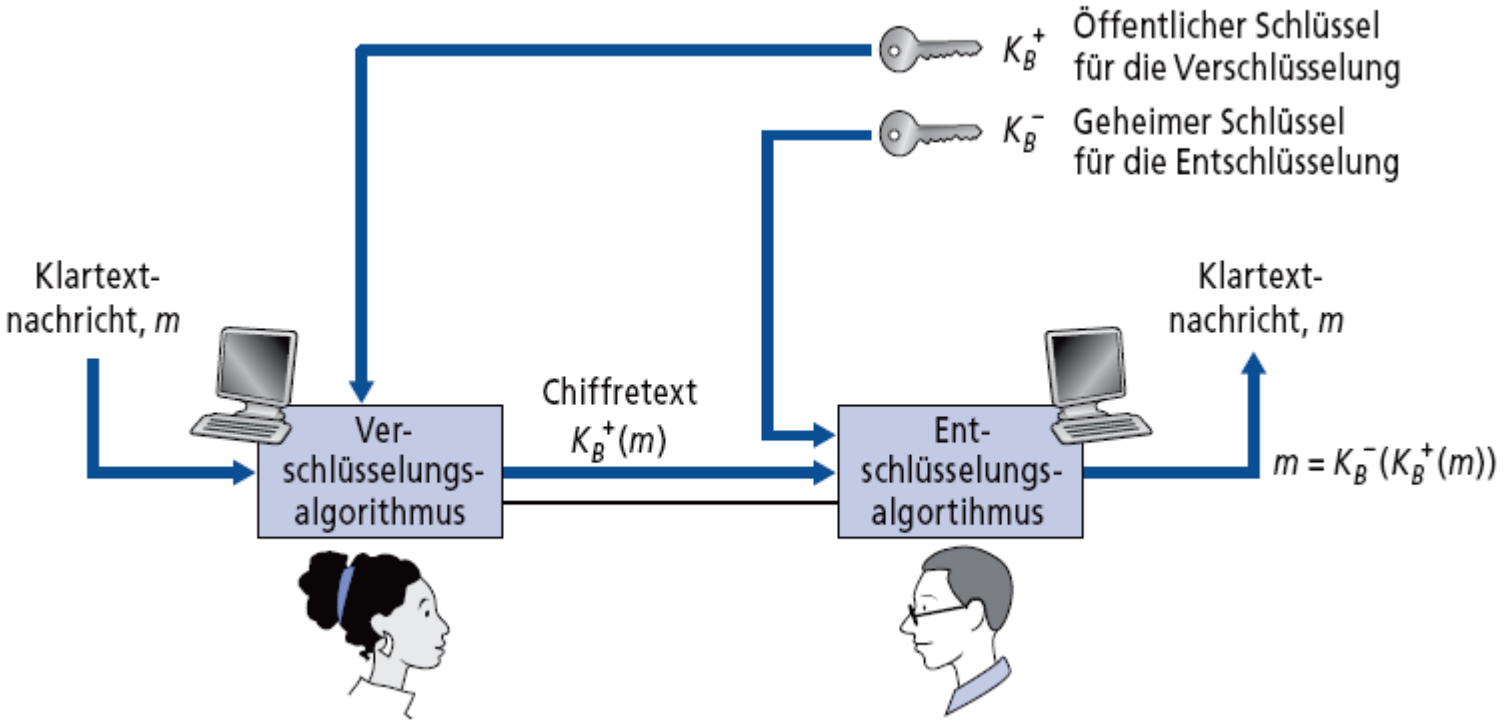
Symmetrische Kryptographie:

- erfordert, dass Sender und Empfänger über einen gemeinsamen Schlüssel verfügen
- Problem der symmetrischen Kryptographie: Wie kann man sich überhaupt auf einen Schlüssel einigen (vor allem dann, wenn man sich noch nie “getroffen” hat)?

Public-Key-Kryptographie

- radikal anderer Ansatz [RSA78]
- Sender, Empfänger kennen **keinen gemeinsamen** geheimen Schlüssel
- **öffentlicher** Verschlüsselungsschlüssel, den **alle** kennen
- **geheimen** Entschlüsselungsschlüssel kennt nur der Empfänger

8.2 Public Key Kryptographie



8.2 Public Key Algorithmen

Anforderungen:

1. benötigt $K_B^+( )$ und $K_B^-( )$, für die

$$K_B^-(K_B^+(m)) = m$$

2. gegeben den öffentl. Schlüssel K_B^+, soll es nicht möglich sein, den privaten Schlüssel K^B zu errechnen

RSA: Algorithmus von Rivest, Shamir, Adleman

8.2 RSA - Schlüsselgenerierung

1. Wähle zwei große Primzahlen *p, q*.
   (Wobei das Produkt von *p* und *q* z.B. 1024 Bit lang ist.)
2. Berechne n = *pq*, *z* = *(p-1)(q-1)*.

3. Wähle ein e *(*mit *e<n),* das keine Primfaktoren mit z gemeinsam hat. (*e, z* sind "relative Primzahlen").
4. Wähle d, so dass *ed-1* durch z ohne Rest teilbar ist
   (in anderen Worten: *ed* mod *z* = *1*).
5. **Öffentlicher** Schlüssel: Zahlenpaar *(*n, e*)*. **Privater** Schlüssel: Zahlenpaar *(*n, d*)*.

K_B^+ K_B^-

8.2 RSA – Ver- und Entschlüsselung

0. Gegeben (n,e) und (n,d), berechnet wie oben

1. Um ein Bitmuster m zu verschlüsseln, berechne

   $c = m^e \bmod n$ (Rest beim Teilen von m^e durch n)

2. Zum Entschlüsseln des empfangenen Wetes c berechne

   $m = c^d \bmod n$ (Rest beim Teilen von c^d durch n)

$$m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$$

8.2 Beispiel zu RSA

Bob wählt *p=5, q=7*. Dann gilt: *n=35, z=(p-1)(q-1)=24*.
e=5 (*e* und *z* sind relative Primzahlen)
d=29 (*ed-1* ist ohne Rest teilbar durch z)

verschlüsseln:	Zeichen	m	m^e	$c = m^e \bmod n$
	I	12	1524832	17

entschlüsseln:	c	c^d	$m = c^d \bmod n$	Zeichen
	17	481968572106750915091411825223071697	12	I

8.2 RSA

Warum ist $m = (m^e \bmod n)^d \bmod n$

Resultat aus der Zahlentheorie: Wenn p,q Primzahlen sind und $n = pq$, dann gilt:

$$x^y \bmod n = x^{y \bmod (p-1)(q-1)} \bmod n$$

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$

$$= m^{ed \bmod (p-1)(q-1)} \bmod n$$

$$= m^1 \bmod n$$

(weil wir ed so **gewählt** haben, dass es durch $(p-1)(q-1)$ mit Rest 1 teilbar ist)

$$= m$$

8.2 RSA

Eine wichtige Eigenschaft von RSA:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{Erst öffentlicher Schlüssel angewendet, dann privater Schlüssel}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{Erst privater Schlüssel angewendet, dann öffentlicher Schlüssel}}$$

→ Identische Ergebnisse!

Kapitel 8 - Netzwerksicherheit

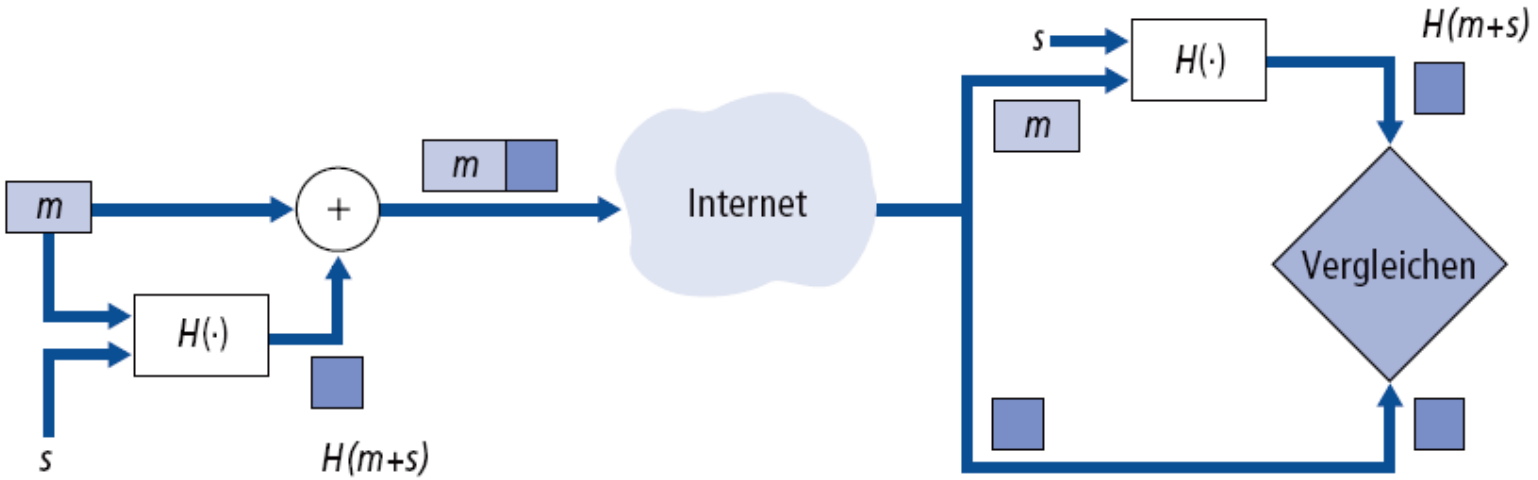
8.3 Nachrichtenintegrität

Bob empfängt eine Nachricht von Alice und möchte sicherstellen, dass…
…die Nachricht tatsächlich von Alice stammt.
…die Nachricht seit dem Versand durch Alice nicht verändert wurde.

Kryptographische Hashfunktion:

- Nimmt Eingabe m, erstellt Hash H(m) fester Länge
  - Wie z.B. die Internet-Prüfsumme
- Rechnerisch nicht möglich, zwei unterschiedliche Nachrichten x, y zu finden, für die H(x) = H(y)

8.3 Message Authentication Code



Legende:

m = Nachricht

s = gemeinsames Geheimnis

8.3 MACs in der Praxis

Für MACs wird oft die Hashfunktion MD5 verwendet (RFC 1321)

- Berechnet einen 128-Bit-MAC in einem 4-stufigen Prozess
- Gegeben einen 128-Bit-String x, erscheint es schwierig, eine Nachricht m zu konstruieren, deren MD5-Hash x ist
  - In der jüngeren Vergangenheit (2005) gab es Ansätze für Angriffe auf MD5

Die Hashfunktion SHA-1 wird auch oft für MACs verwendet

- US-Standard [NIST, FIPS PUB 180-1]
- 160-Bit-MAC
- Seit 2005 sind auch Angriffe auf SHA-1 bekannt

8.3 Digitale Unterschriften

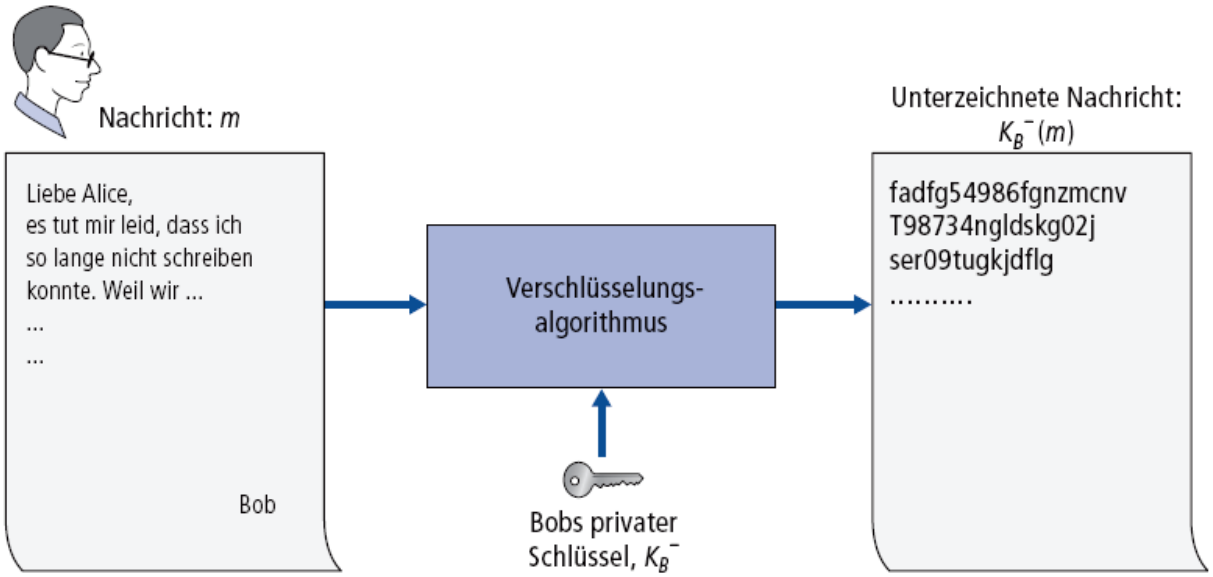
Kryptographische Technik, die der eigenhändigen “analogen” Unterschrift entspricht

- Sender (Bob) unterschreibt ein Dokument digital und hält dadurch fest, dass er der Urheber/Besitzer ist
- **Überprüfbar, nicht fälschbar:** Empfänger (Alice) kann beweisen, dass Bob (und niemand sonst, einschließlich ihr selbst) das Dokument unterschrieben hat

8.3 Digitale Unterschriften

Einfache digitale Unterschrift für Nachricht m:

- Bob "unterschreibt" m durch Verschlüsseln mit seinem geheimen Schlüssel K_B^-, wodurch die "signierte" Nachricht $K_B^-(m)$ entsteht



8.3 Digitale Unterschriften

- Angenommen Alice empfängt m und die digitale Signatur $K_B^-(m)$.
- Alice überprüft Bobs Unterschrift unter m, indem sie mittels Bobs öffentlichem Schlüssel überprüft, ob $K_B^+(K_B^-(m)) = m$.
- Wenn $K_B^+(K_B^-(m)) = m$, dann muss der Unterzeichner (wer auch immer es ist) Bobs geheimen Schlüssel besitzen.

Alice stellt damit sicher:

- ✓ Bob hat m unterschrieben.
- ✓ Niemand sonst kann die Unterschrift erzeugt haben.
- ✓ Bob hat m und nicht eine andere Nachricht m' unterschrieben.

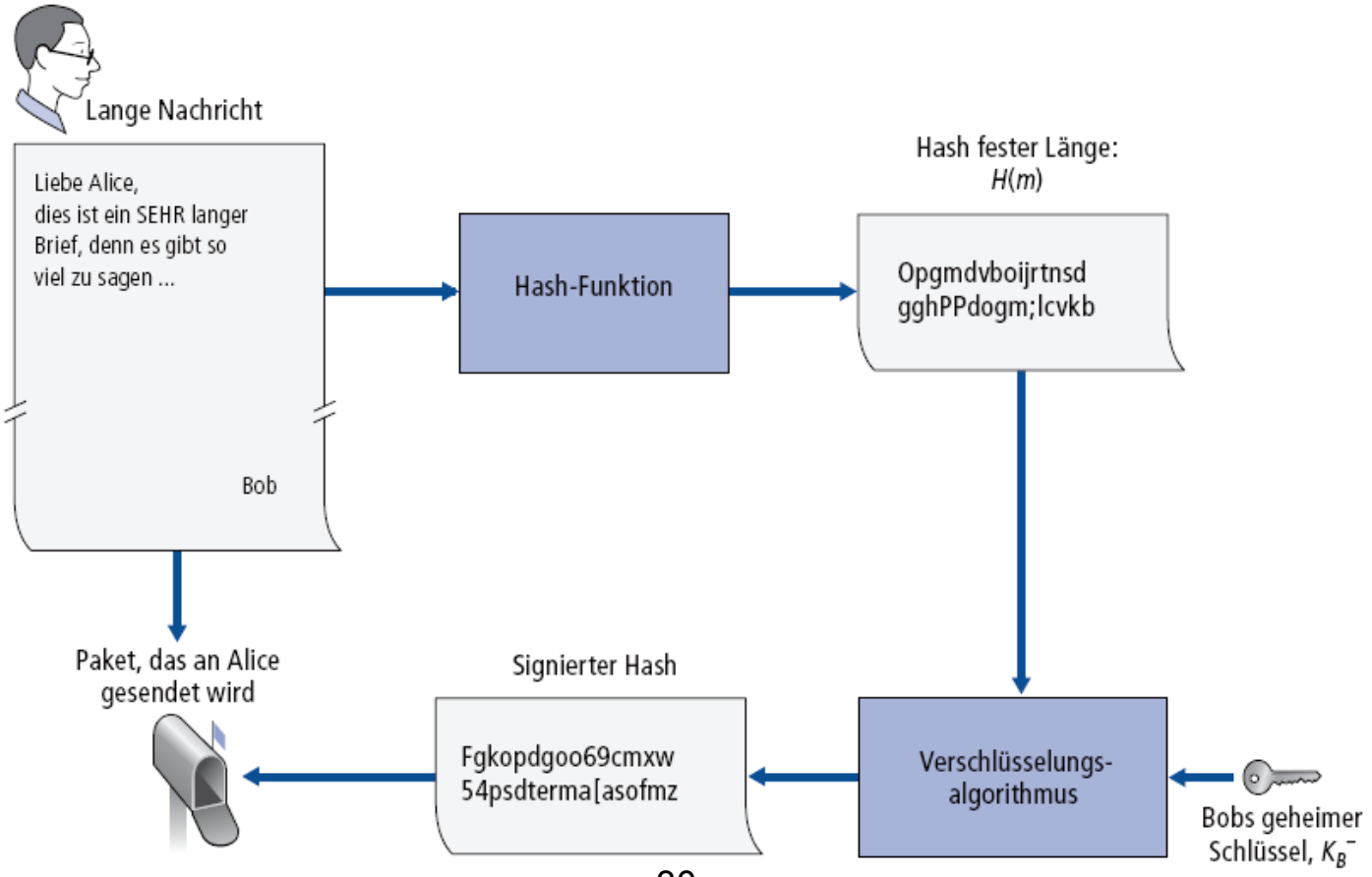
Nicht-Abstreitbarkeit:

- ✓ Alice kann mit m und der Signatur $K_B^-(m)$ vor Gericht ziehen und beweisen, dass Bob m unterschrieben hat.

8.3 Digitale Unterschrift: signierter Hash

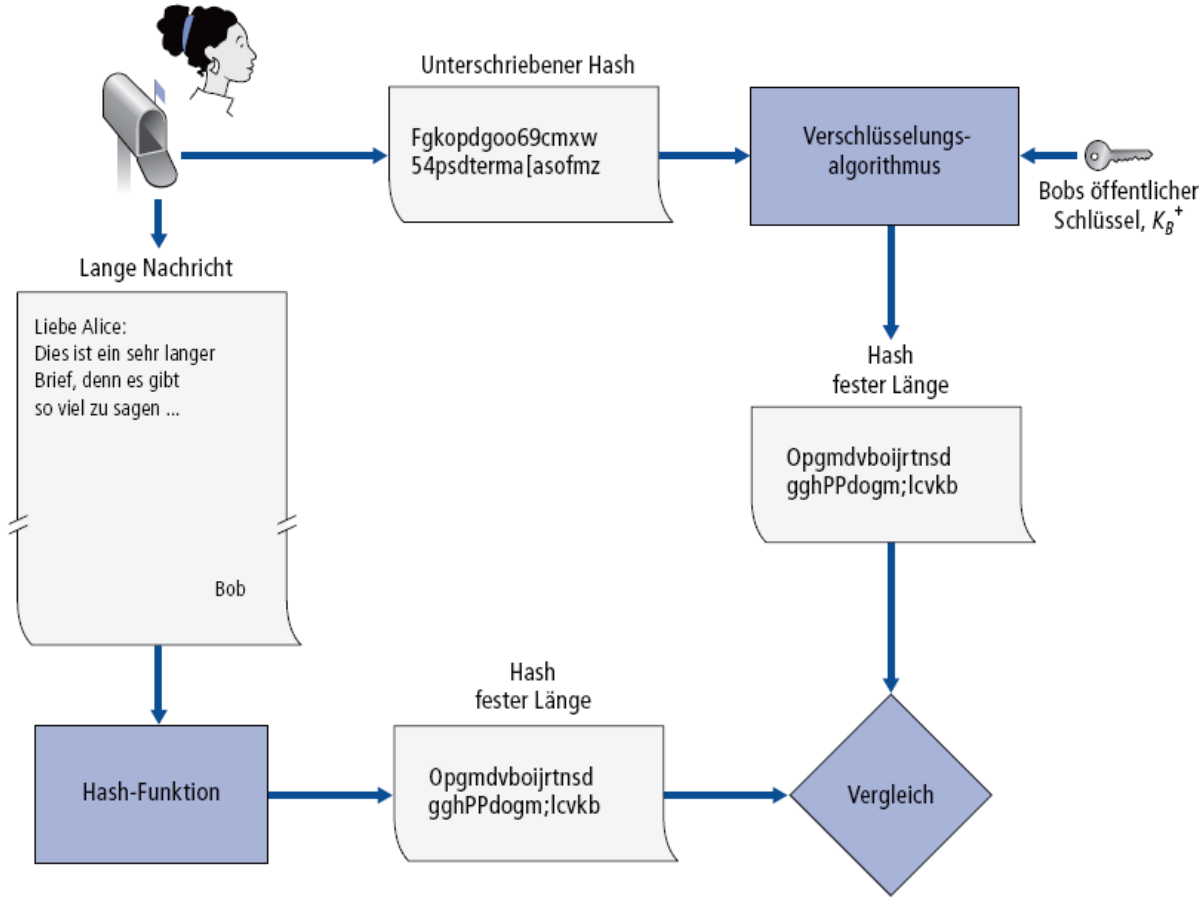
Das Signieren eines Hash-Wertes ist wesentlich weniger rechenaufwendig als das Signieren einer möglicherweise sehr großen Originalnachricht.

Bob verschickt eine digital signierte Nachricht:



8.3 Digitale Unterschrift: signierter Hash

Alice überprüft die Signatur und die Integrität der digital signierten Nachricht von Bob:



8.3 Zertifizierung öffentlicher Schlüssel

Problem bei öffentlichen Schlüsseln:

- Wenn Alice den öffentlichen Schlüssel von Bob erhält (von einer Webseite, per E-Mail, usw.), wie kann sie sicherstellen, dass es wirklich Bobs Schlüssel ist, und nicht ein von Trudy erzeugter Schlüssel?

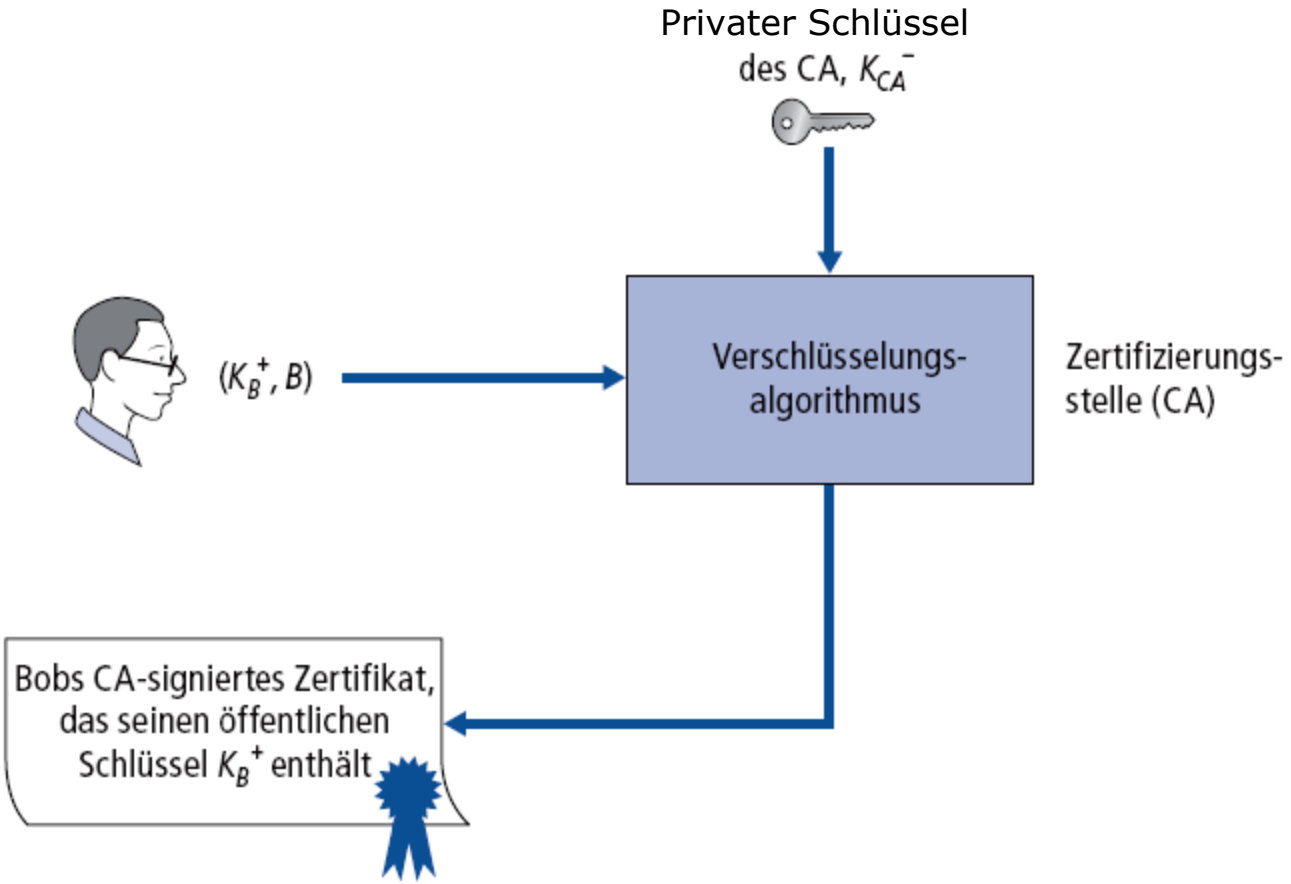
Lösung:

- Vertrauenswürdige Zertifizierungsstelle (Certification Authority, CA)

8.3 Zertifizierungsstelle

- Zertifizierungsstelle / Certification Authority (CA): verknüpft einen öffentlichen Schlüssel mit einer bestimmten *Entität E*.
- E registriert den öffentlichen Schlüssel bei der CA:
  - E “beweist” die eigene Identität gegenüber der CA.
  - CA erstellt ein Zertifikat, das E mit dem öffentlichen Schlüssel von E verknüpft.
  - Das Zertifikat wird von der CA digital unterschrieben und besagt: “Das ist der öffentliche Schlüssel von E.”

8.3 Zertifizierungsstelle



Die Version dieses Bildes im Buch beinhaltet Fehler!

8.3 Zertifizierungsstelle

Wenn Alice Bobs öffentlichen Schlüssel benötigt:

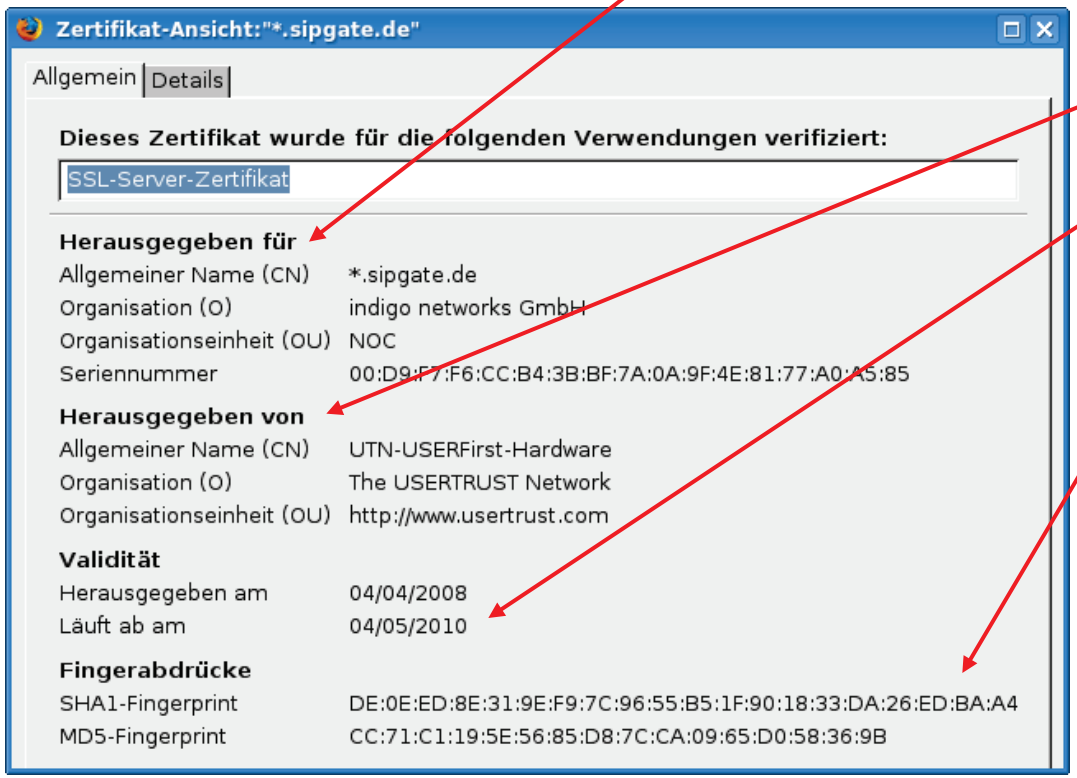
- Bobs Zertifikat besorgen (von Bob oder von irgendwo sonst).
- Öffentlichen Schlüssel der CA anwenden, um die Verbindung zwischen Bobs öffentlichem Schlüssel und seiner Identität zu überprüfen
- Verwenden des so überprüften öffentlichen Schlüssels von Bob

Frage: Wie kommt Alice zum öffentlichen Schlüssel der CA?

- Der öffentliche Schlüssel der CA muss entweder in der Anwendung (z.B. Webbrowser) oder im Betriebssystem schon vorinstalliert sein

8.3 Zertifikat

- Seriennummer (eindeutig pro Aussteller)
- Information über den Zertifikatsinhaber, einschließlich des Algorithmus und des Schlüssels selbst (hier nicht gezeigt)



- Info über Aussteller
- Gültigkeitszeitraum
- Digitale Unterschrift des Ausstellers

Kapitel 8 - Netzwerksicherheit

8.4 Authentifizierung

WICHTIGE ANMERKUNG: Die Folien 38-50 enthalten zu didaktischen Zwecken erfundene (d.h. fiktive) Authentifizierungsprotokolle **ap1.0** bis **ap5.0**.

Ziel: Bob möchte, dass Alice ihm ihre Identität "beweist"

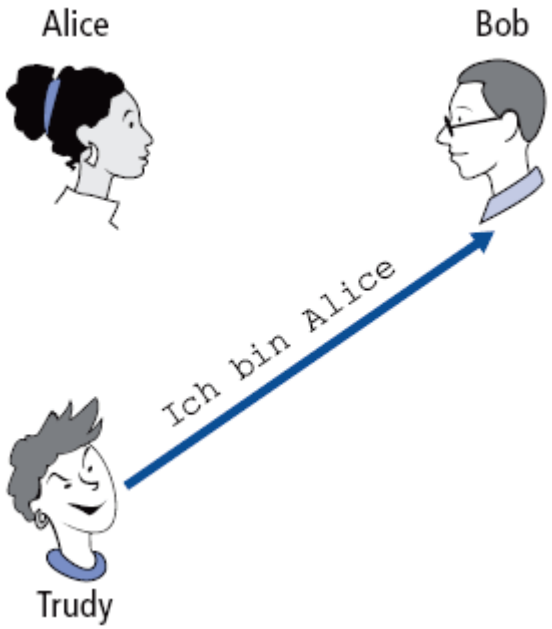
Protokoll ap1.0: Alice sagt "Ich bin Alice"

↯ Angriffsszenario?

8.4 Authentifizierung

Ziel: Bob möchte, dass Alice ihm ihre Identität “beweist”

Protokoll ap1.0: Alice sagt “Ich bin Alice”



In einem Netzwerk kann Bob Alice nicht “sehen”, also kann Trudy einfach behaupten, Alice zu sein.

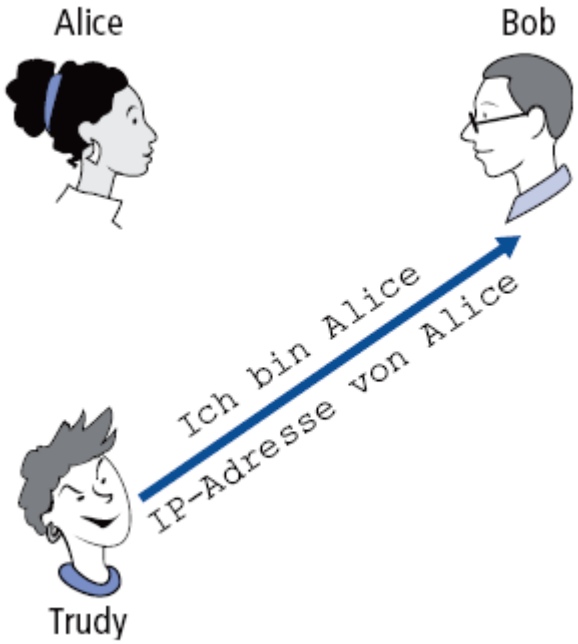
8.4 Authentifizierung

Protokoll ap2.0: Alice sagt “Ich bin Alice” in einem IP-Paket, das ihre Quell-IP enthält

↯ Angriffsszenario?

8.4 Authentifizierung

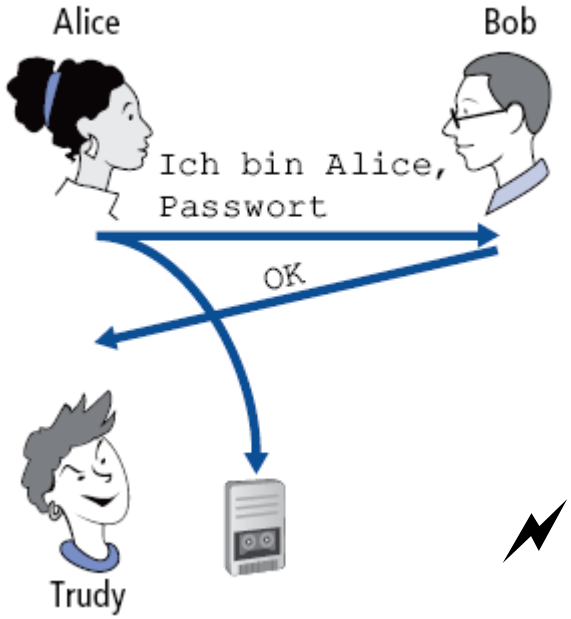
Protokoll ap2.0: Alice sagt “Ich bin Alice” in einem IP-Paket, das ihre Quell-IP enthält



Trudy kann ein Paket mit gefälschter Absenderadresse erzeugen

8.4 Authentifizierung

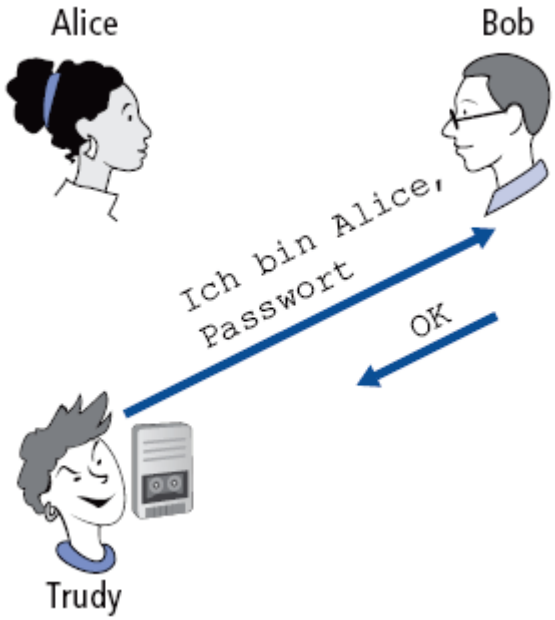
Protokoll ap3.0: Alice sagt “Ich bin Alice” und schickt ihr geheimes Passwort als “Beweis” mit.



↯ Angriffsszenario?

8.4 Authentifizierung

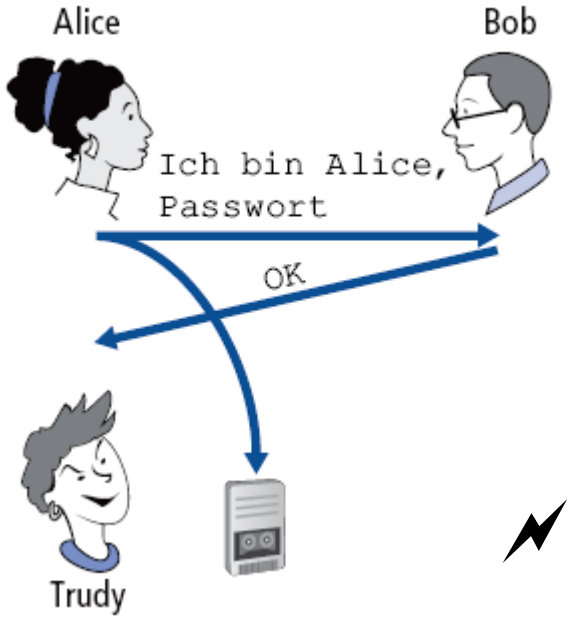
Protokoll ap3.0: Alice sagt “Ich bin Alice” und schickt ihr geheimes Passwort als “Beweis” mit.



Playback-Angriff:
Trudy zeichnet Alices Paket auf und wiederholt es später in ihrer Anfrage an Bob.

8.4 Authentifizierung

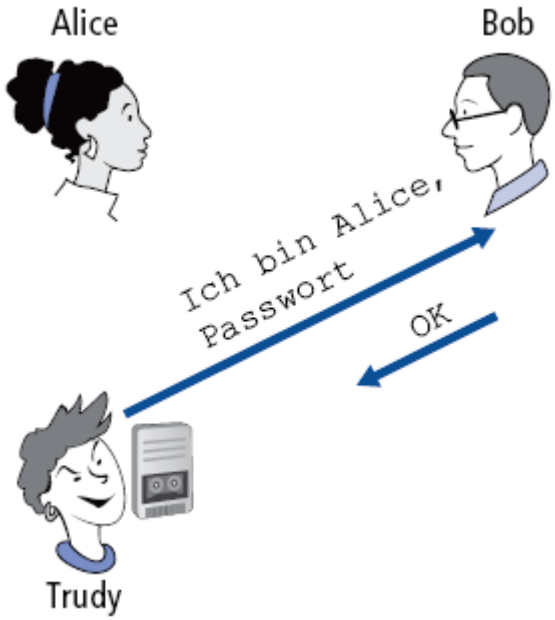
Protokoll ap3.1: Alice sagt “Ich bin Alice” und schickt ihr verschlüsseltes geheimes Passwort als “Beweis” mit.



↯ Angriffsszenario?

8.4 Authentifizierung

Protokoll ap3.1: Alice sagt “Ich bin Alice” und schickt ihr verschlüsseltes geheimes Passwort als “Beweis” mit.



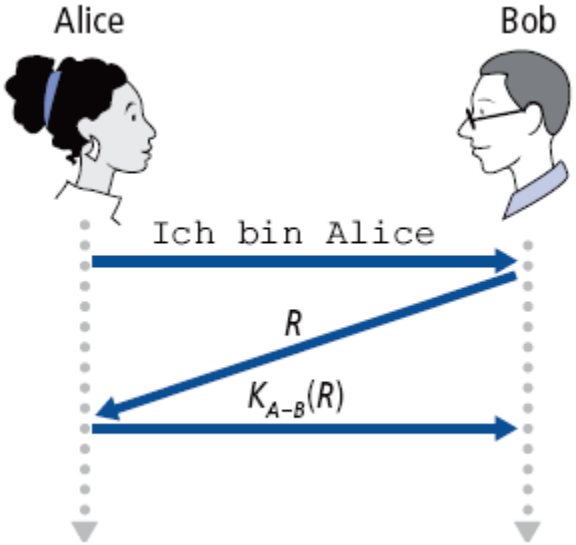
Aufzeichnen und wiederholen funktioniert immer noch!

8.4 Authentifizierung

Ziel: Playback-Angriff verhindern

Nonce: Zahl (R), die genau einmal verwendet wird (“number used once”)

Protokoll ap4.0: Um zu beweisen, dass Alice “live” an der Kommunikation teilnimmt, schickt Bob eine Nonce R, die Alice symmetrisch verschlüsseln und zurückschicken muss.



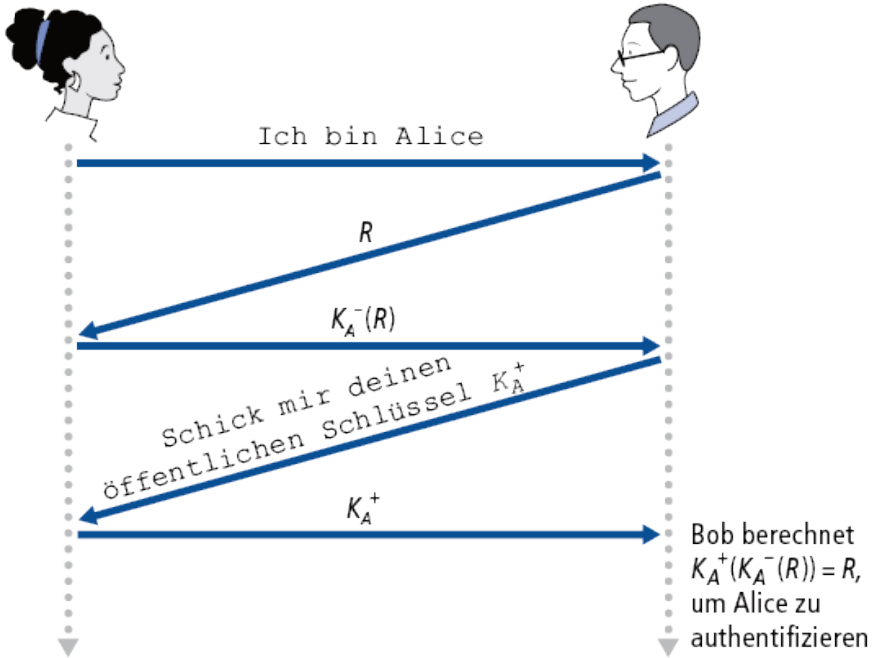
Nur Alice kennt den richtigen Schlüssel, also muss das Alice sein!

→ Fehler, Nachteile?

8.4 Authentifizierung

Protokoll ap4.0 setzt einen symmetrischen Schlüssel voraus.
→ Können wir Public-Key-Kryptographie verwenden?

Protokoll ap5.0: Verwendet eine Nonce und Public-Key-Kryptographie.



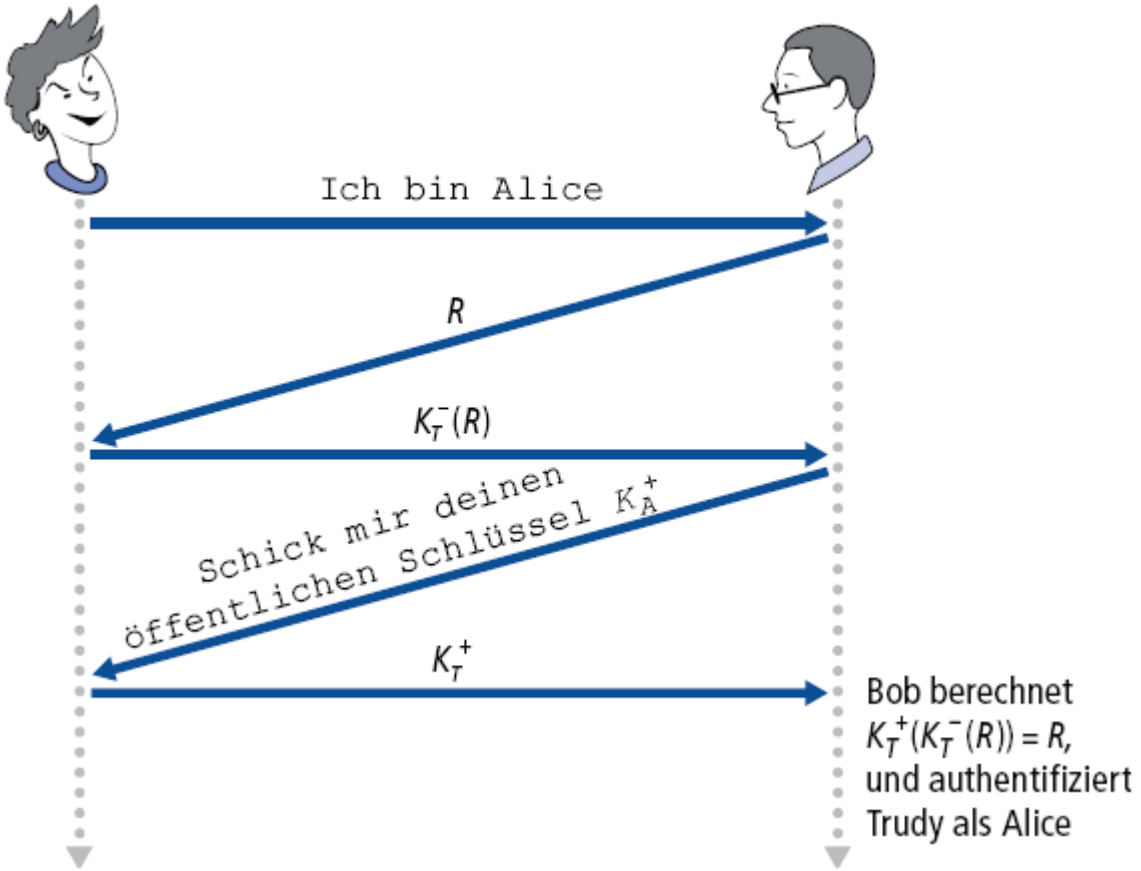
Bob berechnet

$$K_A^+ (K_A^-(R)) = R$$

und weiß, dass nur Alice den privaten Schlüssel hat, mit dem R so verschlüsselt werden kann, dass

$$K_A^+ (K_A^-(R)) = R$$

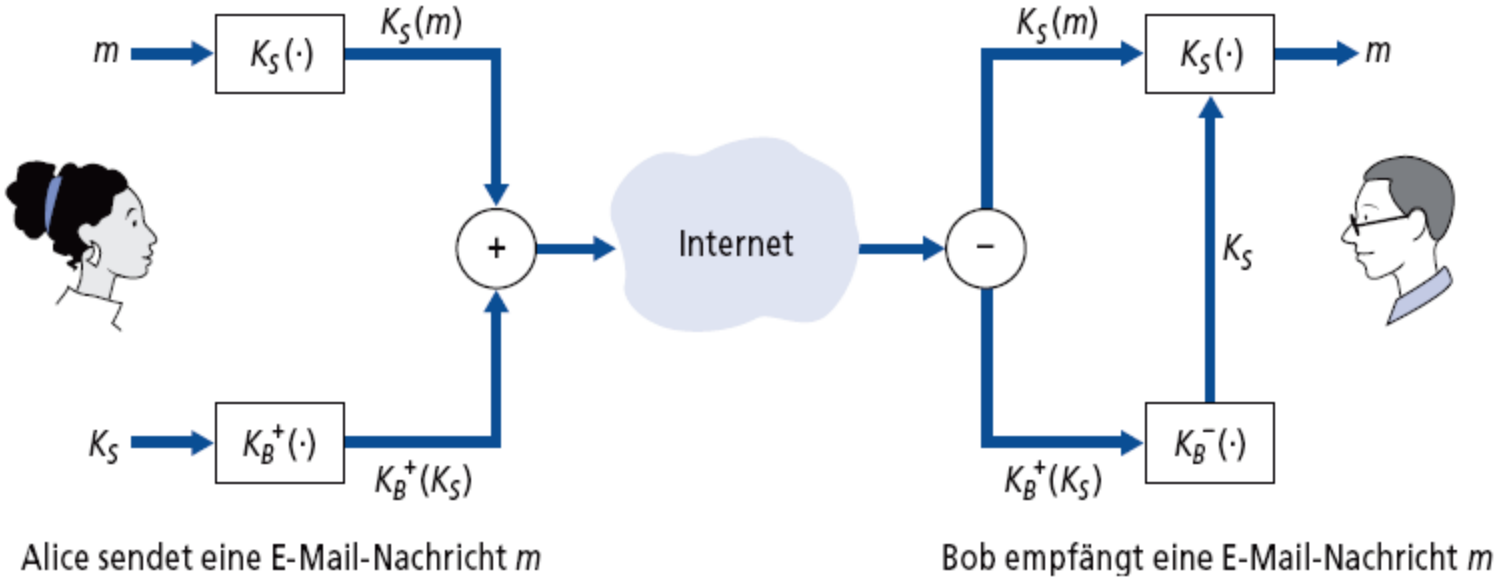
8.4 Ap5.0 benötigt Zertifikate



Kapitel 8 - Netzwerksicherheit

8.5 Sichere E-Mail

Alice möchte eine vertrauliche E-Mail m an Bob schicken:



Alice sendet eine E-Mail-Nachricht *m*

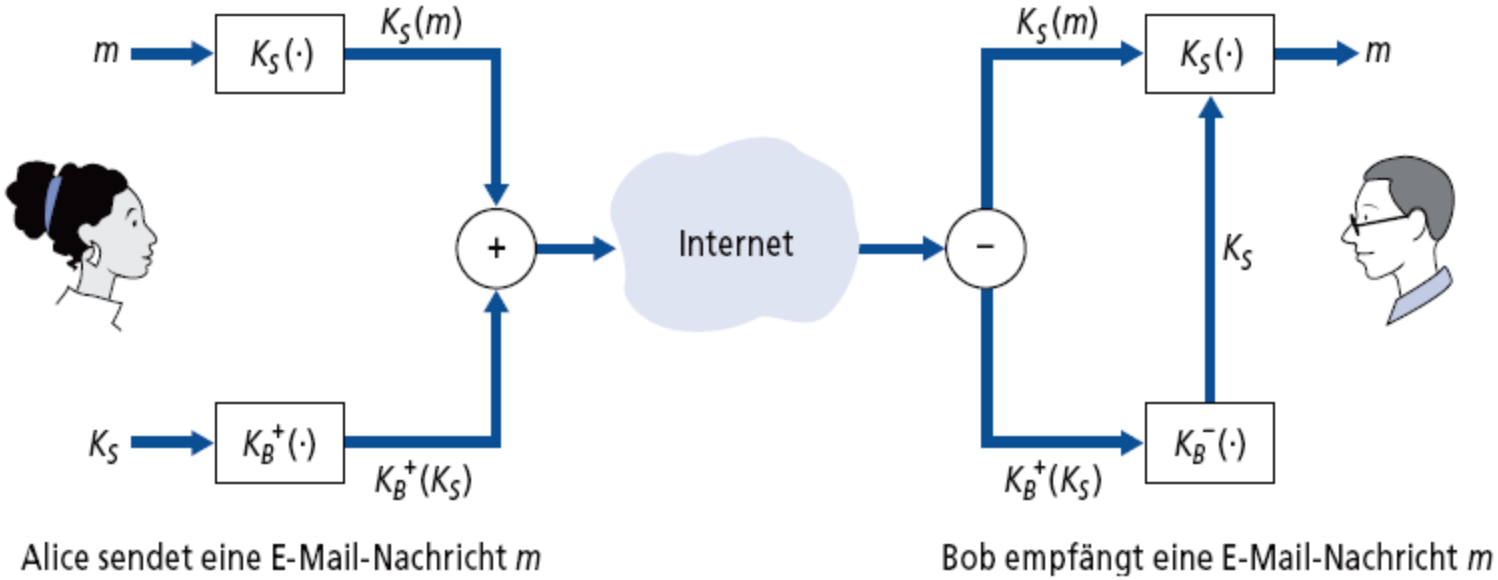
Bob empfängt eine E-Mail-Nachricht *m*

Alice:

- Erzeugt einen zufälligen *symmetrischen* Schlüssel K_S
- Verschlüsselt die Nachricht mit K_S (aus Effizienzgründen)
- Verschlüsselt K_S mit Bobs öffentlichem Schlüssel K_B^+
- Sendet sowohl $K_S(m)$ als auch $K_B^+(K_S)$ an Bob

8.5 Sichere E-Mail

Alice möchte eine vertrauliche E-Mail m an Bob schicken.



Alice sendet eine E-Mail-Nachricht *m*

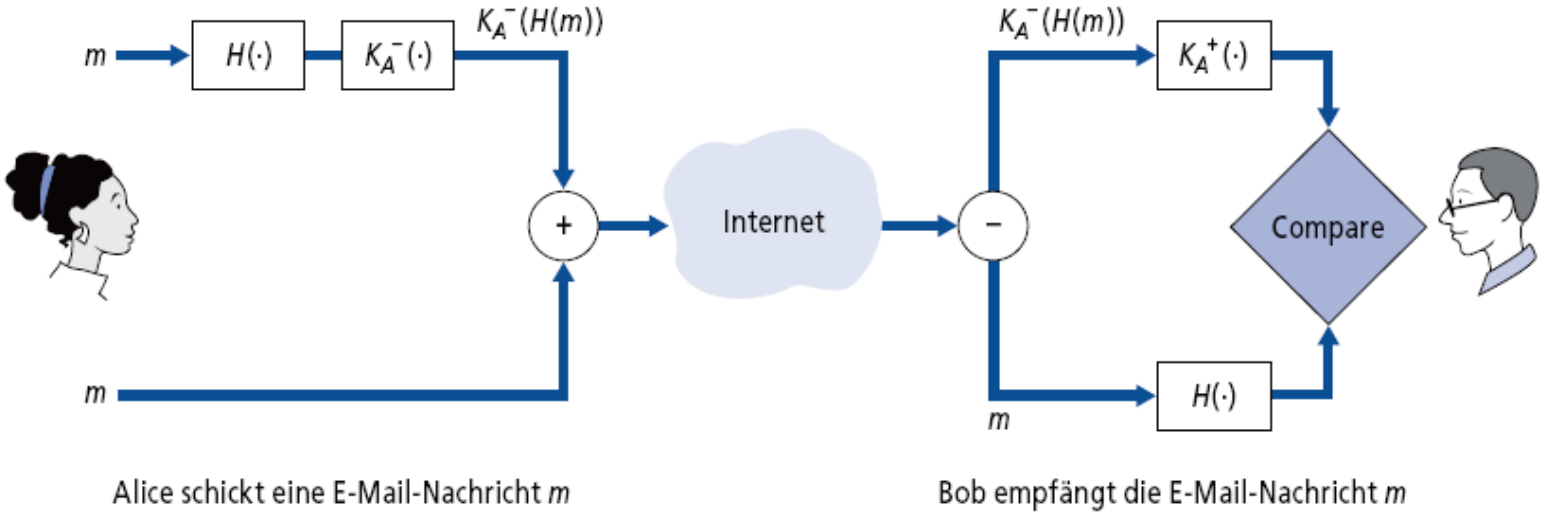
Bob empfängt eine E-Mail-Nachricht *m*

Bob:

- Verwendet seinen privaten Schlüssel K_B^-, um K_S zu erhalten
- Verwendet K_S, um $K_S(m)$ zu entschlüsseln und m zu lesen

8.5 Sichere E-Mail

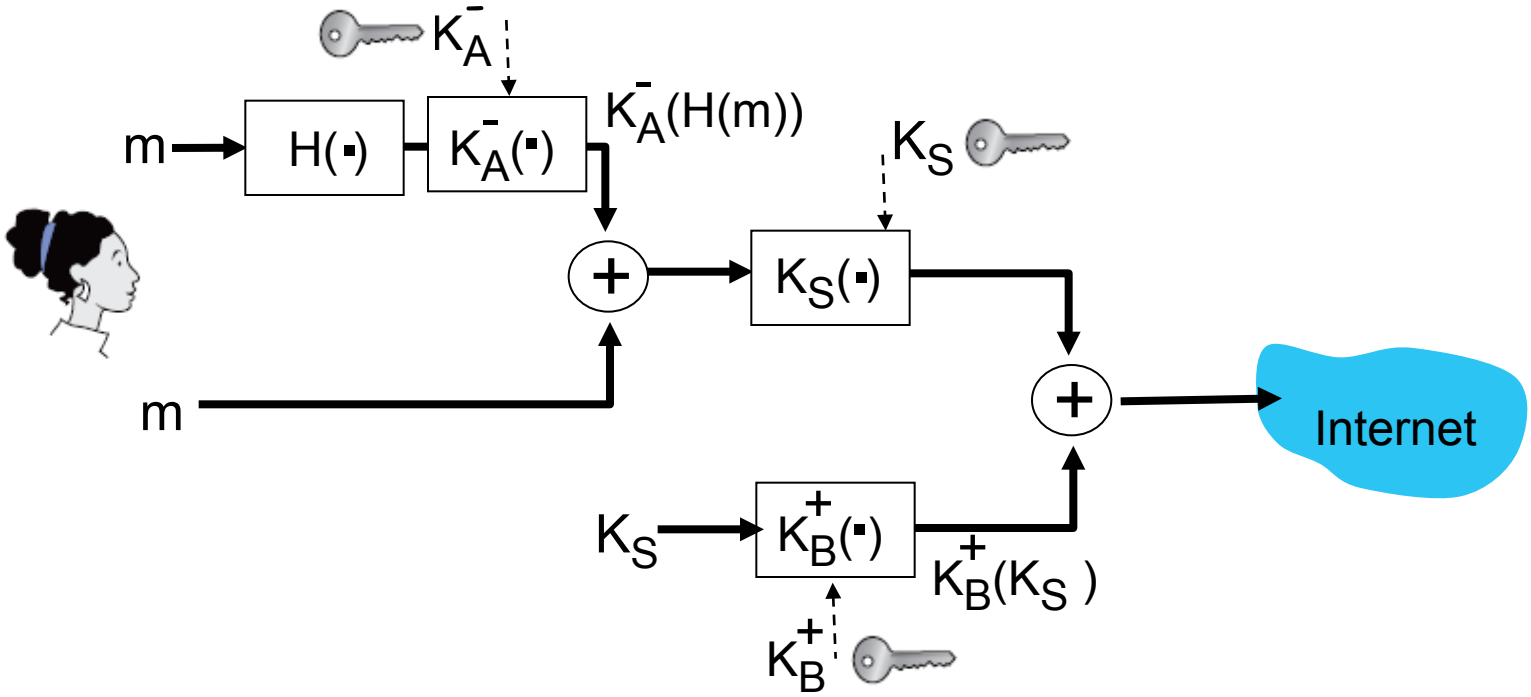
Alice möchte für ihre Nachricht an Bob Absenderauthentifizierung und Nachrichtenintegrität sicherstellen (nicht aber Vertraulichkeit).



- Alice unterschreibt die Nachricht digital
- Sie schickt sowohl die Nachricht als auch die Signatur

8.5 Sichere E-Mail

Alice möchte für ihre Nachricht an Bob Vertraulichkeit, Absenderauthentifizierung und Nachrichtenintegrität sicherstellen.



Alice verwendet drei Schlüssel: ihren privaten Schlüssel K_A^-, Bobs öffentlichen Schlüssel K_B^+ und einen neu erstellten symmetrischen Schlüssel K_S.

8.5 Pretty Good Privacy (PGP)

- Verfahren für E-Mail-Verschlüsselung im Internet, De-facto-Standard
- Verwendet symmetrische Kryptographie, Public-Key-Kryptographie, Hashfunktionen und digitale Unterschriften wie beschrieben
- Bietet Vertraulichkeit, Absenderauthentifizierung, Integrität
- Entwickler: Phil Zimmerman

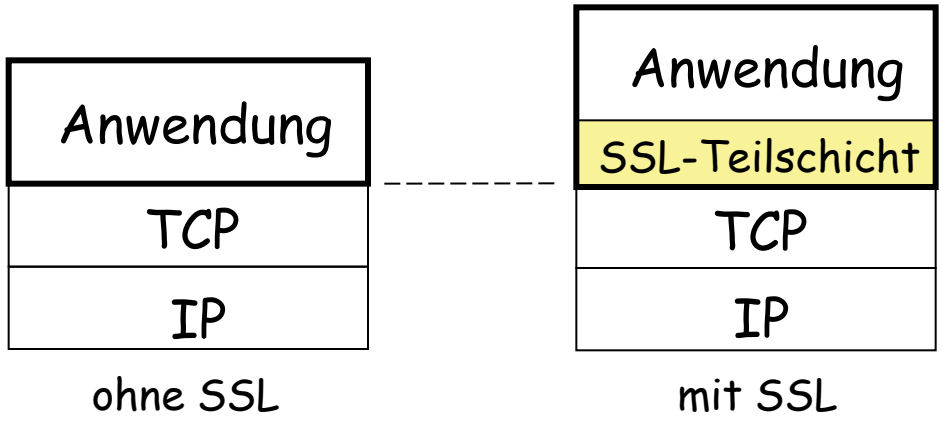
Eine PGP-signierte Nachricht:

```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

   Bob, My husband is out of
   town tonight. Passionately
   yours, Alice

---BEGIN PGP SIGNATURE---
Version: PGP 5.0
Charset: noconv
yhHJRHhGJGhgg/12EpJ+lo8gE4vB3m
   qJhFEvZP9t6n7G6m5Gw2
---END PGP SIGNATURE---
```

Kapitel 8 - Netzwerksicherheit

8.6 Secure Sockets Layer (SSL)

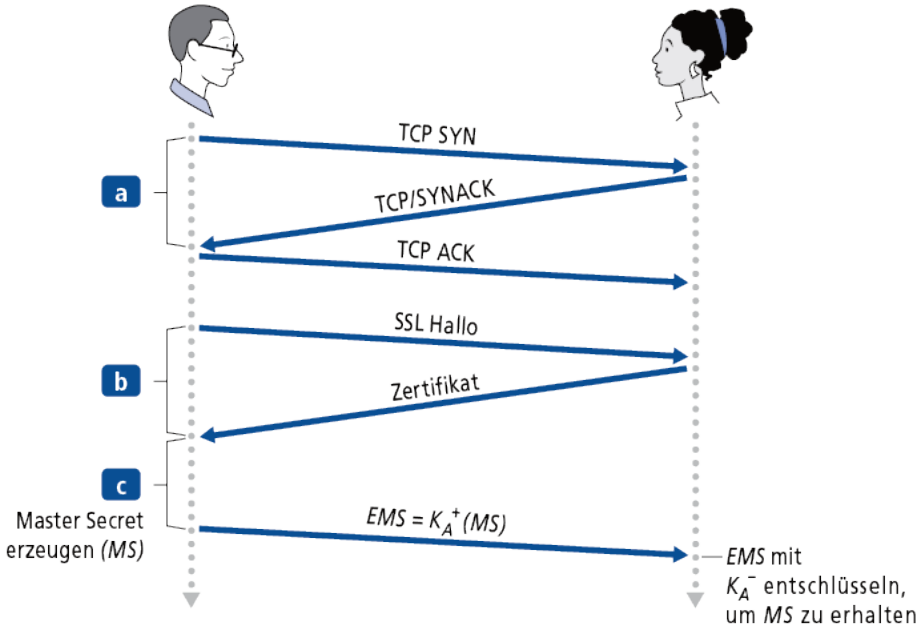
- Transportschichtsicherheit für beliebige TCP-basierte Anwendungen über SSL-Dienste.
  - z.B. zwischen Webbrowser und -server für E-Commerce (**HTTPS**)
- Sicherheitsdienste:
  - Serverauthentifizierung, Datenverschlüsselung, Clientauthentifizierung (optional)
- SSL ist entweder als Teil der Anwendung implementiert, oder die Anwendung macht von den kryptographischen Diensten des Betriebssystems gebrauch
- Seit seiner Standardisierung in IETF wurde SSL in **Transport Layer Security (TLS**) umbenannt → aktuelle Version ist TLS 1.2 in RFC 5246



8.6 SSL – Drei Phasen

1. Handshake:

- Bob baut eine TCP-Verbindung zu Alice auf
- Bob Authentifiziert Alice über ein CA-signiertes Zertifikat
- Bob erzeugt, verschlüsselt (mit Alices öffentl. Schlüssel) und verschickt ein Master Secret an Alice
  - Nonce-Austausch wird hier nicht gezeigt



8.6 SSL – Drei Phasen

2. Schlüsselableitung:

- Alice und Bob verwenden das Master Secret, um vier Schlüssel zu erzeugen:
  - E_B: Schlüssel für Verschlüsselung Bob->Alice
  - E_A: Schlüssel für Verschlüsselung Alice->Bob
  - M_B: MAC-Schlüssel Bob->Alice
  - M_A: MAC-Schlüssel Alice->Bob
- Verschlüsselungs- und MAC-Algorithmen können zwischen Bob und Alice ausgehandelt werden

3. Datentransfer

Kapitel 8 - Netzwerksicherheit

8.7 IPsec: Sicherheit auf der Netzwerkschicht

- Vertraulichkeit:
  - Sender verschlüsselt IP-Payload
- Authentifizierung:
  - Zielhost kann Quell-IP authentifizieren
- Zwei zentrale Protokolle:
  - Authentication-Header-Protokoll (AH)
  - Encapsulation-Security-Protokoll (ESP)
- Zwei Modi:
  - Transport-Modus
  - Tunnel-Modus
- Für AH ebenso wie für ESP: Handshake von Quelle und Ziel
  → logische Netzwerkschicht-Verbindung namens “Security Association (SA)”
- Jede SA ist unidirektional
  - Eindeutig bestimmt durch: Sicherheitsprotokoll (AH oder ESP), Quell-IP, 32-Bit-Verbindungs-ID

8.7 Authentication Header Protokoll (AH)

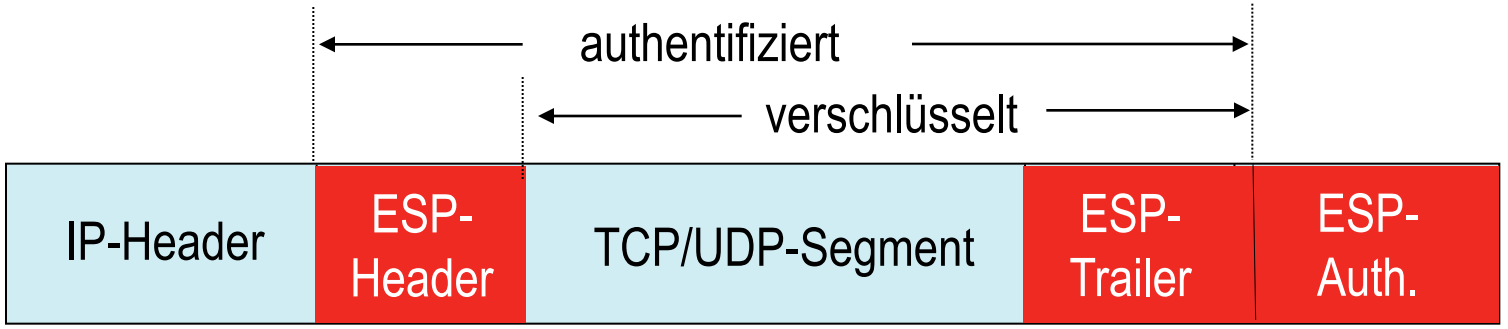
- Bietet Quellauthentifizierung, Datenintegrität, keine Vertraulichkeit
- AH-Header zwischen IP-Header und Datenfeld
- IP-Protokollfeld: 51
- Router unterwegs behandeln das Datagramm wie üblich

AH-Header umfasst:

- Verbindungs-ID
- Authentifizierungsdaten: Von der Quelle signierter Hashwert über das Original-Datagramm
- Next-Header-Feld: Payload-Typ (z.B., TCP, UDP, ICMP)

8.7 ESP-Protokoll

- Bietet Vertraulichkeit, Hostauthentifizierung, Datenintegrität
- Daten und ESP-Trailer sind verschlüsselt
- Next-Header-Feld im ESP-Trailer
- ESP-Authentifizierungsfeld funktioniert ähnlich wie das in AH
- IP-Protokollfeld: 50



Kapitel 8 - Netzwerksicherheit

8.8 IEEE-802.11-Sicherheit

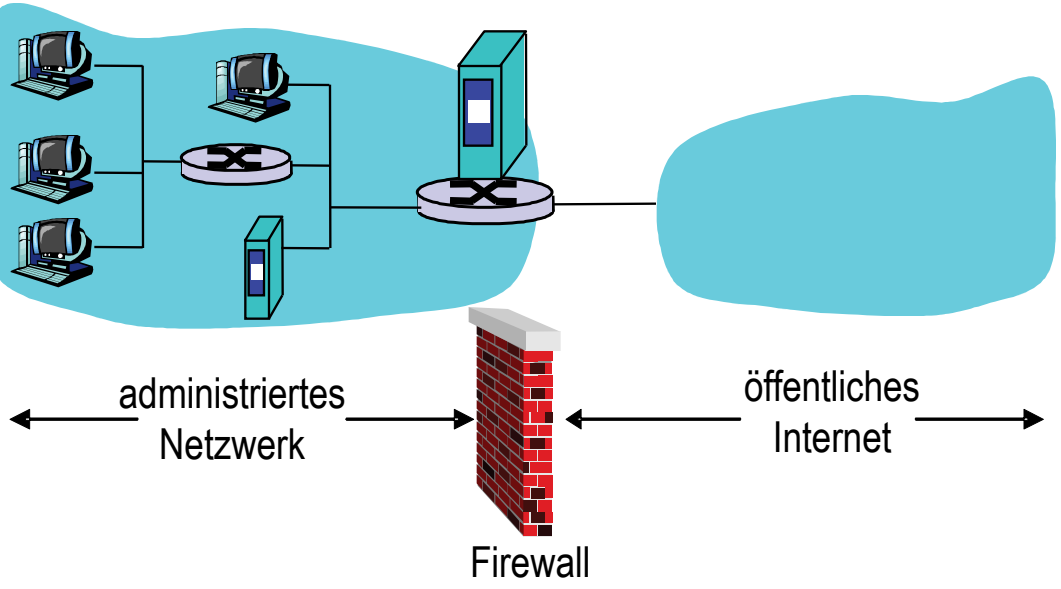
- Viele WLAN Access Points werden befinden sich immer noch unverschlüsselt im Einsatz:
  - → WLAN / Internet Mitbenutzung (inkl. illegaler Aktivitäten!) möglich
  - → Paket-Sniffing ist sehr einfach

- **IEEE 802.11 WLAN sollte man absichern!**
  - Authentifizierung + Verschlüsselung notwendig!
  - Erster Versuch, 802.11 sicher zu machen: Wired Equivalent Privacy (WEP) → Ein Fehlschlag
  - Neuer Anlauf: WPA2 → gilt als sicher!
  - Mehr Details zum Thema im Buch!

Kapitel 8 - Netzwerksicherheit

8.9 Firewalls

Trennt das interne Netz der Organisation vom Rest des Internet; manche Pakete dürfen passieren, andere werden herausgefiltert.



8.9 Firewalls – Warum?

Denial-of-Service-Angriffe abwehren:

- SYN-Flooding: Angreifer baut viele nutzlose TCP-Verbindungen auf, es bleiben keine Ressourcen für die “richtigen” Verbindungen

Illegalen Zugriff auf oder Manipulation von internen Daten verhindern:

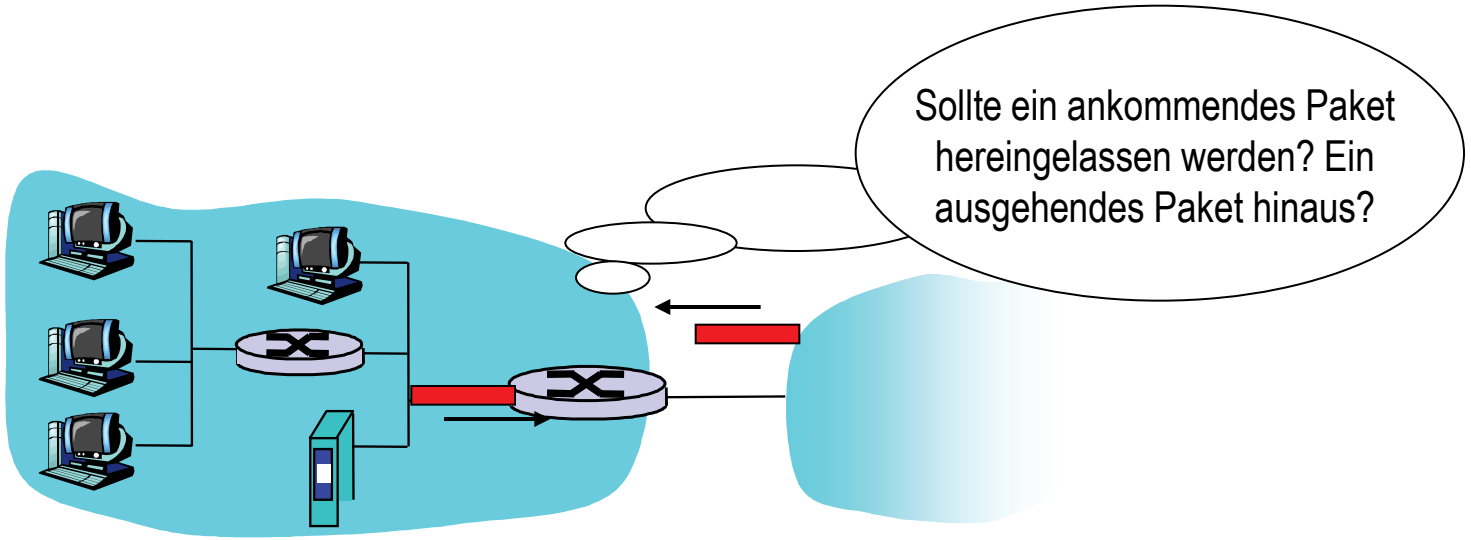
- Ein Angreifer könnte z.B. die Homepage der Organisation „hacken“

→ Nur autorisierten Zugriff auf das interne Netz erlauben (definierte Menge von autorisierten Hosts/Benutzern)

Drei Arten von Firewalls:

- **Zustandslose Paketfilter**
- **Zustandsbasierte Paketfilter**
- Anwendungs-Gateways

8.9 Zustandslose Paketfilter



- Internes Netz ist mit dem Internet über eine **Router-Firewall** verbunden
- Der Router **betrachtet jedes Paket für sich,** die Entscheidung, ob weitergeleitet wird, basiert auf :
  - Quell- und Ziel-IP
  - TCP/UDP-Quell- und Zielportnummern
  - ICMP-Nachrichtentyp
  - TCP-SYN- und ACK-Bits

8.9 Zustandslose Paketfilter

Beispiel 1:

Blockiere eingehende und ausgehende Datagramme mit IP-Protokollfeld 17 (UDP) und entweder Quell- oder Ziel-Port 23 (Telnet).

- Alle ein- oder ausgehenden UDP-Flows und Telnet-Verbindungen werden blockiert.

Beispiel 2:

Eingehende TCP-Segmente mit ACK=0 blockieren.

- Hält externe Hosts davon ab, zu internen Hosts TCP-Verbindungen aufzubauen, erlaubt es aber internen Hosts, nach Verbindungen nach außen zu initiieren.

8.9 Zustandslose Paketfilter

Ziel	Firewall-Regel
Kein Web-Zugriff nach außen.	Alle eingehenden Pakete zu jeder IP-Adresse und Port 80 verwerfen.
Keine eingehenden TCP-Verbindungen, außer sie sprechen den eigenen Webserver an.	Alle eingehenden TCP-SYN-Pakete verwerfen, außer sie gehen an IP-Adresse 130.207.244.203, Port 80.
Vermeiden, dass Web-Radio die gesamte Bandbreite belegt.	Alle eingehenden UDP-Pakete verwerfen, außer DNS Verkehr.
Verhindern, dass das eigene Netz mit Traceroute untersucht wird.	Ausgehende ICMP-TTL-Expired-Pakete verwerfen.

8.9 Access Control Lists

ACL: Liste von Regeln, die von oben nach unten auf eingehende Pakete angewandt wird → Paare von Aktionen und Kriterien.

Aktion	Quell-IP	Ziel-IP	Protokoll	Quell-Port	Ziel-Port	Flags
erlaube	222.22/16	nicht in 222.22/16	TCP	> 1023	80	egal
erlaube	nicht in 222.22/16	222.22/16	TCP	80	> 1023	ACK
erlaube	222.22/16	nicht in 222.22/16	UDP	> 1023	53	---
erlaube	nicht in 222.22/16	222.22/16	UDP	53	> 1023	----
verbiete	alle	alle	alle	alle	alle	alle

8.9 Zustandsbehaftete Paketfilter

Zustandslose Paketfilter sind oft unbeholfen:

- Pakete werden zugelassen, die “keinen Sinn machen”, z.B. Quell-Port 80, ACK-Flag gesetzt, obwohl keine TCP-Verbindung existiert:

Aktion	Quell-IP	Ziel-IP	Protokoll	Quell-Port	Ziel-Port	Flags
erlaube	nicht in 222.22/16	222.22/16	TCP	80	> 1023	ACK

Zustandsbehafteter Paketfilter: verfolgt den Zustand jeder TCP-Verbindung

- Liest den Verbindungsauf- (SYN) und –abbau (FIN) mit: kann bestimmen, ob ein- und ausgehende Pakete “sinnvoll” sind
- Timeout für inaktive Verbindungen in der Firewall: alte Verbindungen nicht mehr durchlassen

8.9 Zustandsbehaftete Paketfilter

ACL wird erweitert um anzuzeigen, ob es notwendig ist, die Zustandstabelle der Verbindungen ebenfalls zu prüfen.

Aktion	Quell-IP	Ziel-IP	Protokoll	Quell-Port	Ziel-Port	Flags	Verbindung prüfen
erlaube	222.22/16	nicht in 222.22/16	TCP	> 1023	80	egal	
erlaube	nicht in 222.22/16	222.22/16	TCP	80	> 1023	ACK	JA
erlaube	222.22/16	nicht in 222.22/16	UDP	> 1023	53	---	
erlaube	nicht in 222.22/16	222.22/16	UDP	53	> 1023	----	JA
verbiete	alle	alle	alle	alle	alle	alle	

8.9 Intrusion-Detection-Systeme

- Paketfilter:
  - Arbeiten nur auf den TCP/IP-Headern
  - Daten unterschiedlicher Sitzungen können nicht korreliert werden

- **IDS: Intrusion-Detection-System**
  - *Deep Packet Inspection:* betrachtet den Paketinhalt (vergleicht z.B., ob im Paket Zeichenketten vorkommen, die in einer Datenbank bekannter Angriffe und Viren vorkommen)
  - Korrelation mehrerer Pakete, z.B. bei Port-Scanning