Netzwerktechnologien
3 VO

Dr. Ivan Gojmerac
ivan.gojmerac@univie.ac.at

12. Vorlesungseinheit, 12. Juni 2013

Bachelorstudium Medieninformatik
SS 2013

Kapitel 8 - Netzwerksicherheit

8.1 Sicherheitsanforderungen in Netzen

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

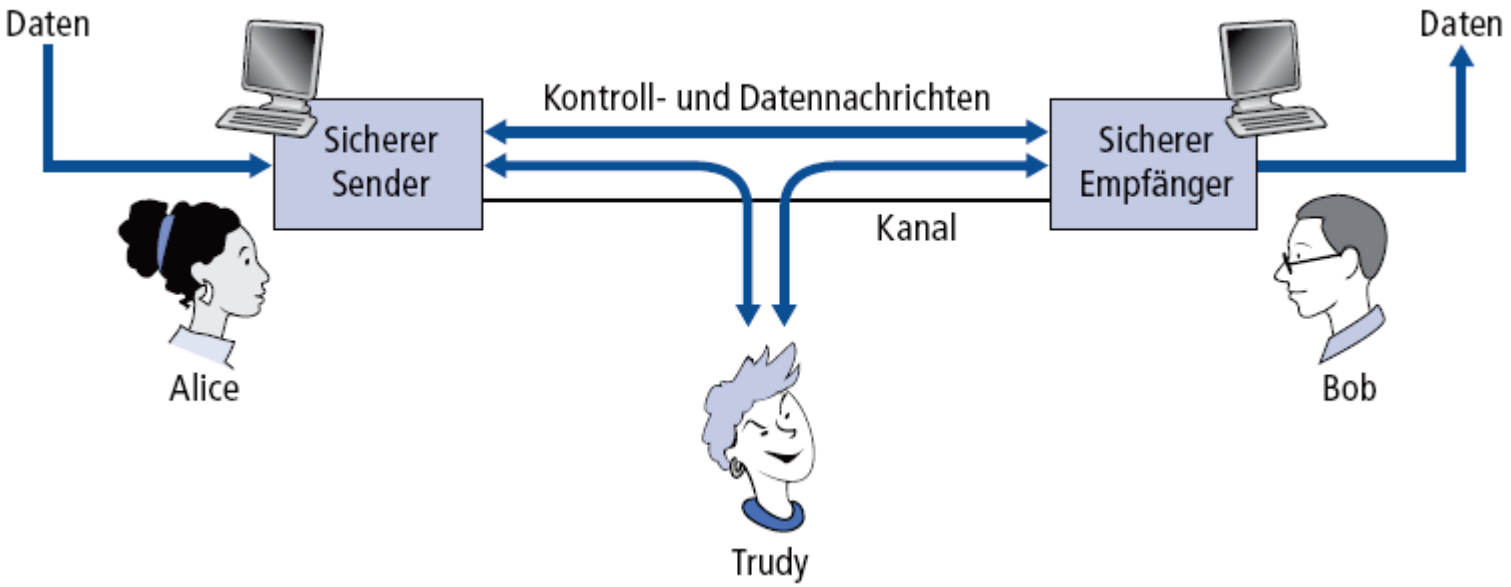
Authentifizierung: Sender und Empfänger wollen gegenseitig ihre Identität sicherstellen

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
- Usw.

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**
- **Übernahme einer bestehenden Verbindung** indem der Sender oder Empfänger entfernt wird und sich der Angreifer an seiner Stelle platziert
- **Denial of Service:** Der Angreifer verhindert, dass andere einen Dienst nutzen können (z.B. durch Überlasten von Ressourcen)
- Usw.

Kapitel 8 - Netzwerksicherheit

8.1 Was ist Netzwerksicherheit?

8.2 Grundlagen der Kryptographie

8.3 Endpunktauthentifizierung

8.4 Nachrichtenintegrität

8.5 Absichern von E-Mail

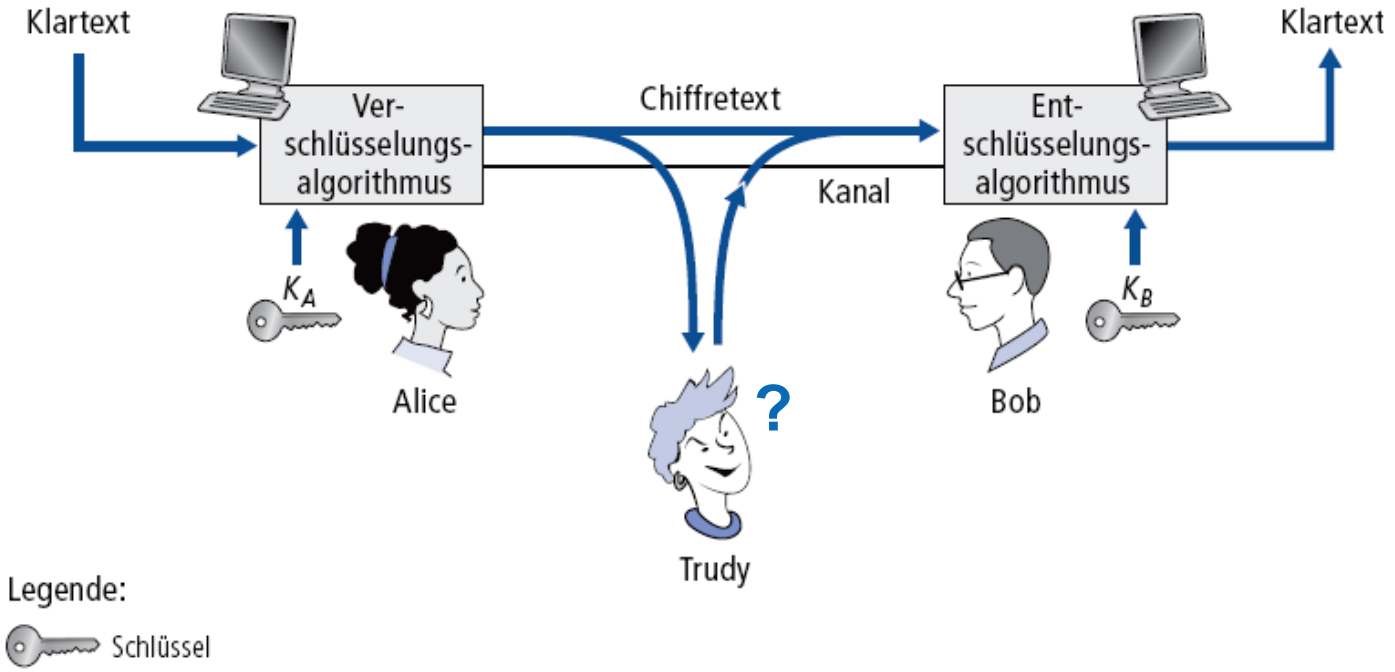
8.6 Absichern von TCP-Verbindungen: SSL

8.7 Sichern auf der Netzwerkschicht: Ipsec und VPNs

8.8 Sicherheit von Wireless LAN

8.9 Operative Sicherheit: Firewalls und IDS

8.2 Terminologie der Kryptographie



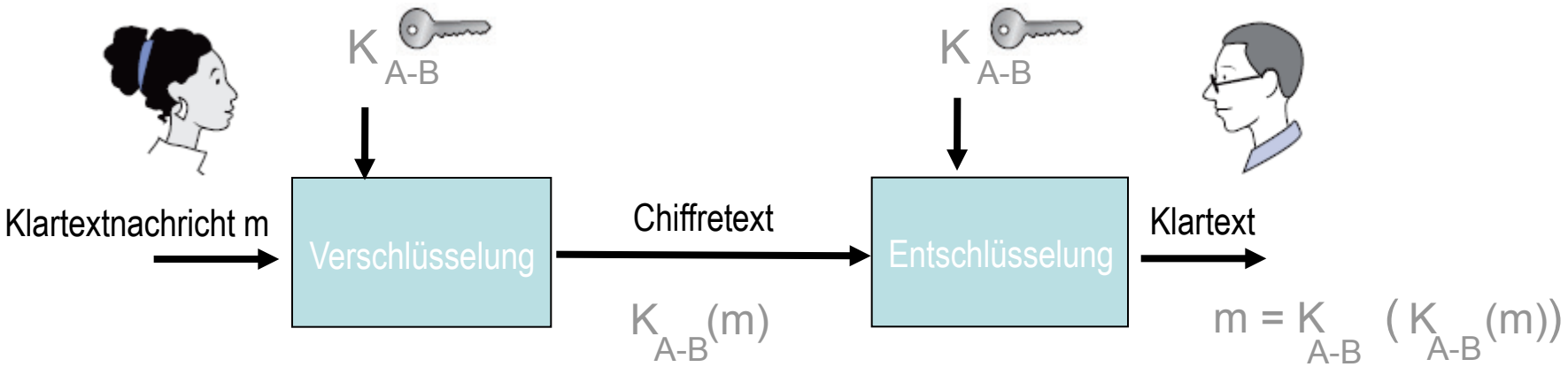
Symmetrische Kryptographie: Sender- und Empfängerschlüssel sind *identisch*

Public-Key-Kryptographie: Es existieren jeweils Paare von öffentlichen und privaten Schlüsseln

8.2 Kryptographie mit symmetrischen Schlüsseln

Kryptographie mit symmetrischen Schlüsseln: Bob and Alice kennen denselben (symmetrischen) Schlüssel $\mathbf{K}_{A-B}$

- Der Schlüssel könnte zum Beispiel das Ersetzungsmuster der monoalphabetischen Chiffre (siehe nächste Folie) sein



8.2 Kryptographie mit symmetrischen Schlüsseln

Ersetzungschiffre: Eine Sache durch eine andere ersetzen

– *Monoalphabetische Chiffre*: Einen Buchstaben durch einen anderen ersetzen

Klartext Alphabet aus 26 Zeichen: **`abcdefghijklmnopqrstuvwxyz`**

Chiffretext Alphabet aus 26 Zeichen: **`mnbvcxzasdfghjklpoiuytrewq`**

z.B.:

Klartext: **`bob, ich liebe dich. alice`**

Chiffretext: **`nkn, sba pscbc vsba. mgsbc`**

8.2 Kryptographie mit symmetrischen Schlüsseln

Ein weiterentwickelter Verschlüsslungsversuch:

- n Ersetzungschiffre (M_1, M_2... , M_n)...
- ...die periodisch wechseln:
  - z.B. bei n=4: M_1,M_3,M_4,M_3,M_2 ; M_1,M_3,M_4,M_3,M_2 ; …
- Wechsle bei jedem neuen Klartextsymbol zum nächsten Ersetzungschiffre
  - Bei der oben definierten Chiffrefolge:
    Beispiel *"**Hut**"*: ***H*** mit M_1, ***u*** mit M_3, ***t*** mit M_4 verschlüsseln

→ Bei einem Verschlüsselungsschlüssel:

- n Ersetzungschiffre die periodisch wechseln
- Nun kann der Schlüssel aus mehr als einem n Bit langen Muster bestehen

8.2 Symmetrische Kryptographie: DES Algorithmus

DES: Data Encryption Standard

- US-Verschlüsselungsstandard [NIST 1993]
- Symmetrische 56-Bit-Schlüssel, 64 Bit lange Klartext-Eingaben
- Blockchiffrierung mit Cipher Block Chaining (CBC)

Aber wie sicher ist DES?

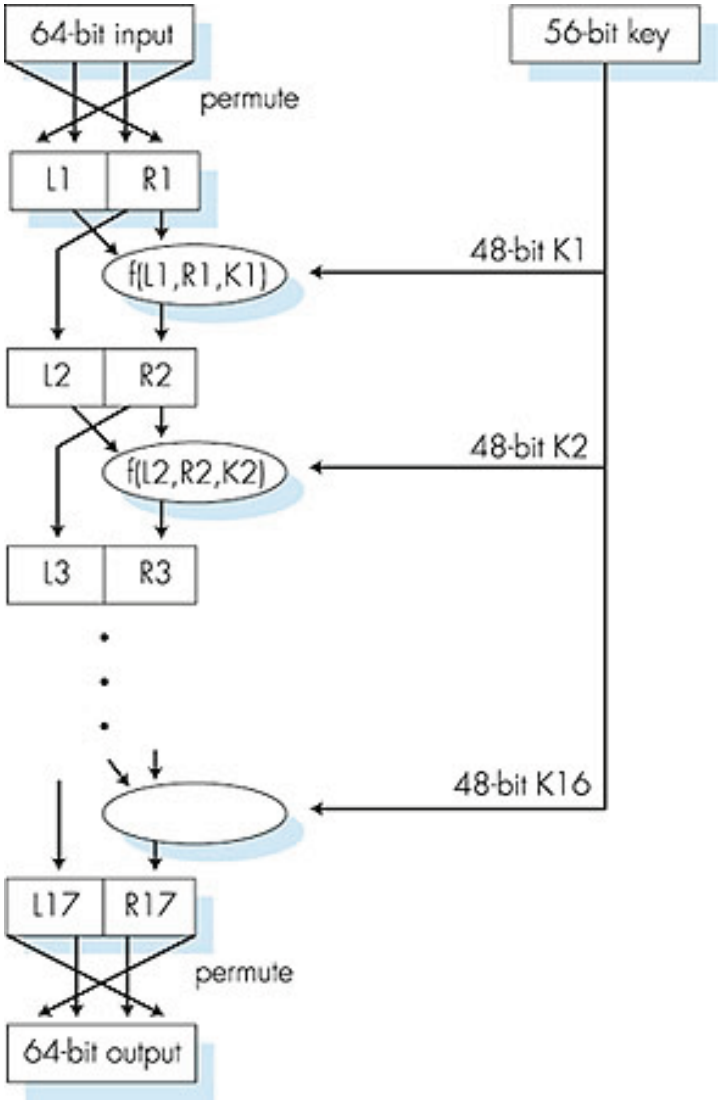
- DES Test: Ein Satz der mittels 56-Bit-Schlüssel kodiert wurde wird mit Brute-Force-Entschlüsselung in weniger als einem Tag entschlüsselt
- Um DES sicherer zu machen:
  - 3DES: Verschlüssle drei Mal mit drei unterschiedlichen Schlüsseln

- Problem: DES ist nicht ausreichend sicher → [RFC 4772]

8.2 Symmetrische Kryptographie: DES Algorithmus

DES Vorgangsweise

- Erste Permutation
- 16 identische "Runden" in denen die Funktion jedes Mal mit einem anderen 48 Bit Schlüssel angewendet wird
- Letzte Permutation

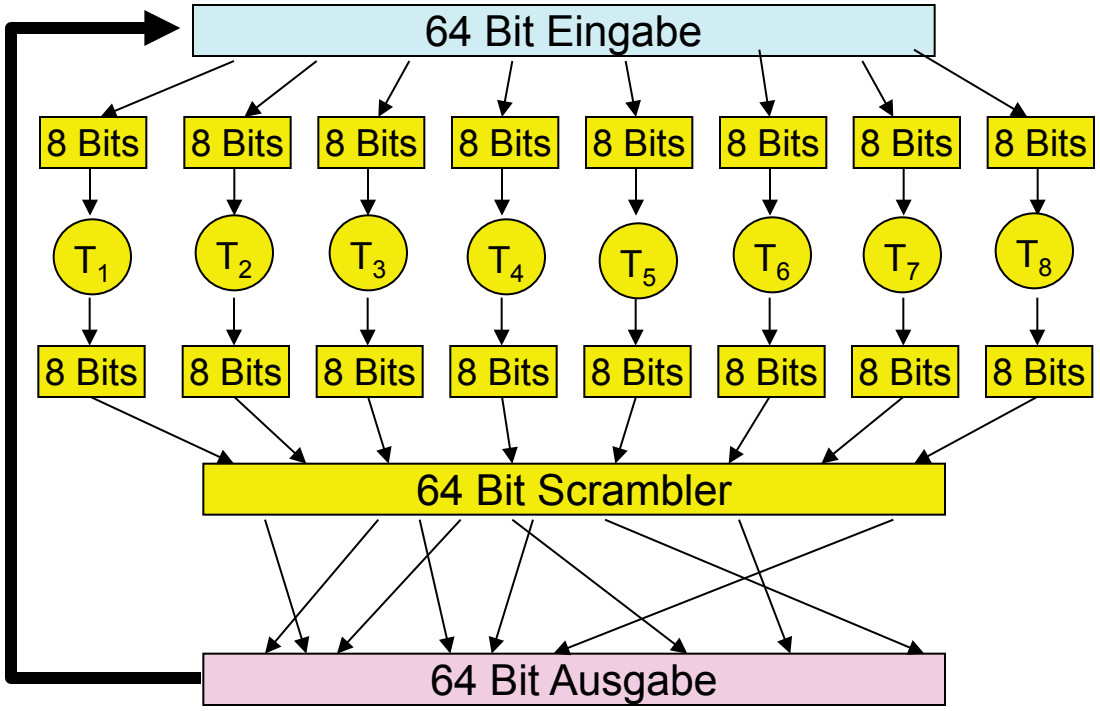


8.2 AES: Advanced Encryption Standard

- Symmetrischer NIST-Standard der DES 2001 ersetzt hat
- 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

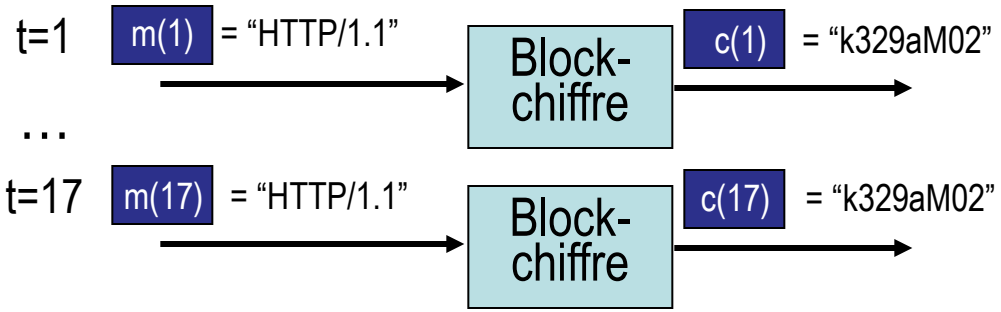
Schleife über n Runden



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

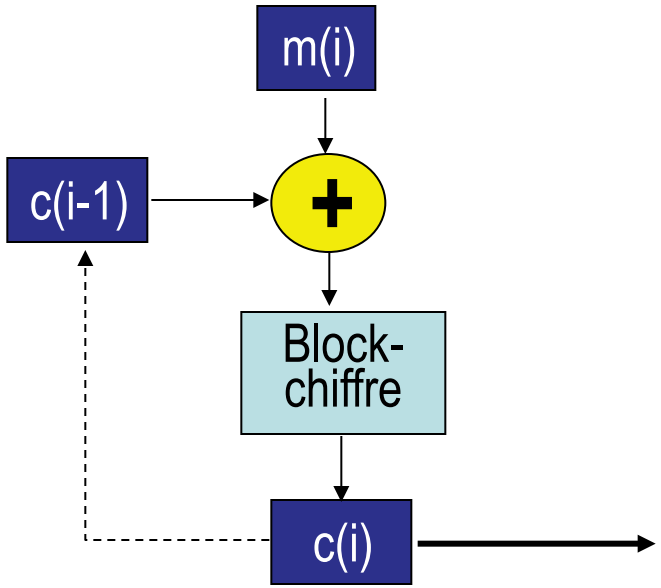
8.2 Cipher Block Chaining

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

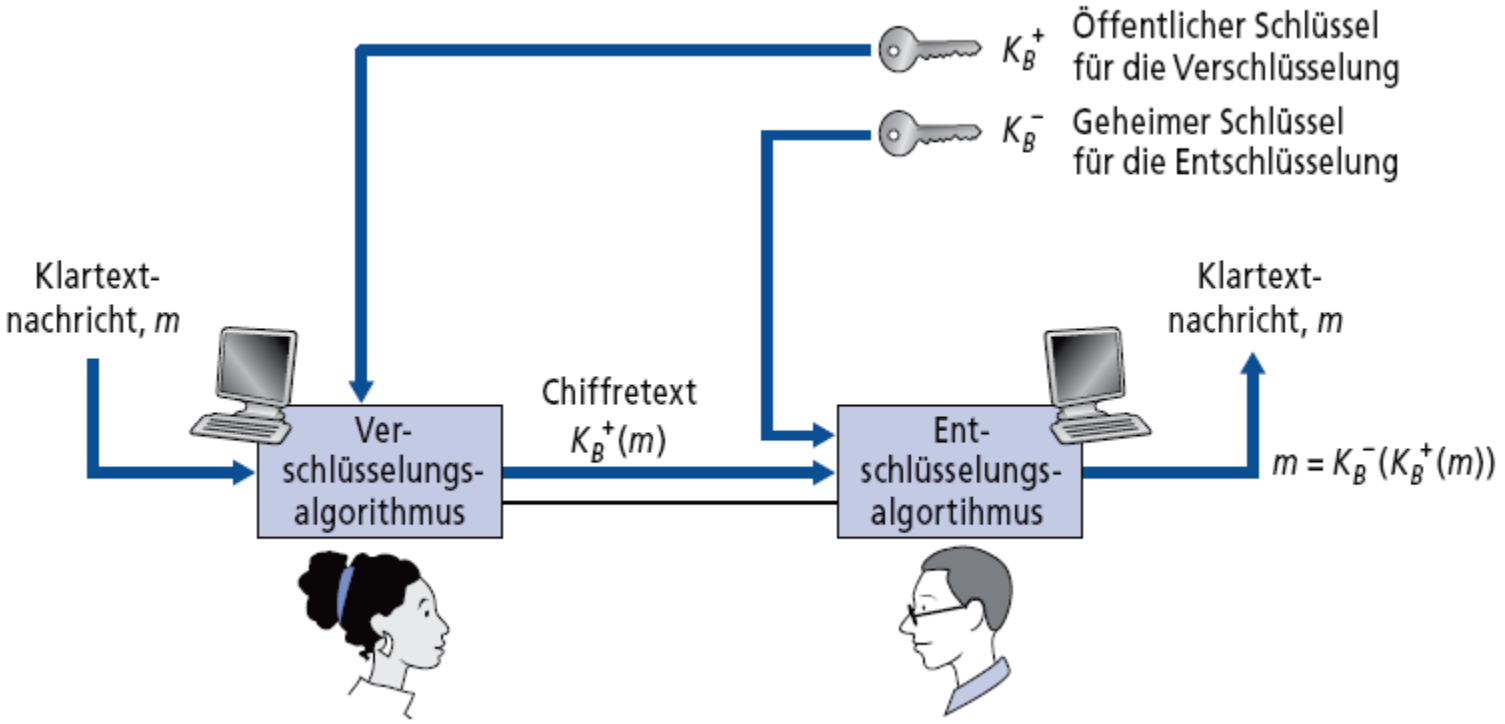
Symmetrische Kryptographie:

- Erfordert, dass Sender und Empfänger ein gemeinsamer geheimer Schlüssel bekannt ist.
- Problem der symmetrischen Kryptographie: Wie kann man sich zu Anfang überhaupt auf einen Schlüssel einigen (vor allem vor der ersten Verbindung)?

Public-Key-Kryptographie

- Radikal anderer Ansatz [Diffie-Hellman76, RSA78]
- Sender, Empfänger kennen **keinen gemeinsamen** geheimen Schlüssel
- **Public** Verschlüsselungsschlüssel, den **alle** kennen
- **Geheimen** Entschlüsselungsschlüssel kennt nur der Empfänger

8.2 Public Key Kryptographie



8.2 Public Key Algorithmen

Anforderungen:

Der Empfänger benötigt $K_B^+()$ und $K_B^-()$ um die Nachricht m zu entschlüsseln:

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

Aus dem öffentlichen Schlüssel K_B^+ soll der private Schlüssel K_B^- nicht errechenbar sein!

(→ **RSA:** Algorithmus von Rivest, Shamir, Adleman)

8.2 Public Key – Vorraussetzung: Modulo-Arithmetik

→ **x mod n** = Rest der übrig bleibt wenn x durch n geteilt wird.

Regeln:

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$$

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$$

$$[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$$

also ist $(a \bmod n)^d \bmod n = a^d \bmod n$

Beispiel:

$x=14,\ n=10,\ d=2$

$$(x \bmod n)^d \bmod n = x^d \bmod n = 14^2 \bmod 10 = 196 \bmod 10 = \underline{6}$$

Test :

$$(x \bmod n)^d \bmod n = (14 \bmod 10)^2 \bmod 10 =$$

$$4^2 \bmod 10 = 16 \bmod 10 = \underline{6}$$

8.2 RSA – Einleitung

- Eine Nachricht ist nur ein Bitmuster
- Ein Bitmuster kann eindeutig durch eine Integer Zahl repräsentiert werden
- Also ist der Vorgang um eine Nachricht zu verschlüsseln gleich dem Vorgang um eine Zahl zu verschlüsseln!

Beispiel:

m = **10010001** (Diese Nachricht kann eindeutig durch die Zahl 145 repräsentiert werden)

→ Um m zu verschlüsseln, verschlüsseln wir die entsprechende Zahl, was uns wiederrum eine neue Zahl gibt (den Chiffretext).

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 e und d 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!

Das lässt sich direkt aus der Modulo Arithmetik ableiten:

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n\end{aligned}$$

8.2 RSA in der Praxis – Session Schlüssel

- Das Potenzieren bei RSA braucht viel Rechenleistung
- DES ist 100 Mal schneller als RSA
- Also:
  - Verwende **Public Key Kryptographie** um eine **sichere Verbindung** zwischen den Hosts herzustellen
  - Dann verwende **symmetrische Kryptographie** (z.B. DES) um einen Session Schlüssel zu erstellen mit dem die **Daten verschlüsselt** werden

<u>Beispiel</u>

→ Session Schlüssel K_S

1. Bob und Alice verwenden RSA um einen symmetrischen Schlüssel K_S auszutauschen
2. Wenn beide den Schlüssel K_S haben verwenden sie symmetrische Kryptographie

Kapitel 8 - Netzwerksicherheit

8.1 Was ist Netzwerksicherheit?
8.2 Grundlagen der Kryptographie
8.3 Endpunktauthentifizierung
8.4 Nachrichtenintegrität
8.5 Absichern von E-Mail
8.6 Absichern von TCP-Verbindungen: SSL
8.7 Sichern auf der Netzwerkschicht: Ipsec und VPNs
8.8 Sicherheit von Wireless LAN
8.9 Operative Sicherheit: Firewalls und IDS

8.3 Authentifizierung

WICHTIGE ANMERKUNG: *Die Folien 29-40 enthalten zu didaktischen Zwecken erfundene (d.h. fiktive) Authentifizierungsprotokolle **ap1.0** bis **ap5.0**.*

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

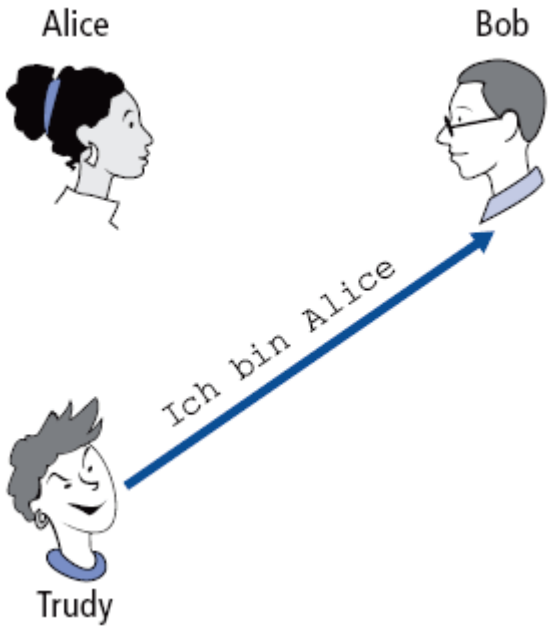
Protokoll ap1.0: Alice sagt "Ich bin Alice"

↯ Angriffsszenario?

8.3 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.

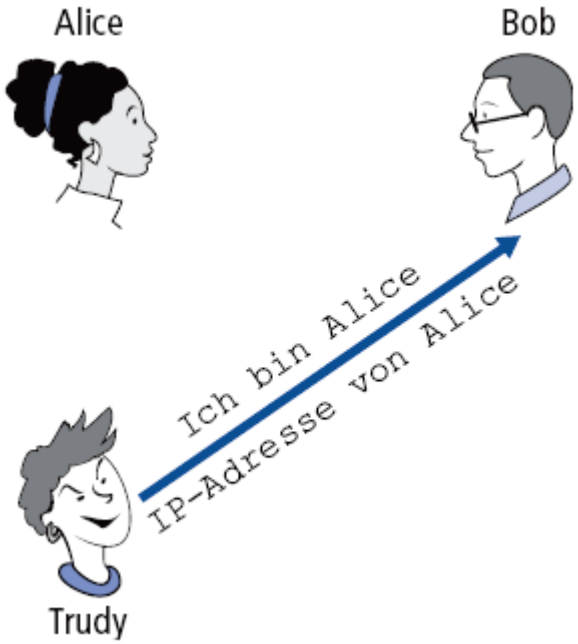
8.3 Authentifizierung

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

↯ Angriffsszenario?

8.3 Authentifizierung

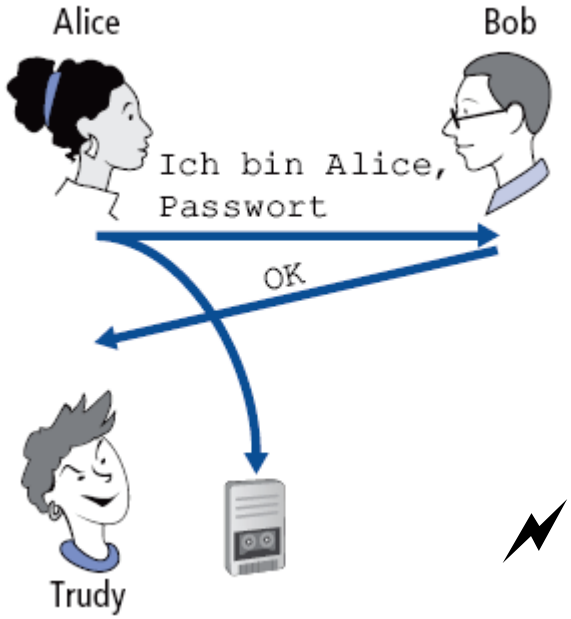
Protokoll ap2.0: Alice sagt “Ich bin Alice” in einem IP-Paket, das ihre Quell-IP enthält



Trudy kann ein Paket mit gefälschter Absenderadresse erzeugen

8.3 Authentifizierung

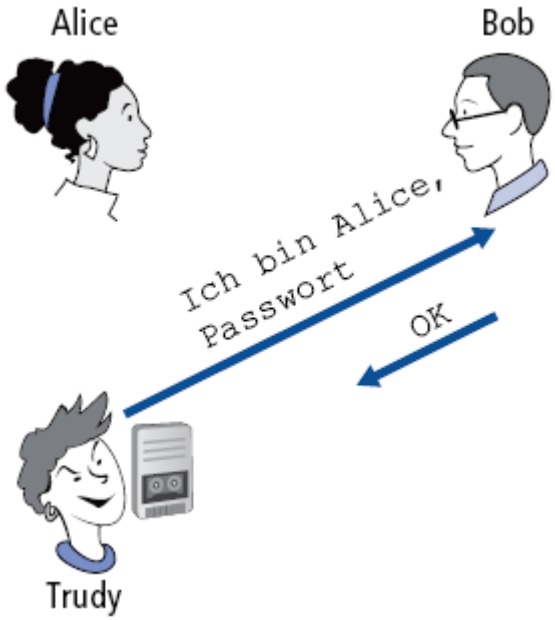
Protokoll ap3.0: Alice sagt “Ich bin Alice” und schickt ihr geheimes Passwort als “Beweis” mit.



↯ Angriffsszenario?

8.3 Authentifizierung

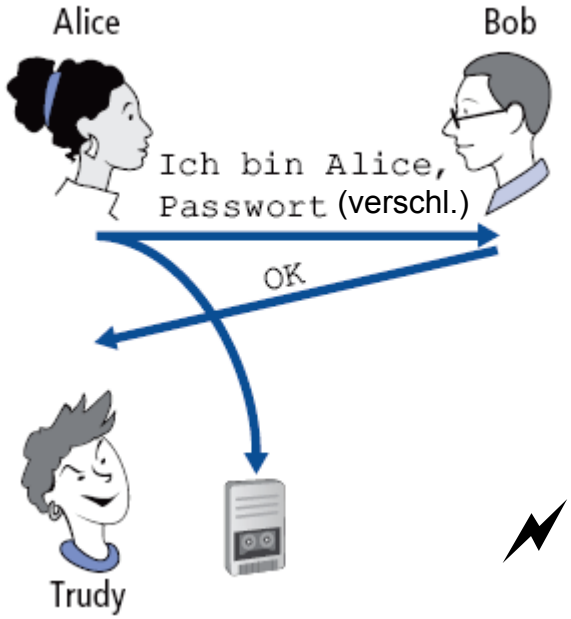
Protokoll ap3.0: Alice sagt “Ich bin Alice” und schickt ihr geheimes Passwort als “Beweis” mit.



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

8.3 Authentifizierung

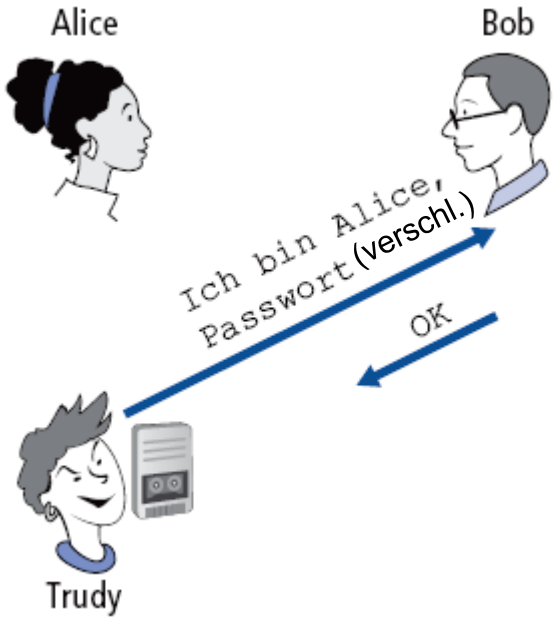
Protokoll ap3.1: Alice sagt “Ich bin Alice” und schickt ihr verschlüsseltes geheimes Passwort als “Beweis” mit.



↯ Angriffsszenario?

8.3 Authentifizierung

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



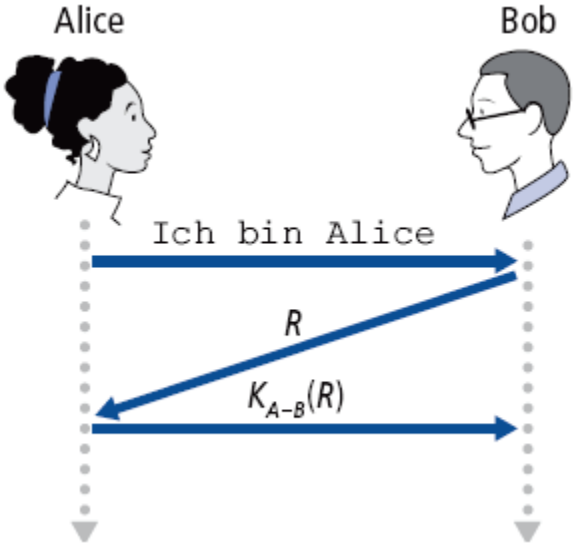
Aufzeichnen und wiederholen funktioniert immer noch!

8.3 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.



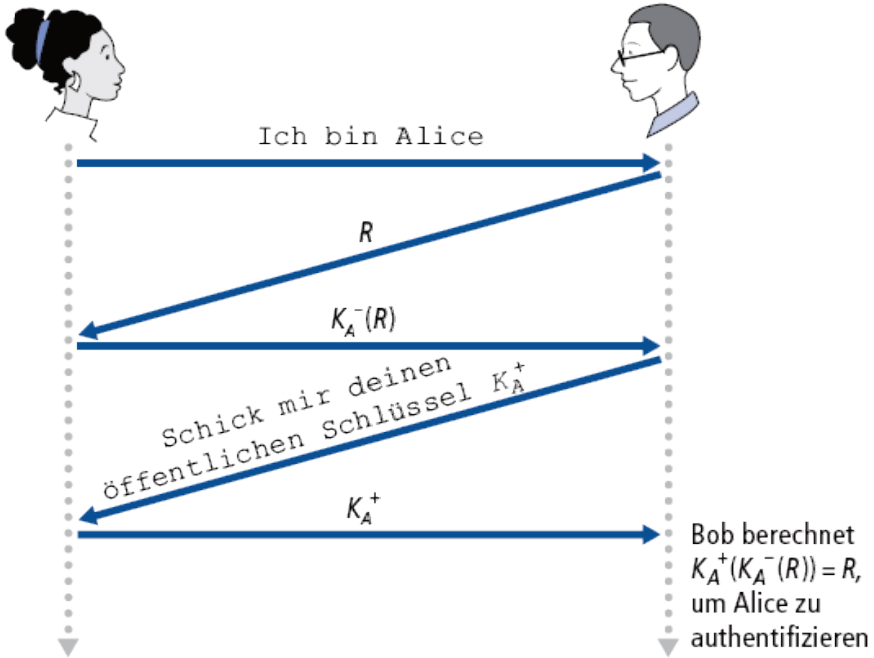
Nur Alice kennt den richtigen Schlüssel, also muss das Alice sein!

→ Fehler, Nachteile?

8.3 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

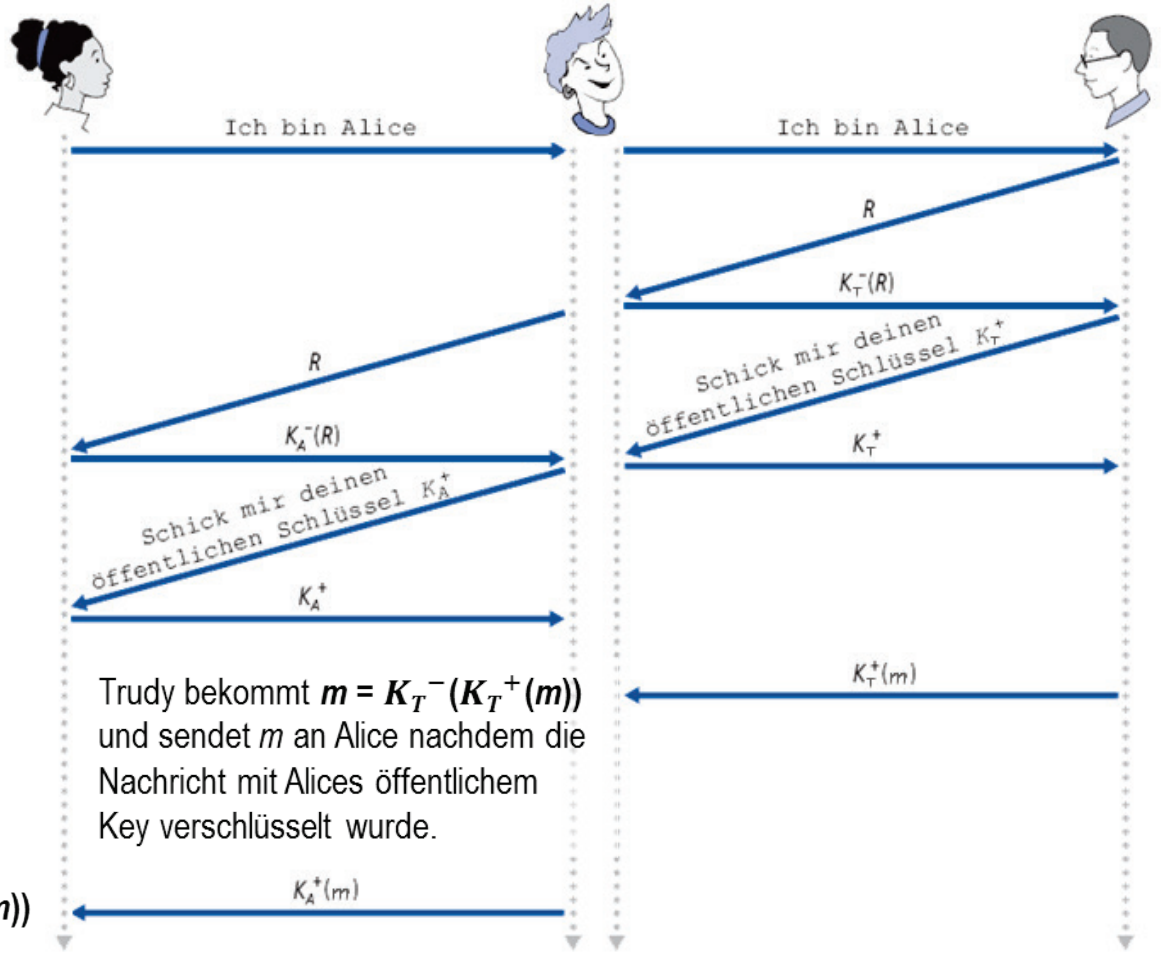
$$K_A^+ (K_A^-(R)) = R$$

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

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

8.3 Ap5.0 Sicherheitslücke

Man in the Middle Attack: Trudy gibt sich Alice gegenüber als Bob aus und gibt sich Bob gegenüber als Alice aus.



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

8.3 Ap5.0 Sicherheitslücke

Man in the Middle Attack: Trudy gibt sich Alice gegenüber als Bob aus und gibt sich Bob gegenüber als Alice aus.

Schwierig zu entdecken:

- Bob empfängt ja alles was Alice sendet und andersrum.

→ Das Problem ist, dass Trudy AUCH alle Nachrichten empfängt!

Kapitel 8 - Netzwerksicherheit

8.4 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.

Digitale Unterschrift:

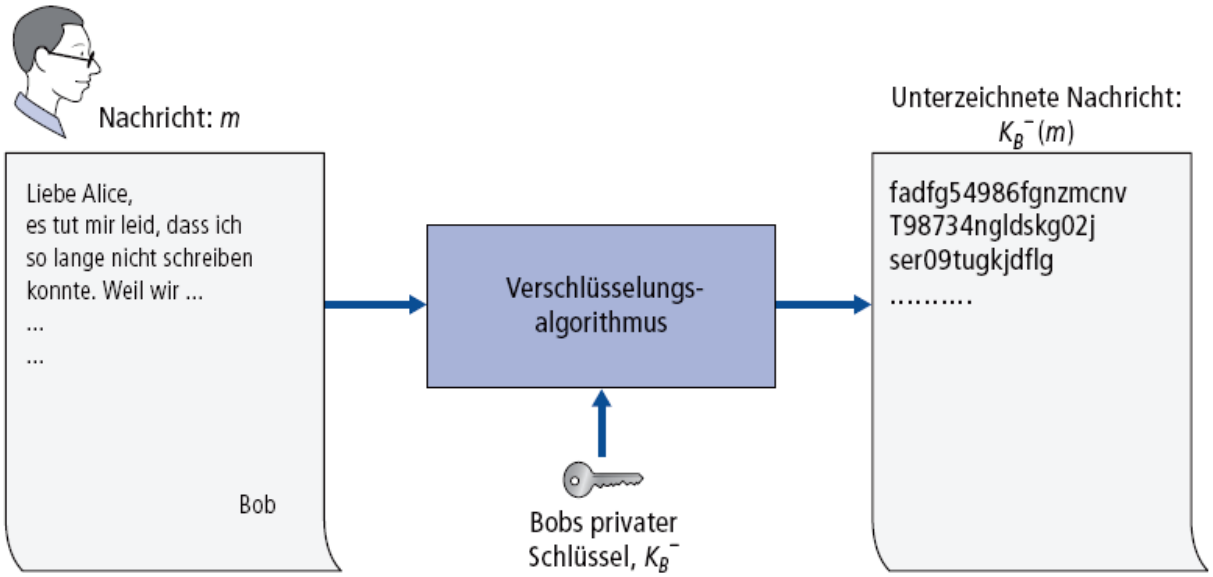
Kryptographische Technik, die der eigenhändigen “analogen” Unterschrift entspricht

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

8.4 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



8.4 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.

8.4 Hashwerte

Es braucht viel Rechenleistung um eine lange Nachricht mit einem öffentlichen Schlüssel zu verschlüsseln.

Deshalb neues Ziel:

Einen digitalen "Fingerabdruck" erstellen können, der schnell zu berechnen ist und einer vordefinierten Länge entspricht.

→ Wende eine Hashfunktion H auf die Nachricht m an um einen Hashwert H(m) mit fixer Länge zu erhalten.

8.4 Hashwerte

Kryptographische Hashfunktion:

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

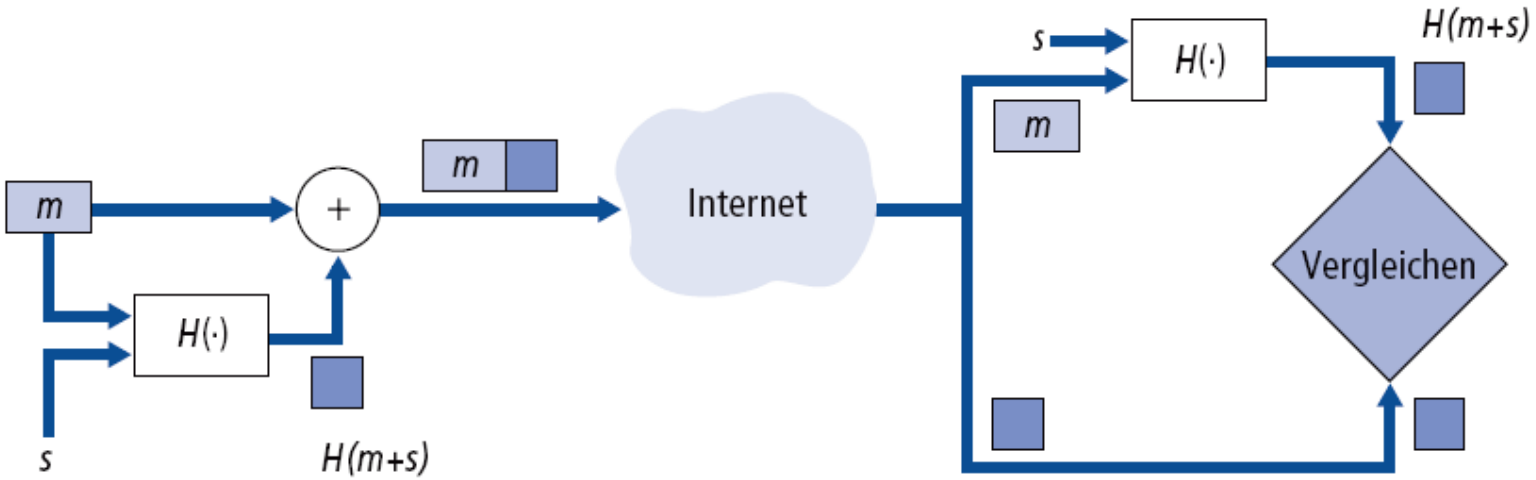
Eigenschaften von Hashfunktionen:

- Mehrfachzuordnungen (Many-to-One)
- Ergibt Hashwert fixer Länge (Fingerabdruck)
- Ein Hashwert x kann nicht eindeutig auf eine Nachricht m zurückgeführt werden im Sinne von x = H(m)

8.4 Hashalgorithmen

- MD5 [RFC 1321]
  - Berechnet einen 128Bit-Hashwert in 4 Schritten
  - Gilt inzwischen nicht mehr als sicher, da es relativ leicht möglich ist andere Nachrichten zu erzeugen, die den gleichen MD5-Hashwert ergeben
- SHA-1
  - US Standard [NIST, FIPS PUB 180-1]
  - Ergibt einen 160Bit-Hashwert
  - Wiederstandsfähiger gegen Brute-Force-Angriffe zur Ermittlung anderer Nachrichten mit dem selben SHA-1-Hashwert → Können aber trotzdem mit 2^{63} Berechnungen (mit Hochleistungsrechnern möglich) ermittelt werden
- SHA-3 (Keccak)
  - 2012 wurde Keccak als SHA-3 von NIST standardisiert
  - Erzeugt einen Hashwert der gewünschten Länge
  - Weitere Informationen auf der Keccak Website

8.4 Message Authentication Code (MAC)



Legende:

m = Nachricht

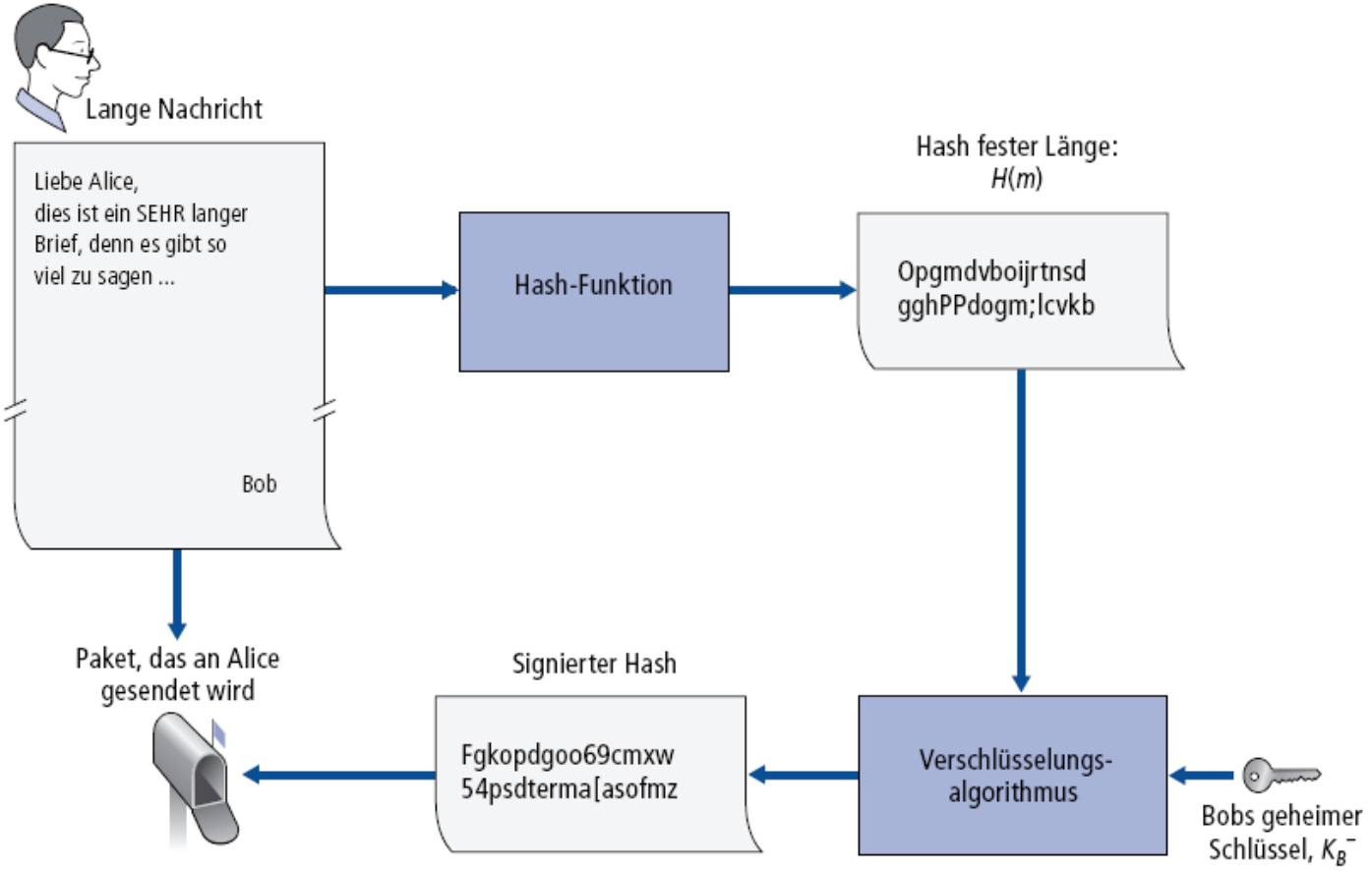
s = gemeinsames Geheimnis

Für MACs werden meistens die Hash-Funktionen von der vorigen Folie verwendet!

8.4 Digitale Unterschrift: Signierter Hash

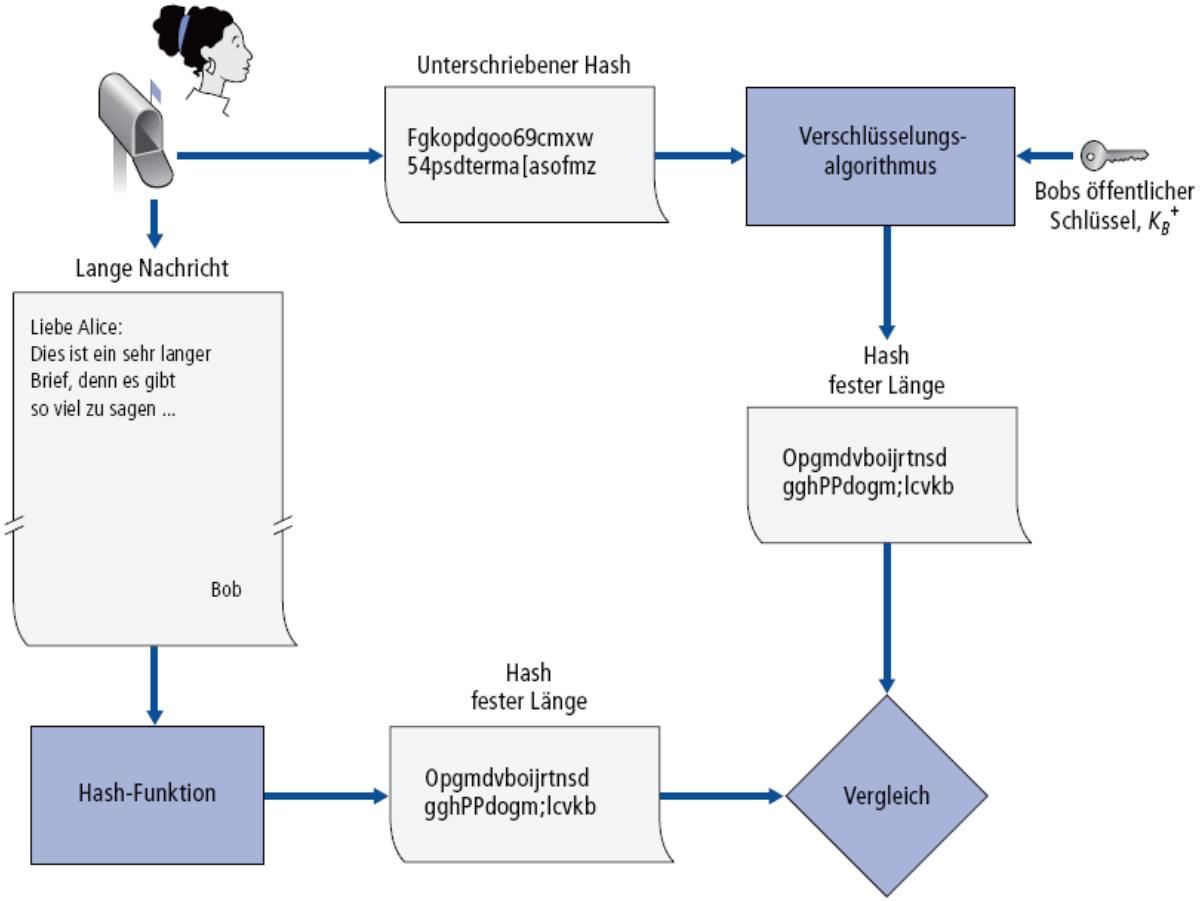
Das Signieren eines Hash-Wertes ist wesentlich weniger rechenaufwendig als das Signieren einer möglicherweise sehr großen Originalnachricht.

Bob verschickt eine digital signierte Nachricht:



8.4 Digitale Unterschrift: Signierter Hash

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



8.4 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:

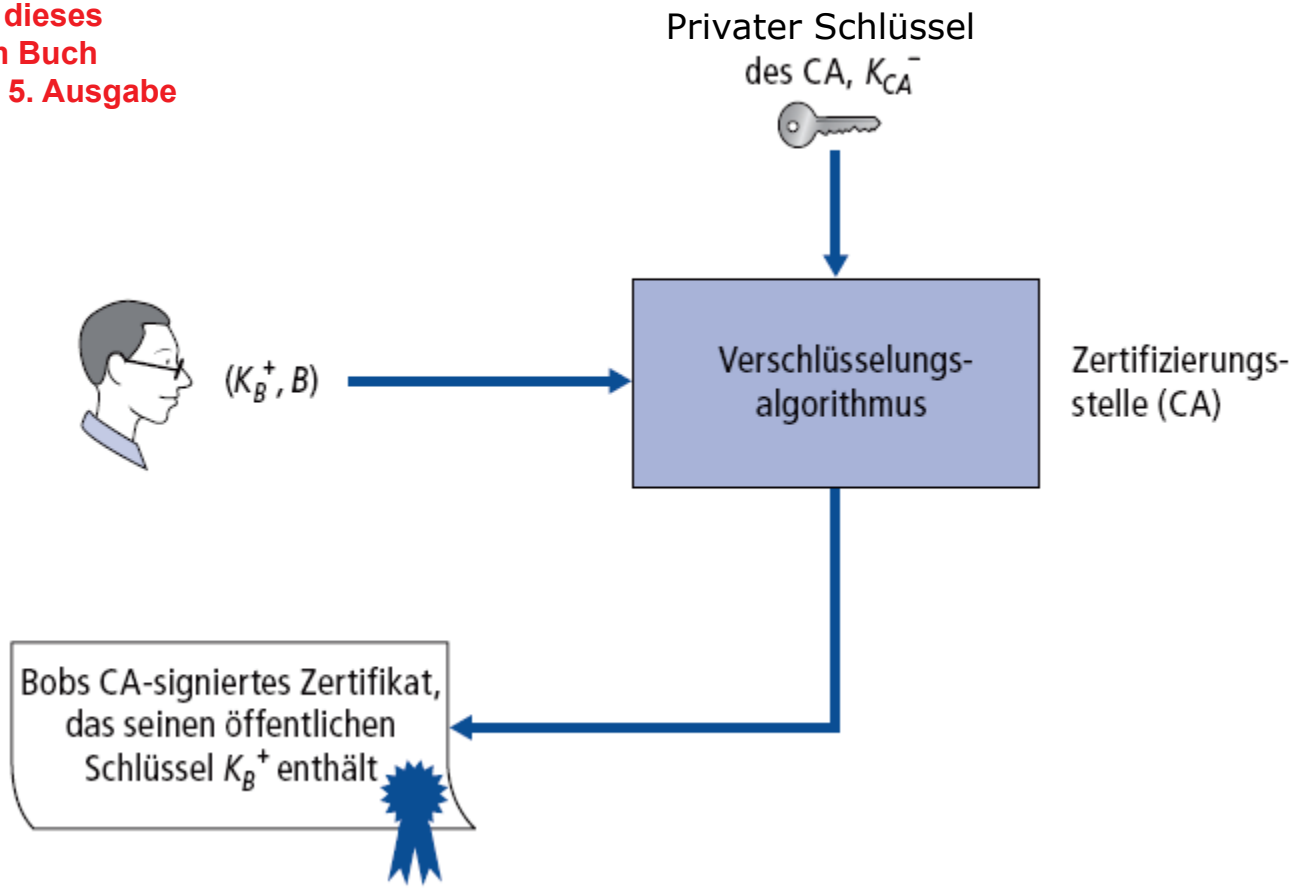
- Vertrauenswürdige Zertifizierungsstelle (Certification Authority, CA)

8.4 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.”

8.4 Zertifizierungsstelle

- **Die Version dieses Bildes ist im Buch bis inkl. der 5. Ausgabe fehlerhaft!**



8.4 Zertifizierungsstelle

Wenn Alice Bobs öffentlichen Schlüssel benötigt:

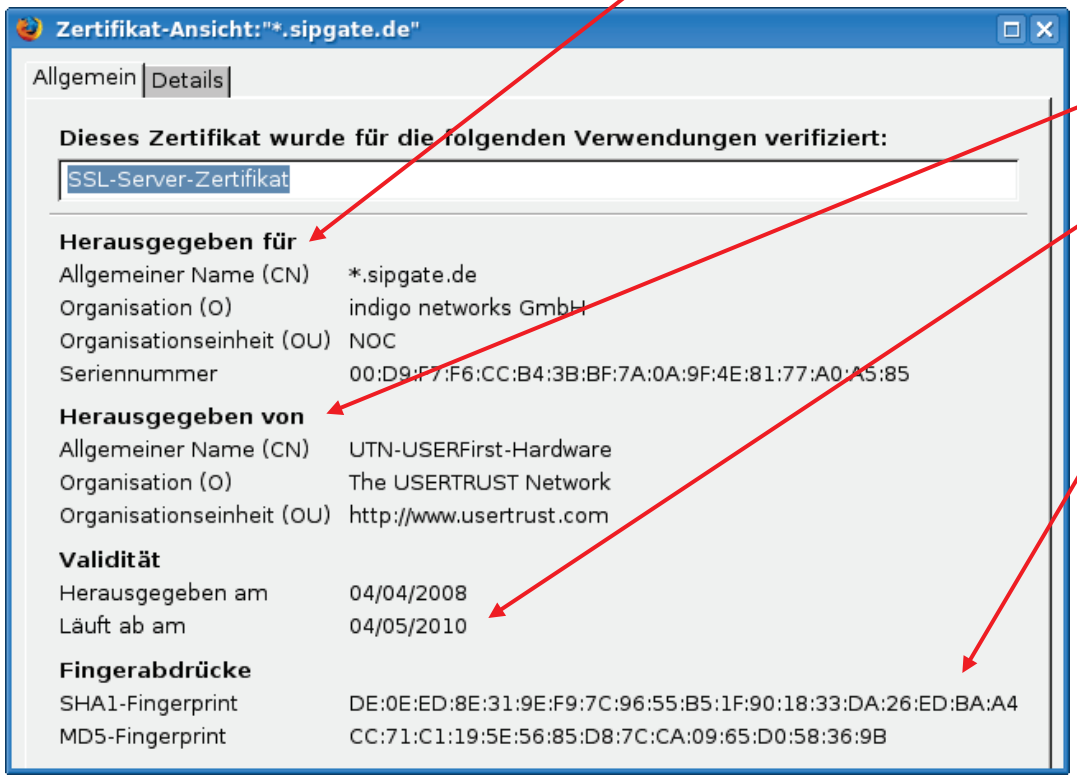
- Bobs Zertifikat besorgen (von Bob oder von irgendwo sonst)
- Öffentlichen Schlüssel der CA anwenden, um die Verbindung zwischen Bobs öffentlichem Schlüssel und seiner Identität zu überprüfen
- Verwenden des so überprüften öffentlichen Schlüssels von Bob

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

8.4 Zertifikat

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



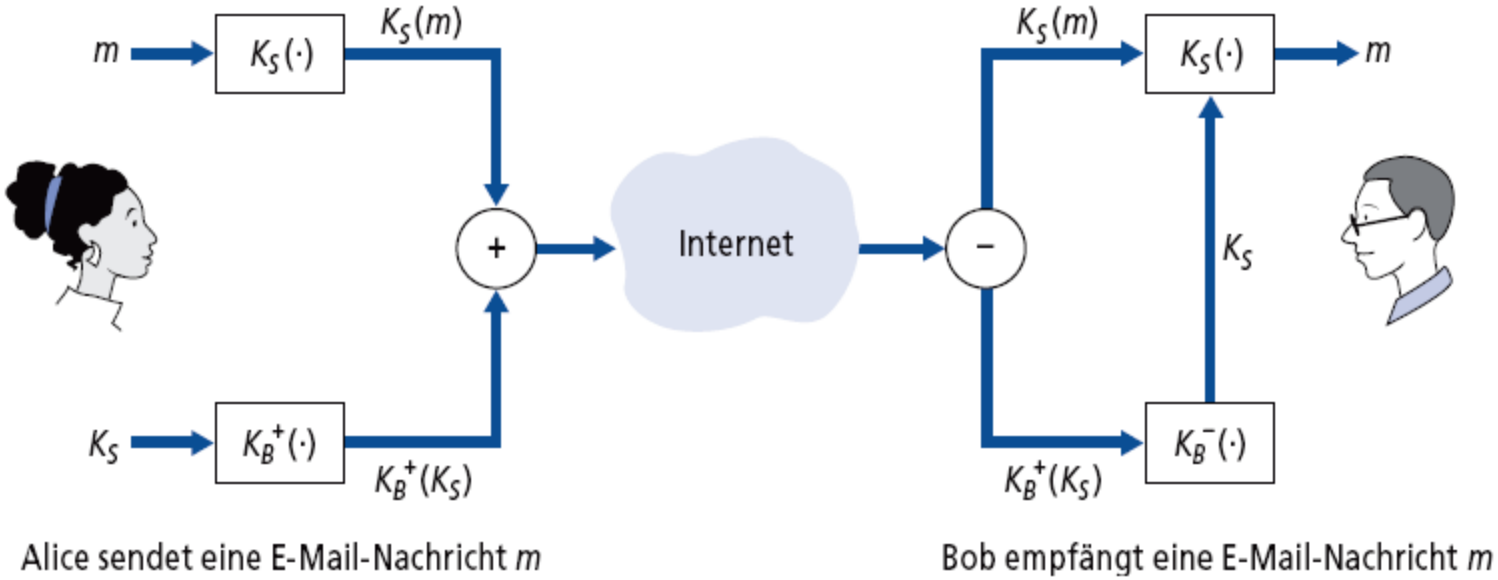
- Info über Aussteller
- Gültigkeitszeitraum
- Digitale Unterschrift des Ausstellers

Kapitel 8 - Netzwerksicherheit

8.1 Was ist Netzwerksicherheit?
8.2 Grundlagen der Kryptographie
8.3 Endpunktauthentifizierung
8.4 Nachrichtenintegrität
8.5 Absichern von E-Mail
8.6 Absichern von TCP-Verbindungen: SSL
8.7 Sichern auf der Netzwerkschicht: Ipsec und VPNs
8.8 Sicherheit von Wireless LAN
8.9 Operative Sicherheit: Firewalls und IDS

8.5 Sichere E-Mail

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



Alice sendet eine E-Mail-Nachricht *m*

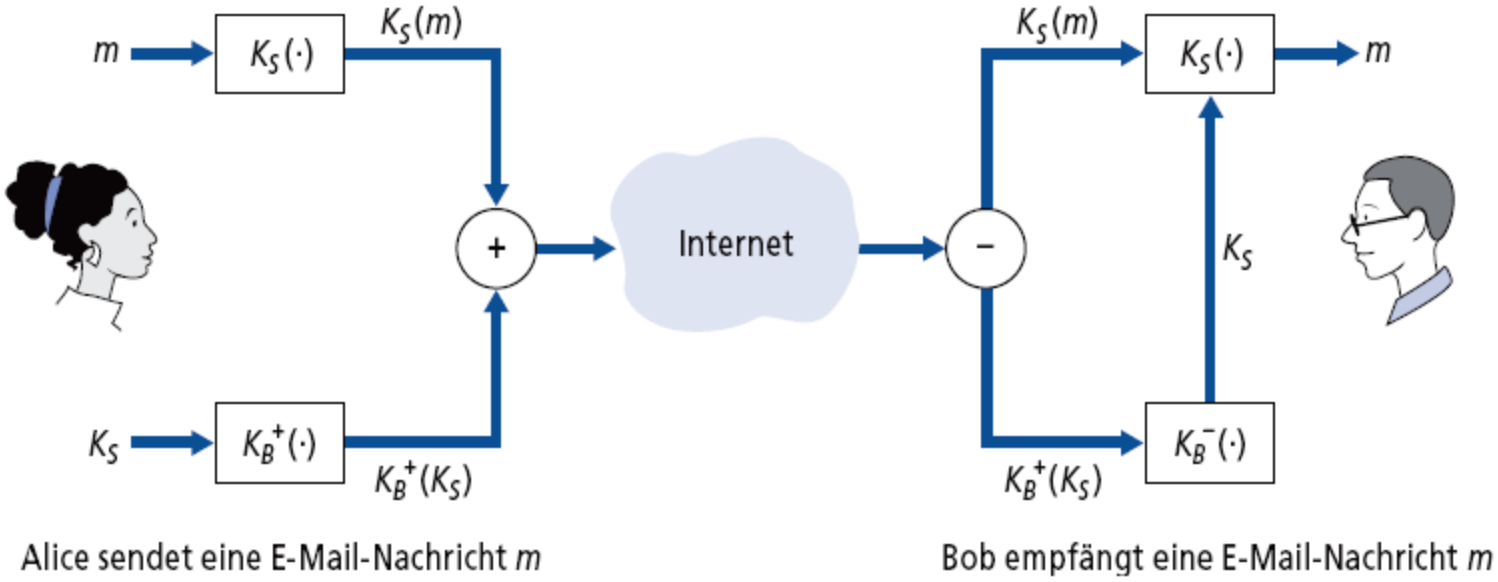
Bob empfängt eine E-Mail-Nachricht *m*

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.

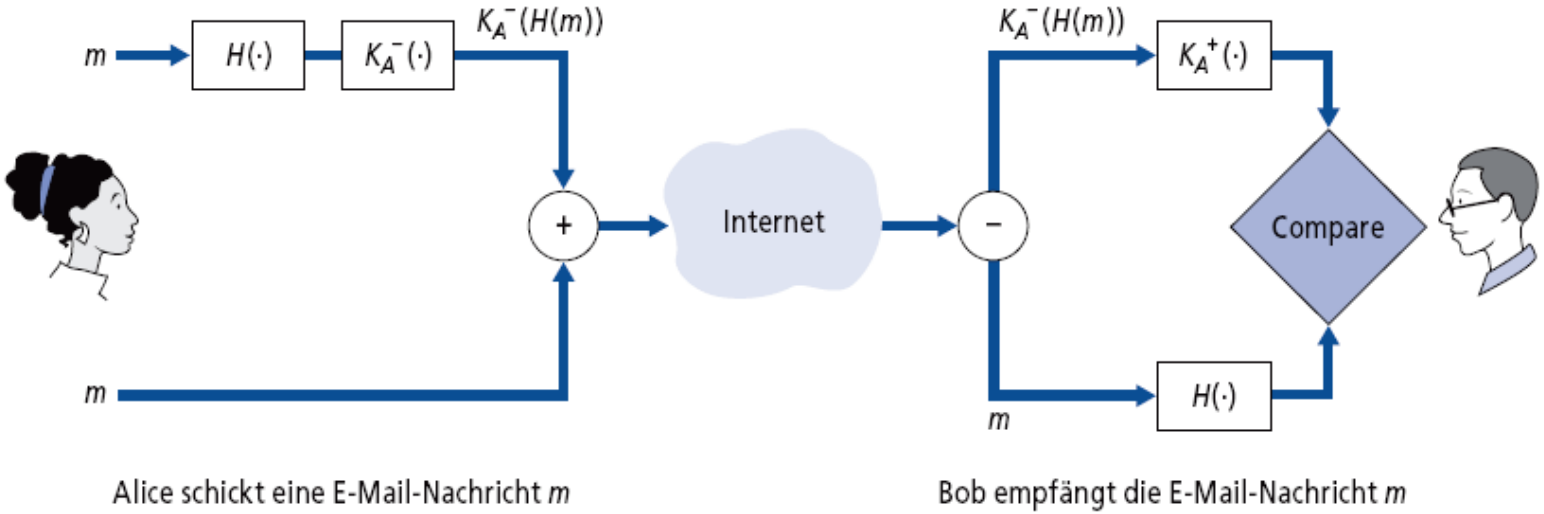


Bob:

- 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

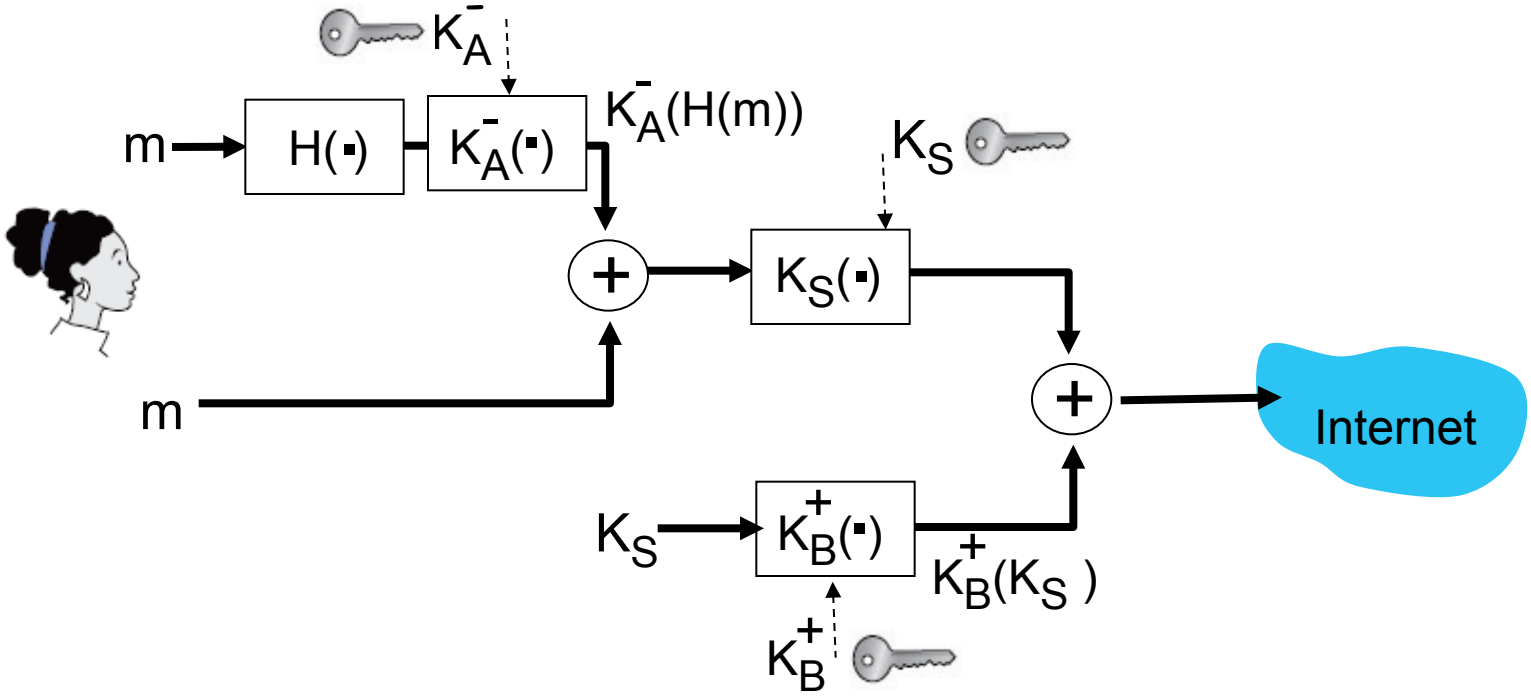
Alice möchte für ihre Nachricht an Bob Absenderauthentifizierung und Nachrichtenintegrität sicherstellen (nicht aber Vertraulichkeit).



- 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

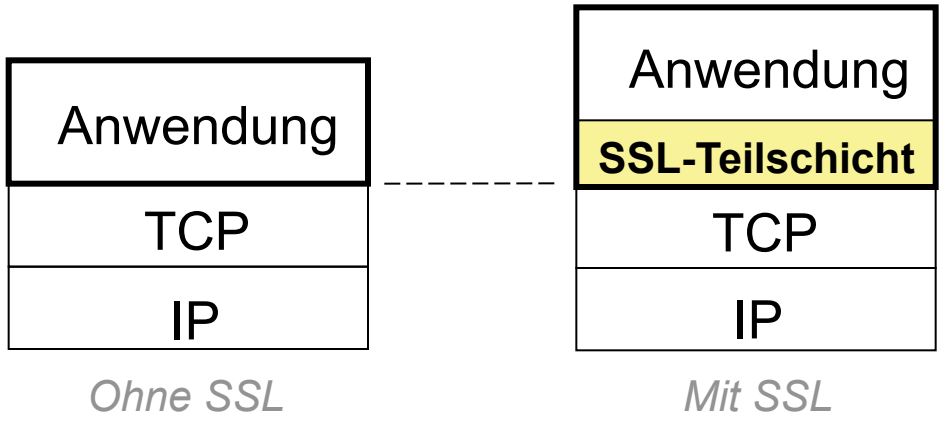
Eine PGP-signierte Nachricht:

```
---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)

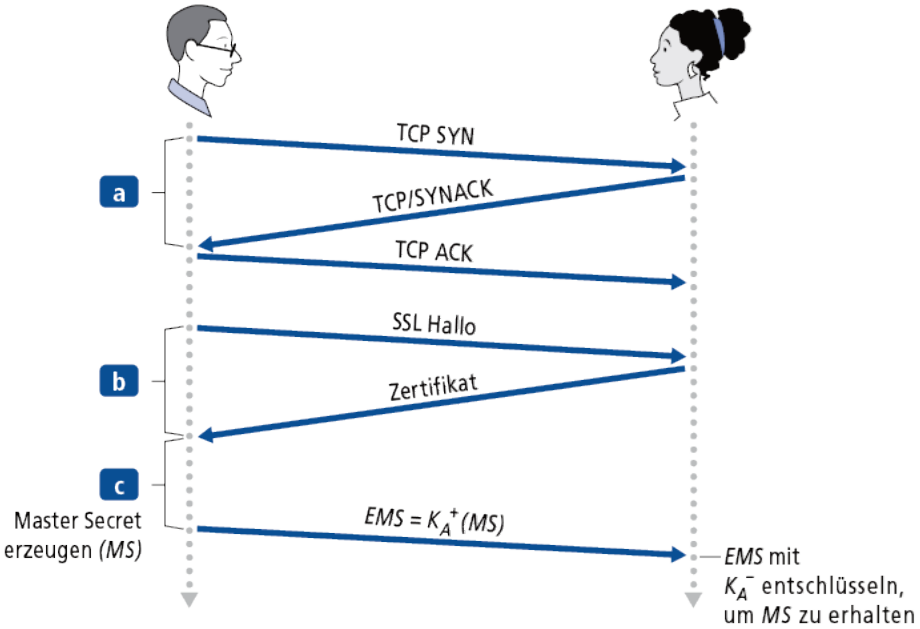
- Transportschichtsicherheit für beliebige TCP-basierte Anwendungen über SSL-Dienste.
  - z.B. zwischen Webbrowser und -server für E-Commerce (**HTTPS**)
- Sicherheitsdienste:
  - Serverauthentifizierung, Datenverschlüsselung, Clientauthentifizierung (optional)
- SSL ist entweder als Teil der Anwendung 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