

# Technische Grundlagen der Informatik 2

## Teil 5: Sicherheit in Layer 4/7

Philipp Rettberg / Sebastian Harnau

# Block 9/18

Sicherheit (Layer 4 / 7)

Kryptographie, Authentifizierung,  
Integrität...

# Netzwerksicherheit

## **Vertraulichkeit:**

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

- Sender verschlüsselt die Nachricht
- Empfänger entschlüsselt die Nachricht

## **Authentifizierung:**

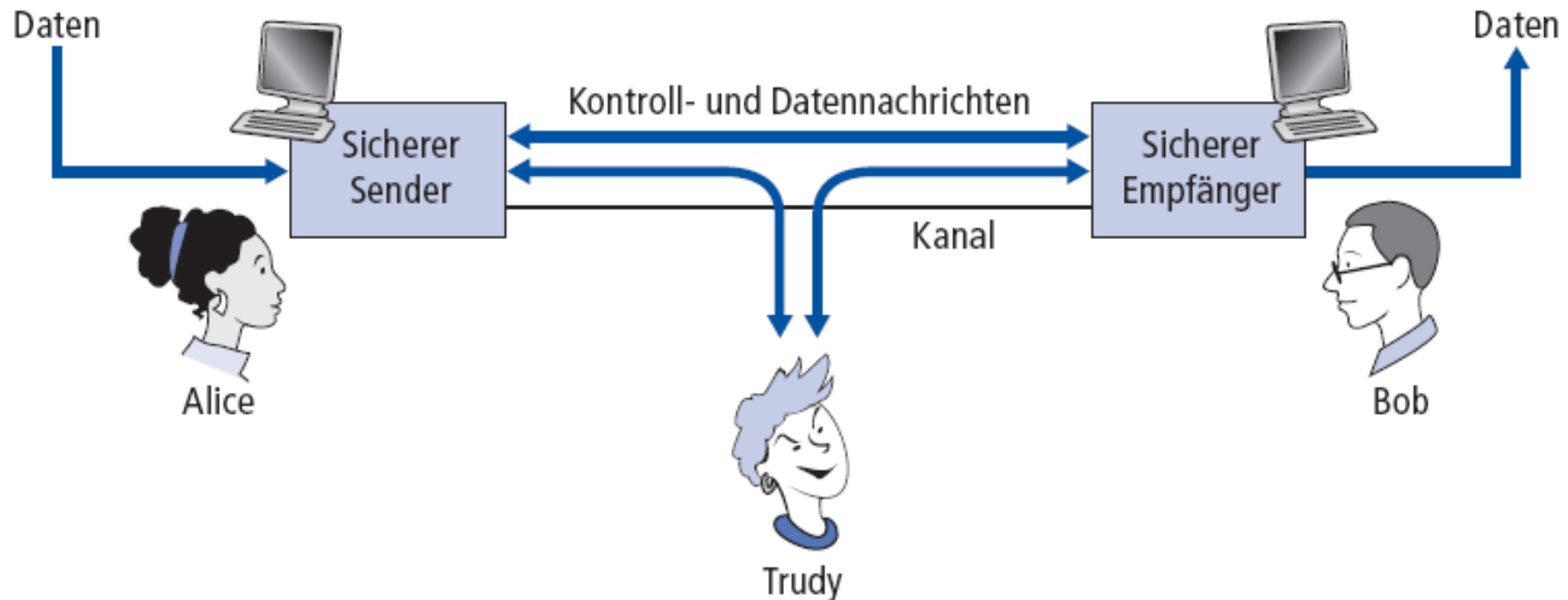
Sender und Empfänger wollen gegenseitig ihre Identität sicherstellen

**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

# Alice, Bob & Trudy

- Alice und Bob wollen "sicher" kommunizieren
- Trudy (ein Eindringling) kann Nachrichten abfangen, löschen, einfügen



# Wer könnten Bob und Alice sein?

- ...wirkliche Personen!
- Webbrowser & -server für elektronische Transaktionen (z.B. Online-Einkäufe)
- Client und Server für Online-Banking
- DNS-Server
- Router, die Routingtabellen-Updates austauschen
- ...

# Mögliche Schadaktionen v. Trudy

- **lauschen**: Nachrichten mitlesen
- aktiv Nachrichten in die Verbindung **einspeisen**
- **fremde Identitäten** annehmen und Quelladressen (oder andere Felder im Paket) fälschen
- bestehende Verbindungen **"kapern"**, durch Entfernen von Sender der Empfänger und Übernahme der entsprechenden Rolle
- **Denial of Service**: verhindern, dass andere einen Dienst nutzen können (z.B. durch Überlasten von Ressourcen)
- ...

# Grundlagen der Kryptographie

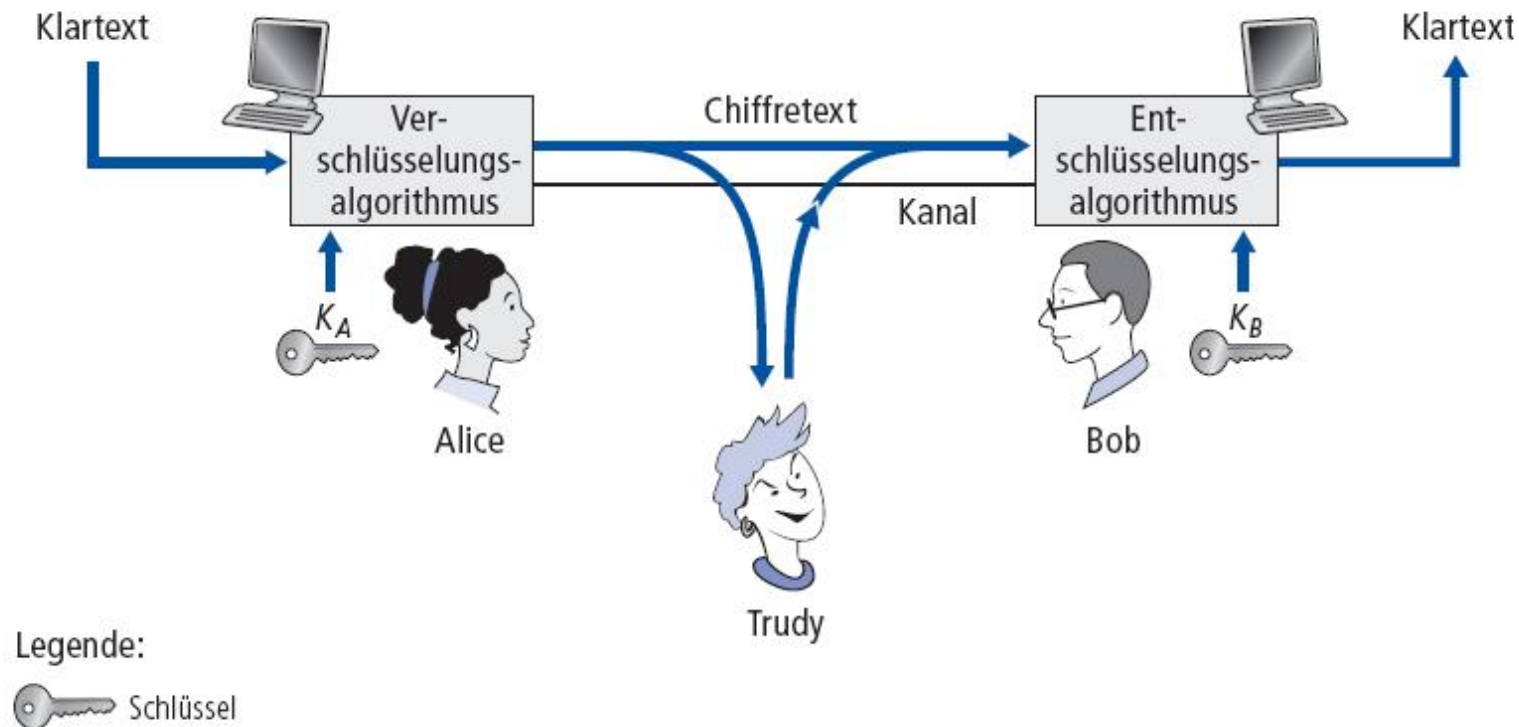
# Terminology

symmetrische Kryptographie:

Sender- und Empfängerschlüssel sind identisch

Public-Key-Kryptographie:

Schlüssel zur Verschlüsselung ist öffentlich bekannt, zur Entschlüsselung geheim






# Kryptographie mit symmetrischen Schlüsseln

**Ersetzungschiffre:** eine Sache durch eine andere ersetzen

- monoalphabetische Chiffre: einen Buchstaben durch einen anderen ersetzen

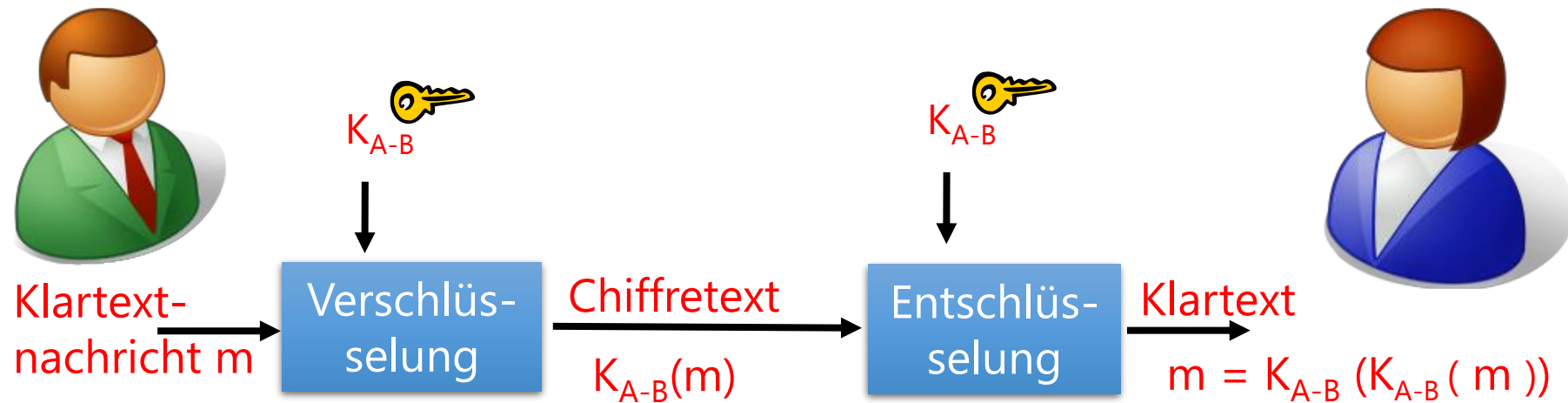
<b>Klartext:</b>	<b>abcdefghijklmnopqrstuvwxyz</b>
	
<b>Chiffretext:</b>	<b>mnbvcxzasdfghjklpoiuytrewq</b>

Frage: Wie knackt man diese Chiffre:

- Brute Force
- andere Wege?

# Kryptographie mit symmetrischen Schlüsseln

- Bob and Alice kennen denselben (symmetrischen) Schlüssel  $K_{A-B}$
- Schlüssel könnte zum Beispiel das Ersetzungsmuster der monoalphabetischen Chiffre sein
- **Frage:** Wie einigen sich Alice und Bob auf einen Schlüssel?



# Symmetrische Kryptographie: DES

## DES: Data Encryption Standard

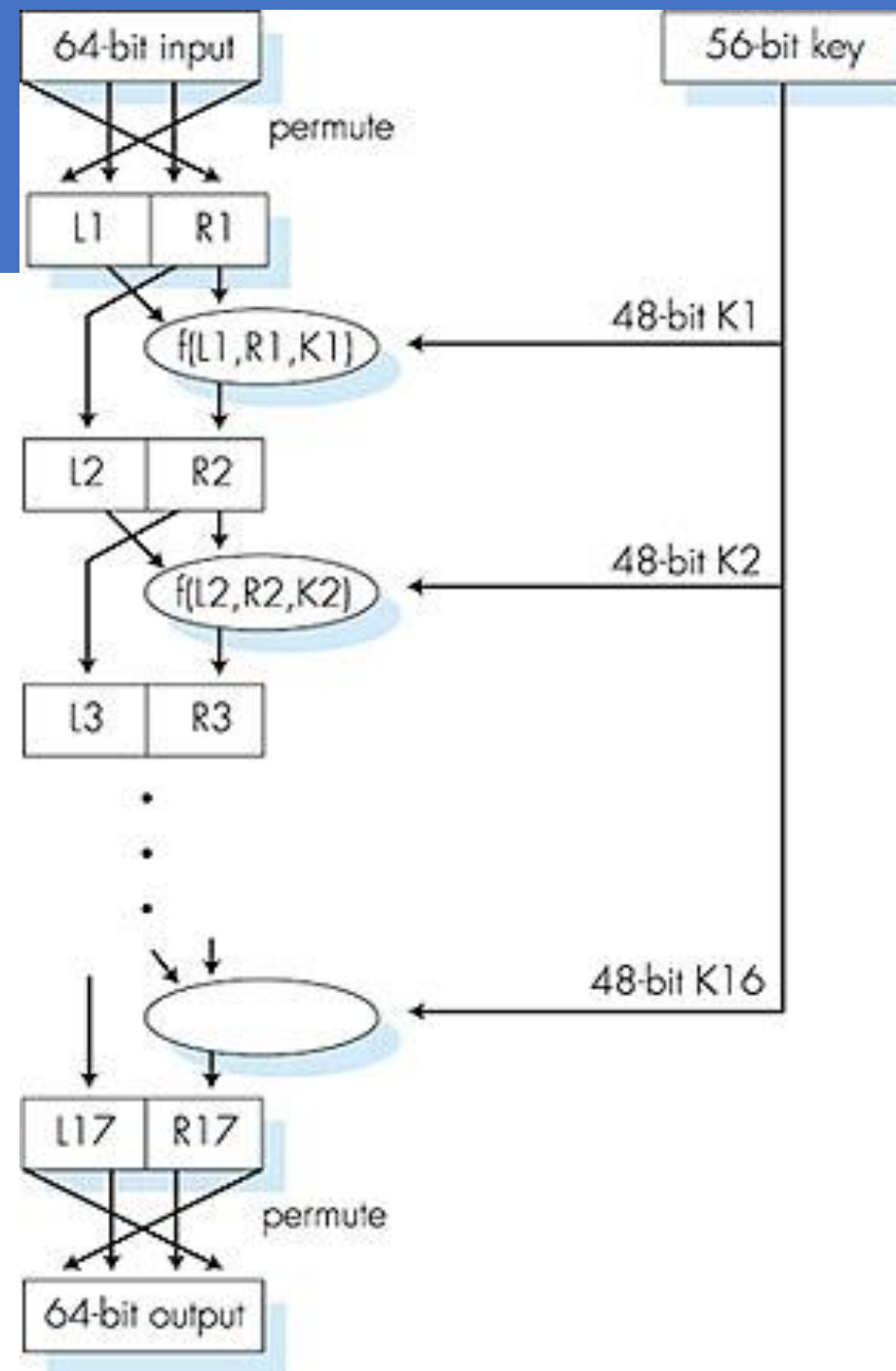
- US-Verschlüsselungsstandard [NIST 1993]
- symmetrische 56-Bit-Schlüssel, 64 Bit lange Klartext-Eingaben
- Wie sicher ist DES?
  - DES Challenge: mit einem 56-Bit-Schlüssel verschlüsselte Phrase ("Strong cryptography makes the world a safer place") wurde (mittels Brute Force) in 4 Monaten entschlüsselt
  - keine bekannte "Hintertür"
- DES sicherer machen:
  - nacheinander mit drei unterschiedlichen Schlüsseln verschlüsseln (3-DES)
  - Cipher Block Chaining (CBC) verwenden

# DES: Arbeitsweise

Anfangspermutation

16 identische "Runden" durch  
Anwenden einer Funktion,  
jedesmal mit unterschiedlichen 48  
Bit des Schlüssels

Endpermutation

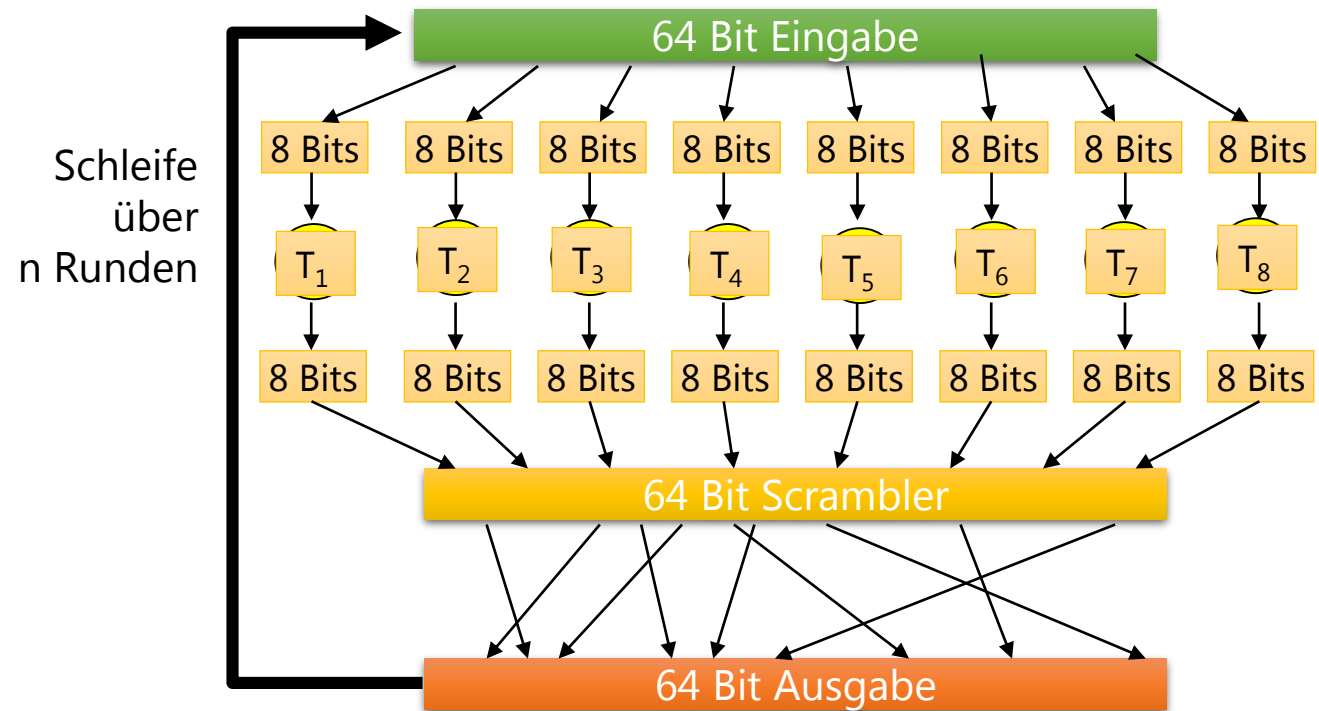


# AES: Advanced Encryption Standard

- 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 149 Billionen Jahre

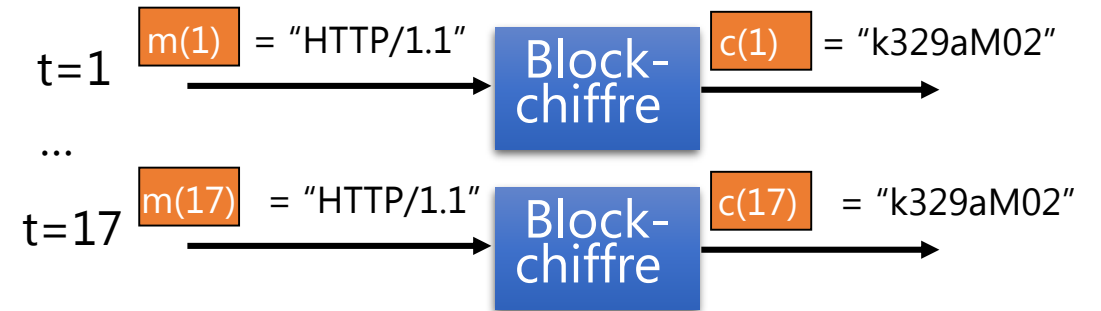
# Blockchiffre

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



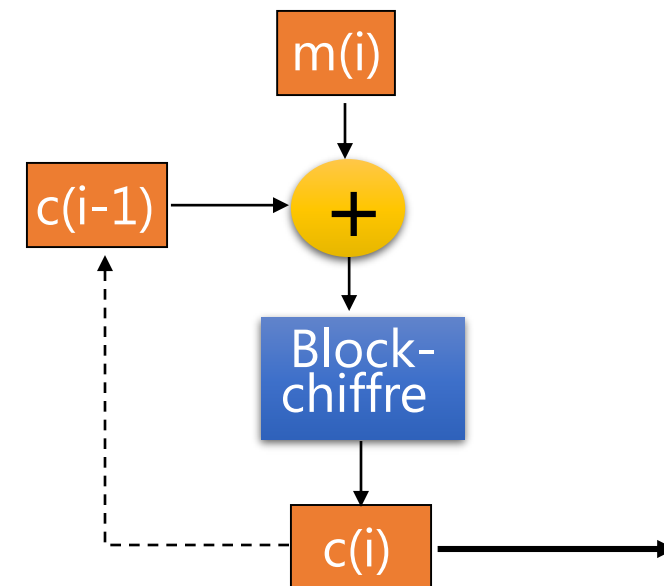
# Cipher Block Chaining

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



## Cipher Block Chaining:

- XOR des  $i$ -ten Eingabeblocks  $m(i)$  mit dem vorangegangenen verschlüsselten Block  $c(i-1)$



# Symmetrisch vs. Public-Key

## Symmetrische Kryptographie

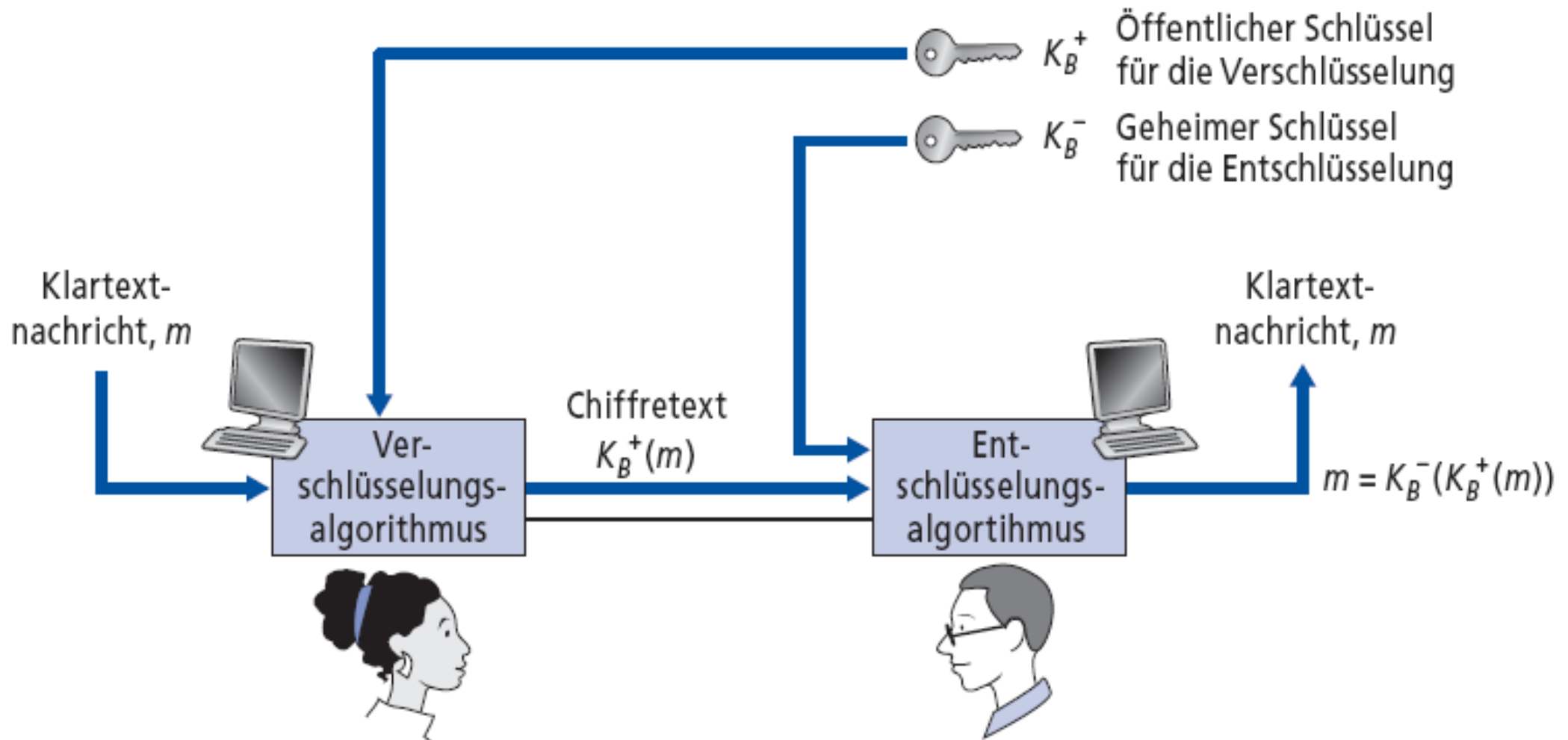
- erfordert, dass Sender und Empfänger über einen gemeinsamen Schlüssel verfügen
- Frage: Wie kann man sich überhaupt auf einen Schlüssel einigen (vor allem dann, wenn man sich noch nie "getroffen" hat)?

## Public-Key-Kryptographie

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



# Public-Key-Kryptographie



# Public-Key-Algorithmen

Anforderungen:

1. Benötigt  $K_B^+()$  und  $K_B^-()$ , für die  $K_B^-(K_B^+(m))=m$  gilt
2. Gegeben den öffentlichen Schlüssel  $K_B^+$ , soll es nicht möglich sein, den privaten Schlüssel  $K_B^-$  zu errechnen

**RSA:** Algorithmus von Rivest, Shamir, Adleman

# RSA: Schlüsselgenerierung

1. Wähle zwei große Primzahlen **p, q**. (jede z.B. 1024 Bit lang)
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: **K<sub>B</sub><sup>+</sup> → (n,e)**. Privater Schlüssel: **K<sub>B</sub><sup>-</sup> → (n,d)**.

# RSA: Ver- und Entschlüsselung

Gegeben  $(n,e)$  und  $(n,d)$ , berechnet wie vorher beschrieben

- Um ein Bitmuster  $m$  zu verschlüsseln, berechne

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

- Zum Entschlüsseln des empfangenen Wertes  $c$  berechne

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

$$\rightarrow m = (m^e \bmod n)^d \bmod n$$

# RSA: Beispiel

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

verschlüsseln:  $\underline{\text{Zeichen}}$      $\underline{m}$      $\underline{m^e}$      $\underline{c = m^e \bmod n}$   
                           1            12        1524832                    17

entschlüsseln:  $\underline{c}$      $\underline{c^d}$      $\underline{m = c^d \bmod n}$      $\underline{\text{Zeichen}}$   
                           17        481968572106750915091411825223071697                    12                    1

# RSA: wichtige Eigenschaft

$$\underbrace{K_B^-(K_B^+(m))}_{\text{erst öffentl. Schlüssel anwenden, dann privaten Schlüssel}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{erst privaten Schlüssel anwenden, dann öffentlichen Schlüssel}}$$

erst öffentl. Schlüssel  
anwenden, dann  
privaten Schlüssel

erst privaten Schlüssel  
anwenden, dann  
öffentlichen Schlüssel

*Identische Ergebnisse!*

# Nachrichtenintegrität

# 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  $H(x) = H(y)$ 
  - äquivalent: gegeben  $m = H(x)$ , ( $x$  unbekannt), ist es nicht möglich,  $x$  zu bestimmen.
  - Anmerkung: Die Internet-Prüfsumme erfüllt diese Bedingung nicht!



# Internet-Prüfsumme

Die Internet-Prüfsumme ist eine schlechte kryptographische Hashfunktion

Die Internet-Prüfsumme weist einige Eigenschaften einer kryptographischen Hashfunktion auf:

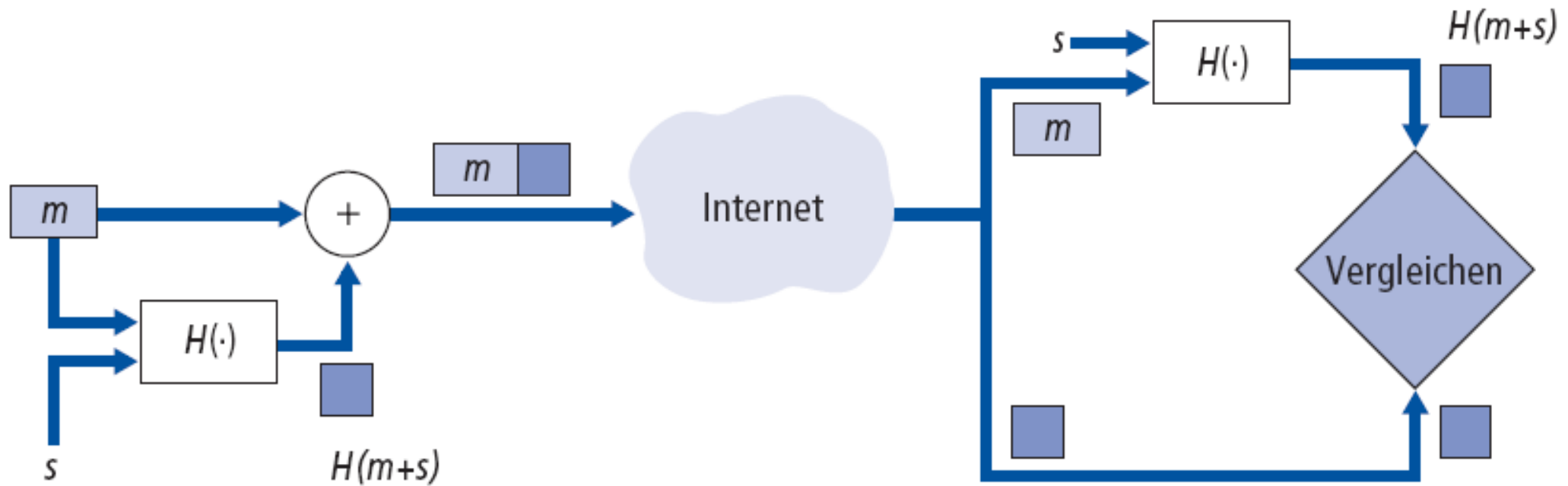
- erstellt eine Prüfsumme fester Länge (16 Bit)
- viele Nachrichten ergeben dieselbe Prüfsumme

Aber zu einer gegebenen Nachricht lässt sich leicht eine andere mit identischem Hashwert finden:

<u>Nachricht</u>	<u>ASCII-Format</u>	<u>Nachricht</u>	<u>ASCII-Format</u>
<b>I O U 1</b>	<b>49 4F 55 31</b>	<b>I O U <u>9</u></b>	<b>49 4F 55 <u>39</u></b>
<b>0 0 . 9</b>	<b>30 30 2E 39</b>	<b>0 0 . <u>1</u></b>	<b>30 30 2E <u>31</u></b>
<b>9 B O B</b>	<b>39 42 4F 42</b>	<b>9 B O B</b>	<b>39 42 4F 42</b>
	<b>B2 C1 D2 AC</b>		<b>B2 C1 D2 AC</b>

unterschiedliche Nachrichten,  
aber identische Prüfsummen!

# Message Authentication Code (MAC)



Legende:

- $m$  = Nachricht
- $s$  = gemeinsames Geheimnis

# MACs in der Praxis

## MD5 (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

1996: erste Kollision  
entdeckt  
Ab 2006: Kollisionen  
reproduzierbar

## SHA-1

- US-Standard [NIST, FIPS PUB 180-1]
- 160-Bit-MAC

## SHA-2 / SHA-3

- **224, 256, 384 oder 512 Bit-MAC**

Ab 2005: Kollision entdeckt;  
Berechnungsaufwand in  
mehreren Anläufen reduziert  
Ab Ende 2015: **deprecated**

**aktuell genutzte  
kryptologische  
Hashfunktionen**

# Digitale Unterschriften

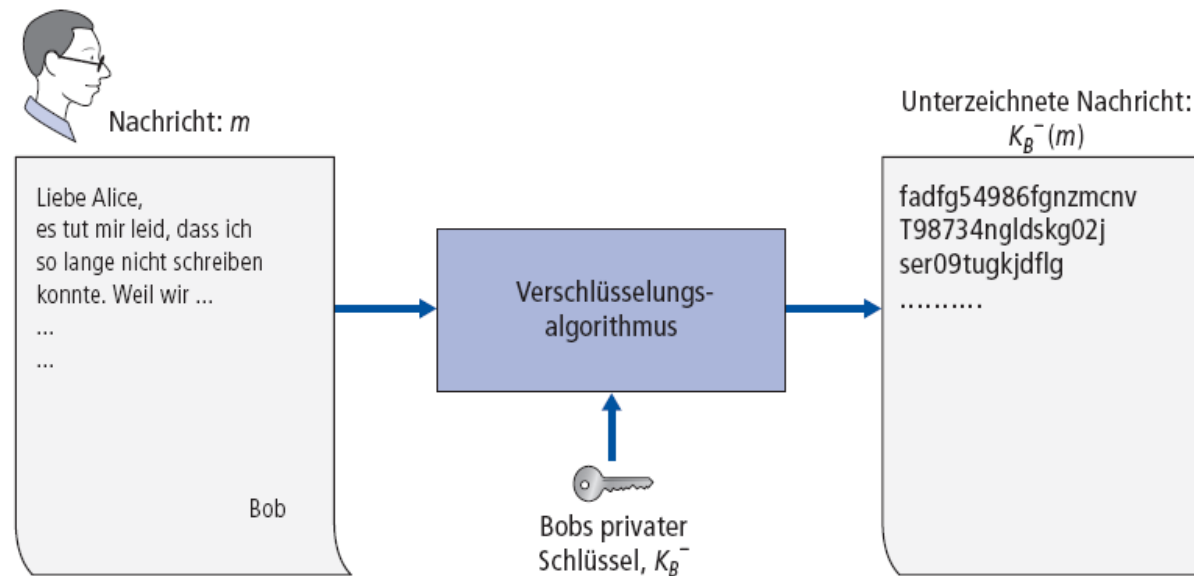
Kryptographische Technik, analog zu eigenhändigen "analogen" Unterschrift

- 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

# 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



# Digitale Unterschriften

angenommen Alice empfängt  $m$  und die digitale Signatur  $K_B^-(m)$

- Alice überprüft Bobs Unterschrift unter  $m$ , indem sie Bobs öffentlichen Schlüssel  $K_B^+$  anwendet (also  $K_B^+(m)$  berechnet) und dann 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:

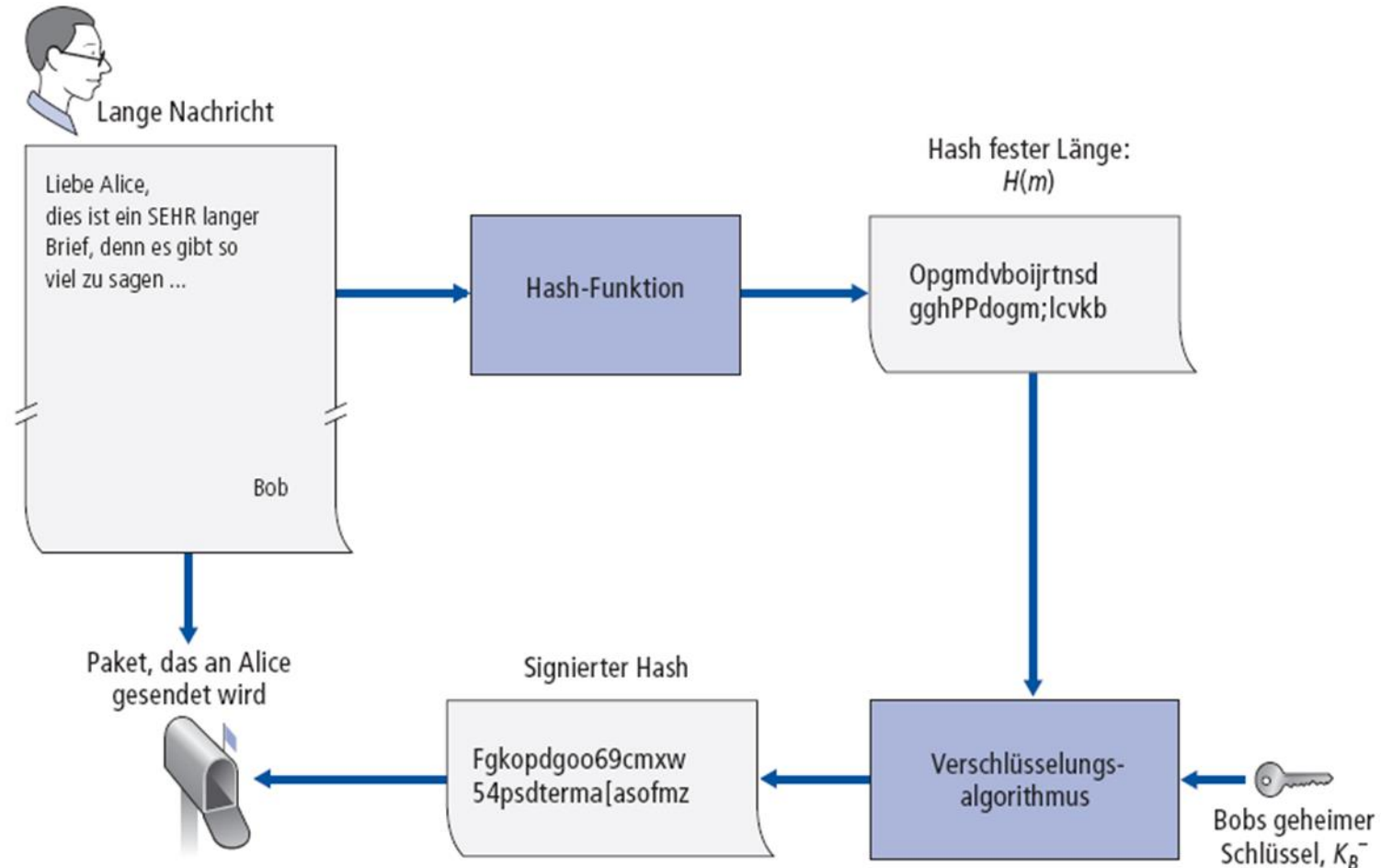
- Bob hat  $m$  unterschrieben.
- Niemand sonst kann die Unterschrift erzeugt haben.
- Bob hat  $m$  und nicht eine andere Nachricht  $m'$  unterschrieben.

Nicht-Abstreitbarkeit:

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

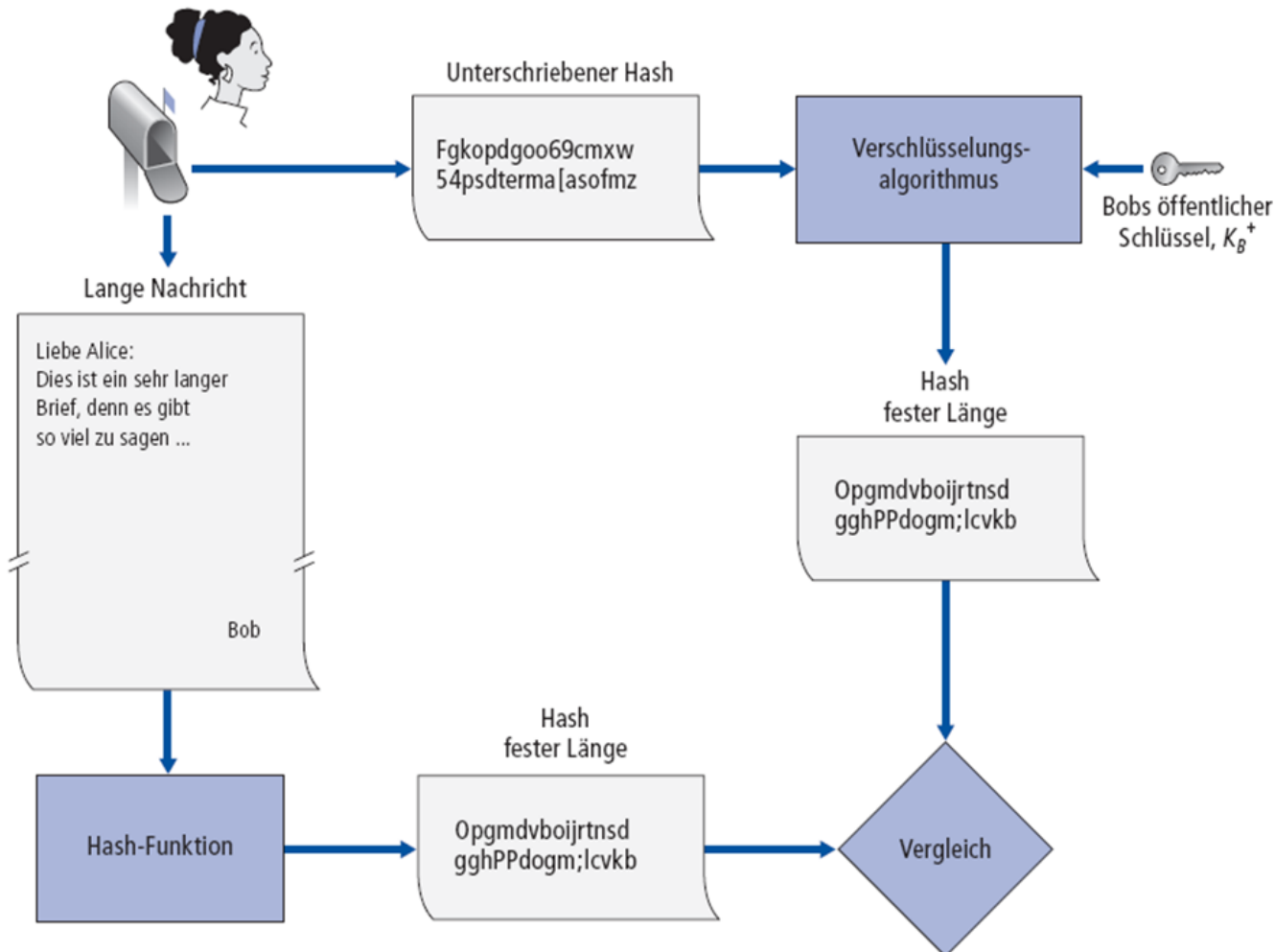
# Digitale Unterschrift = signierter MAC

Bob verschickt eine digital signierte Nachricht:



# Digitale Unterschrift = signierter MAC

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





# Zertifizierung öffentlicher Schlüssel

## Problem bei öffentlichen Schlüsseln:

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

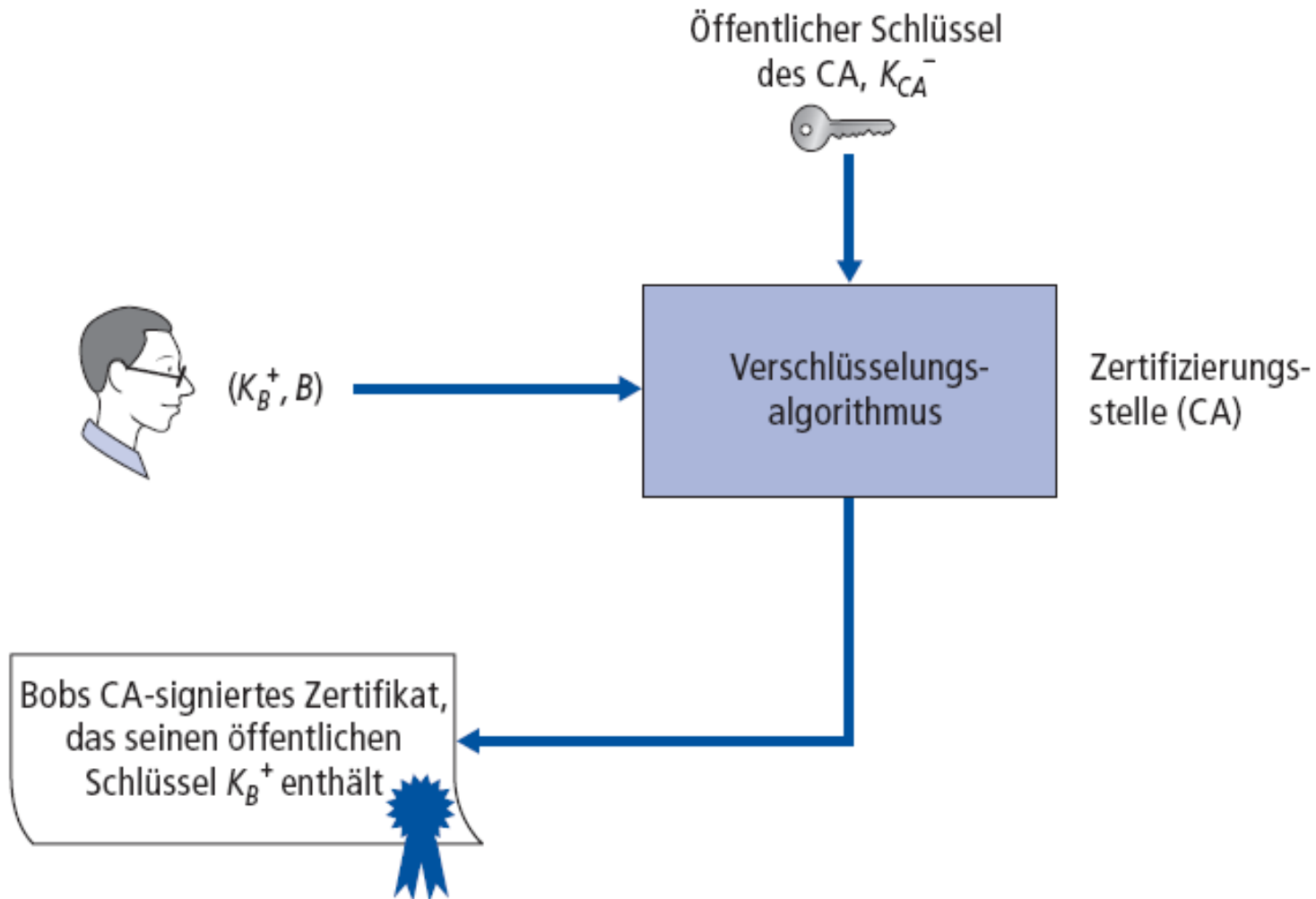
## Lösung:

- vertrauenswürdige Zertifizierungsstelle (Certification Authority, CA)

# 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 eigenen Identität gegenüber der CA.
  - CA erstellt ein Zertifikat, das E und den öffentlichen Schlüssel enthält.
  - Das Zertifikat wird von der CA unterschrieben und besagt:  
"Das ist der öffentliche Schlüssel von E."

# Zertifizierungsstelle

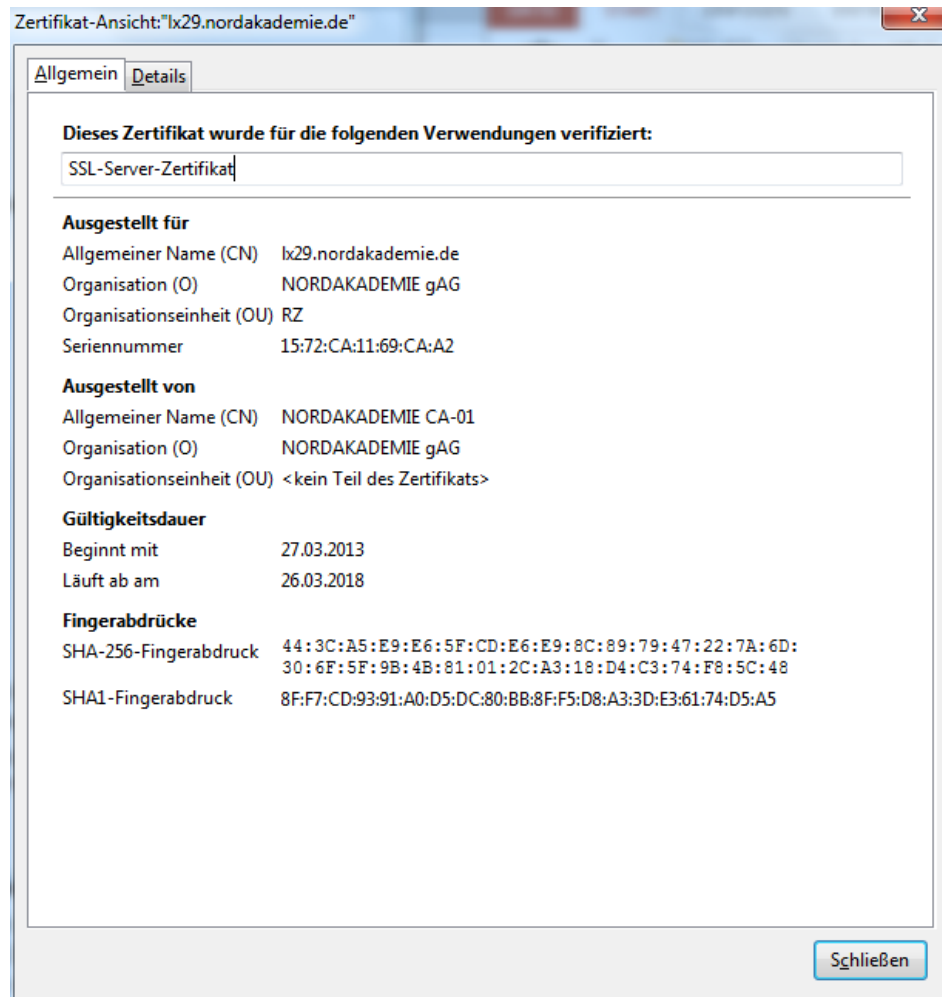


# Zertifizierungsstelle

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

# Inhalt eines Zertifikats



- Information über den Zertifikatsinhaber, einschließlich des Algorithmus und des Schlüssels selbst (hier nicht gezeigt)
- Seriennummer (eindeutig für den Aussteller)
- Info über Aussteller
- Gültigkeitszeitraum
- digitale Unterschrift des Ausstellers

# Endpunktauthentifizierung

# Authentifizierung

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

**Protokoll ap1.0:**

- **Alice sagt "Ich bin Alice"**

Angriffsszenario?

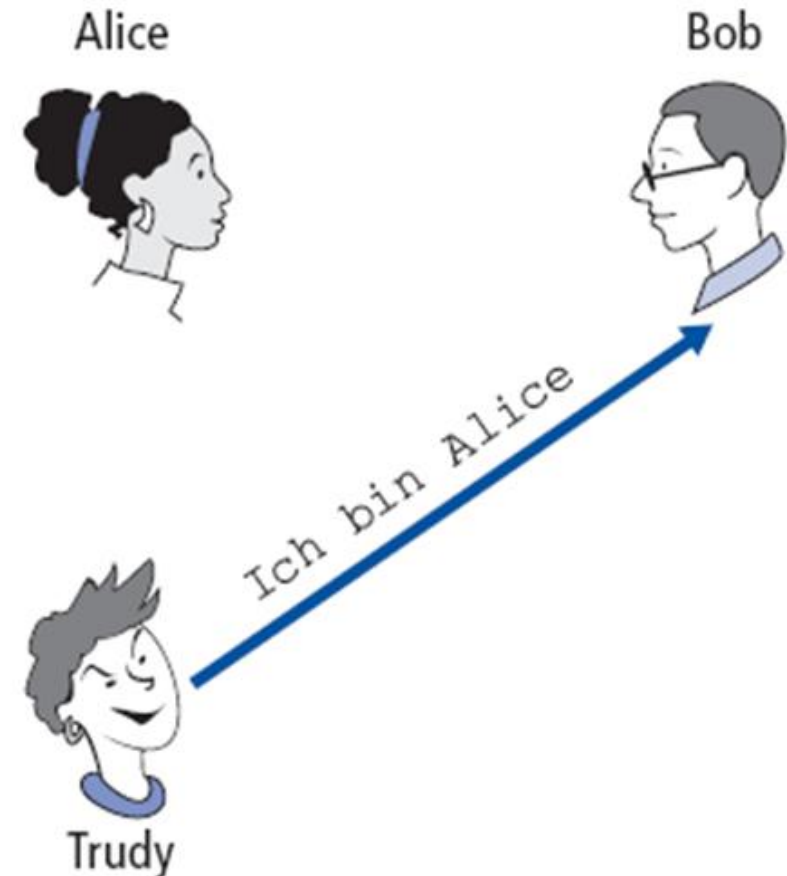


# Authentifizierung

**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





# Authentifizierung

## Protokoll ap2.0:

- **Alice sagt "Ich bin Alice" in einem IP-Paket, das ihre Quell-IP enthält**



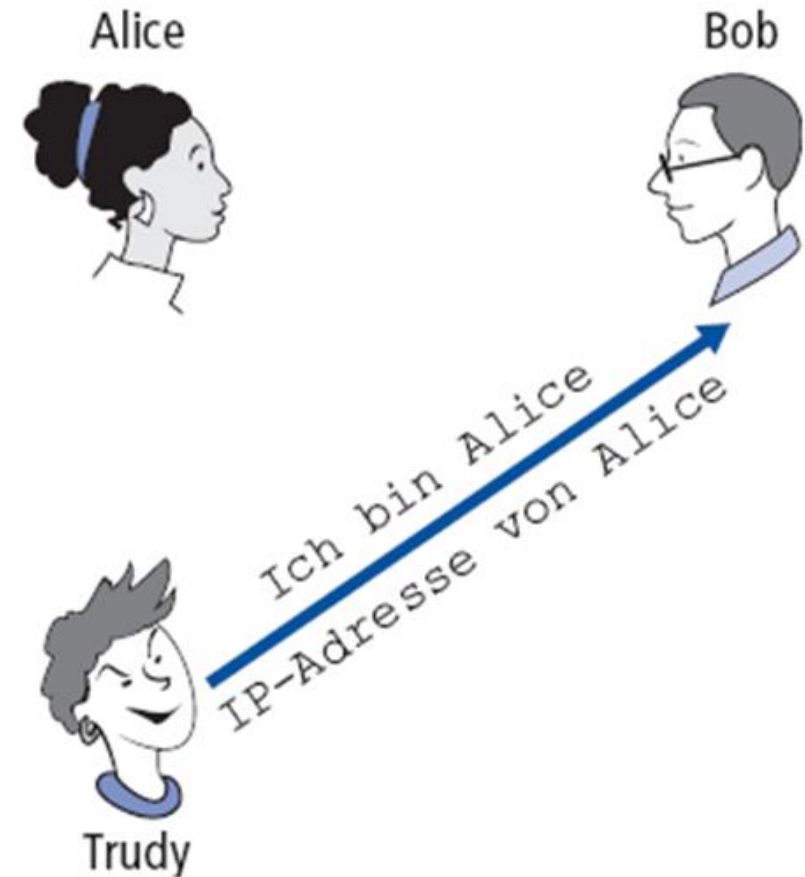
Angriffsszenario?



# 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

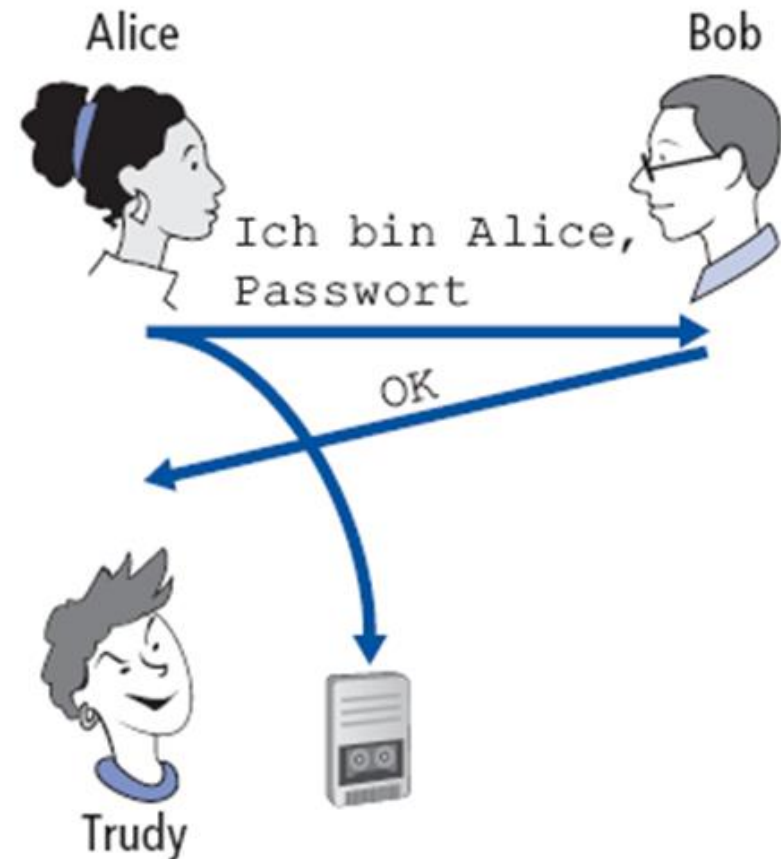


# Authentifizierung

## Protokoll ap3.0:

- **Alice sagt "Ich bin Alice" und schickt ihr geheimes Passwort als "Beweis" mit.**

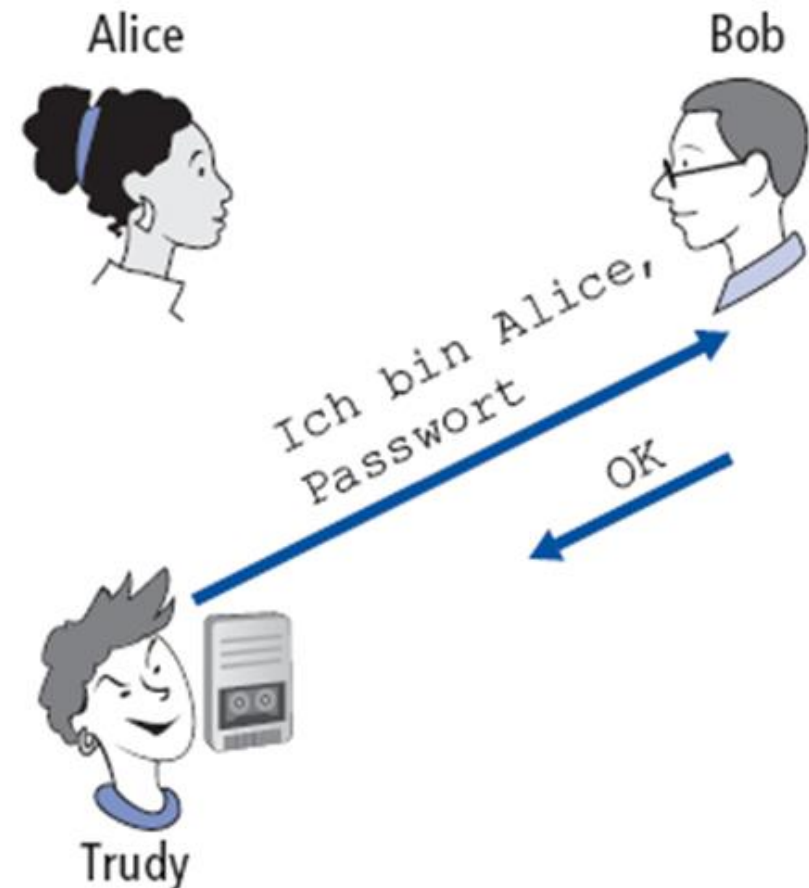
Angriffsszenario?



# 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

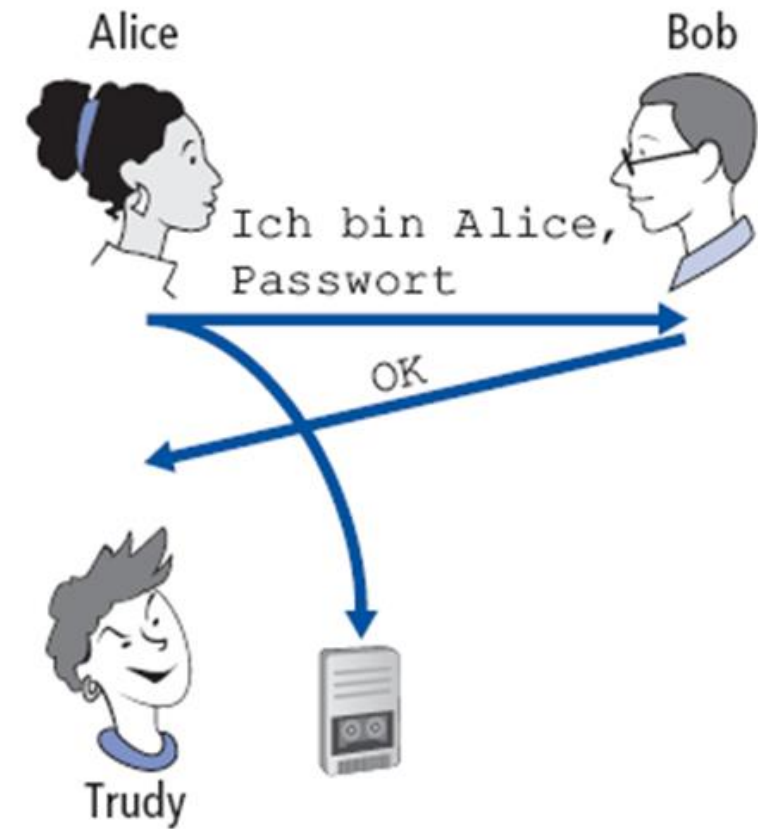


# Authentifizierung

## Protokoll ap3.1:

- **Alice sagt "Ich bin Alice" und schickt ihr verschlüsseltes geheimes Passwort als "Beweis" mit.**

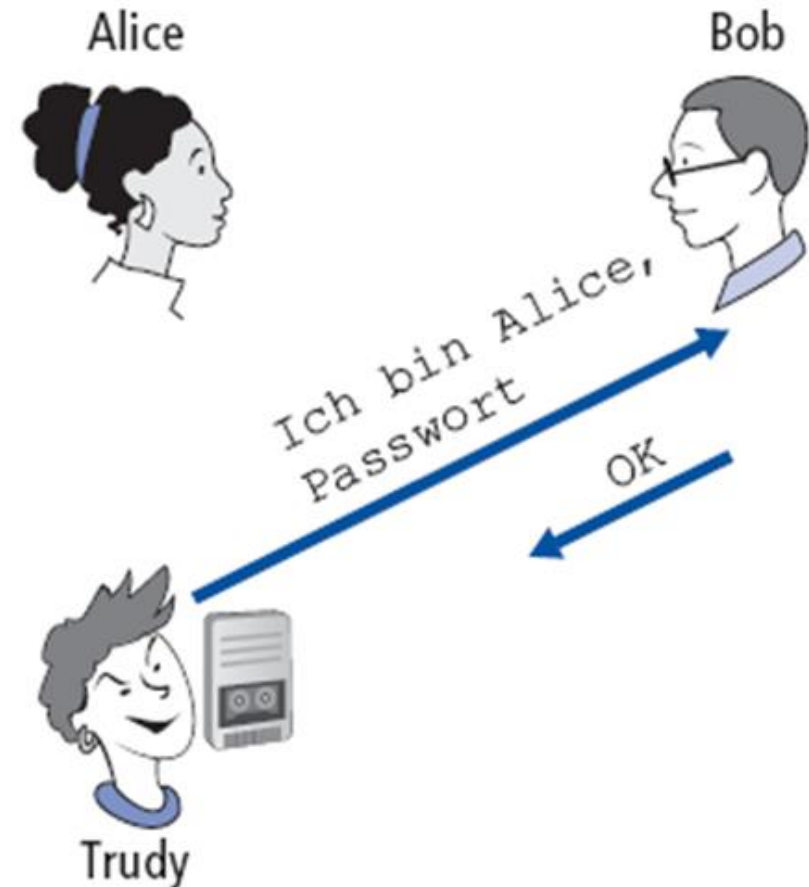
Angriffsszenario?



# Authentifizierung

## Protokoll ap3.1:

- **Alice sagt "Ich bin Alice" und schickt ihr verschlüsseltes geheimes Passwort als "Beweis" mit.**
- aufzeichnen und wiederholen funktioniert immer noch!



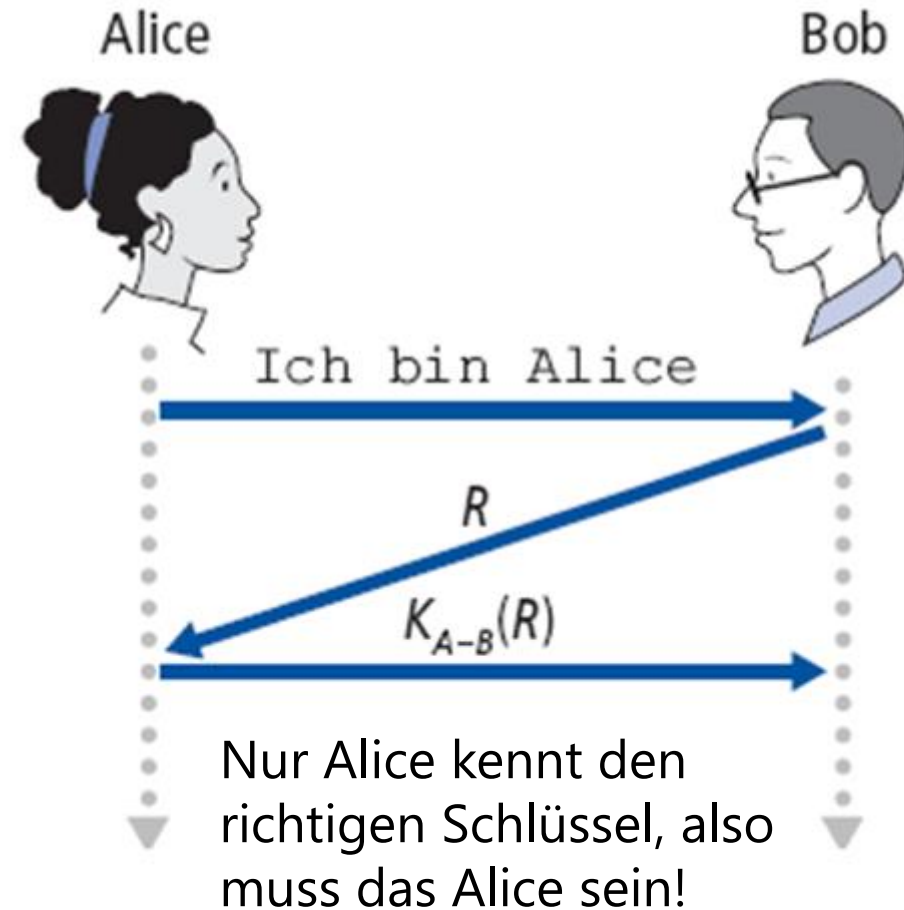
# Authentifizierung

**Ziel:** Playback-Angriff verhindern

**Nonce:** Zahl ( $R$ ), die genau einmal verwendet wird  
("number used **once**")

**ap4.0:**

- um zu beweisen, dass Alice "live" an der Kommunikation teilnimmt, schickt Bob eine Nonce  $R$ , die Alice symmetrisch verschlüsseln und zurückschicken muss
- Fehler / Nachteil?



# Authentifizierung

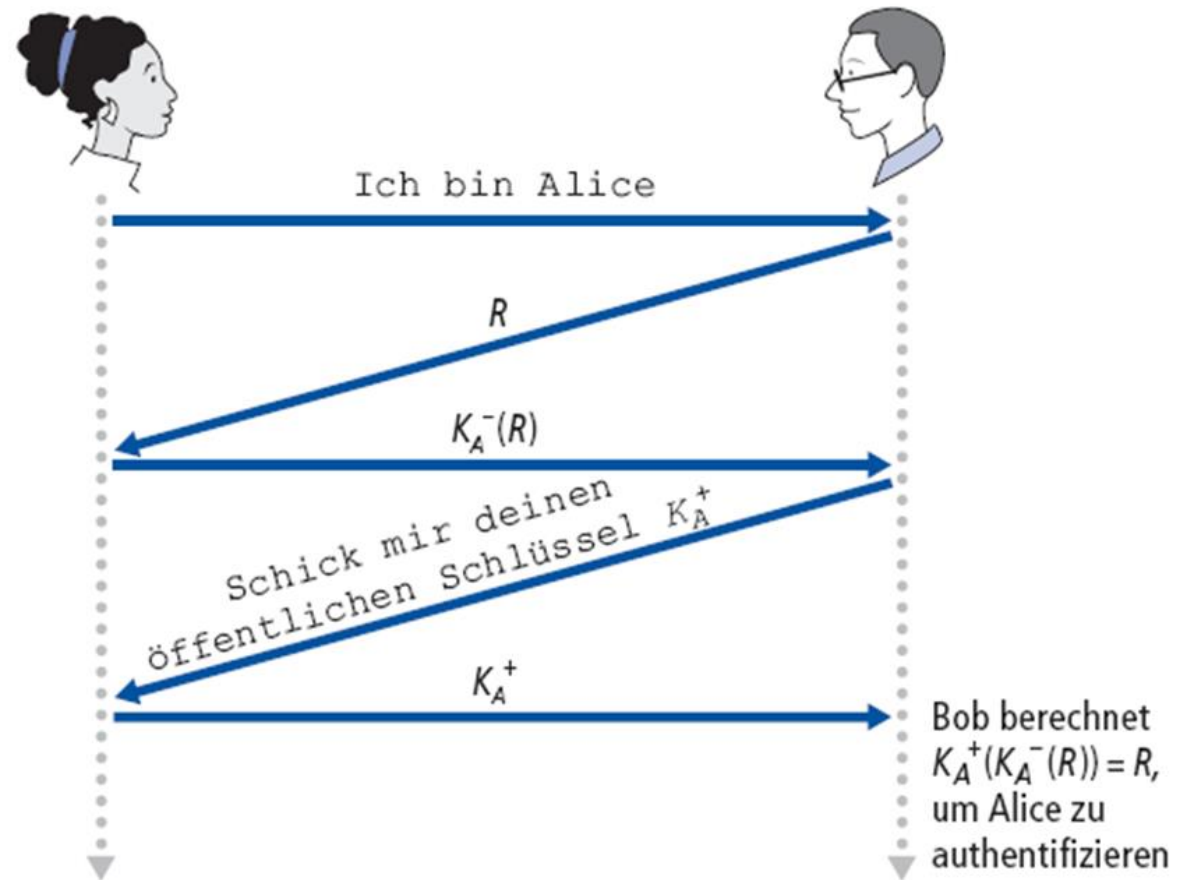
**ap4.0** setzt einen symmetrischen Schlüssel voraus

- können wir Public-Key-Kryptographie verwenden?

**ap5.0:**

- **verwendet eine Nonce und PK-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  $K_A^+(K_A^-(R)) = R$  ist.

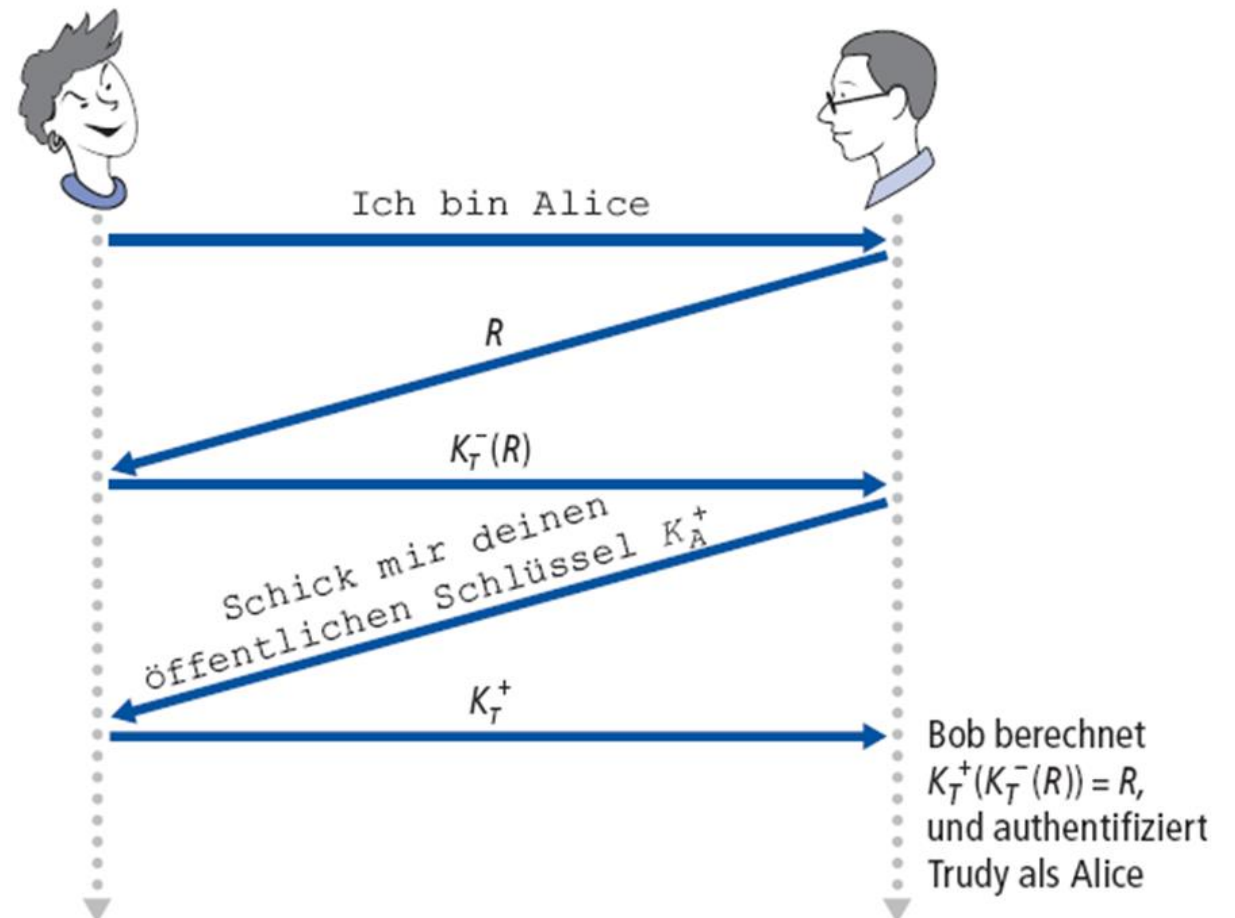




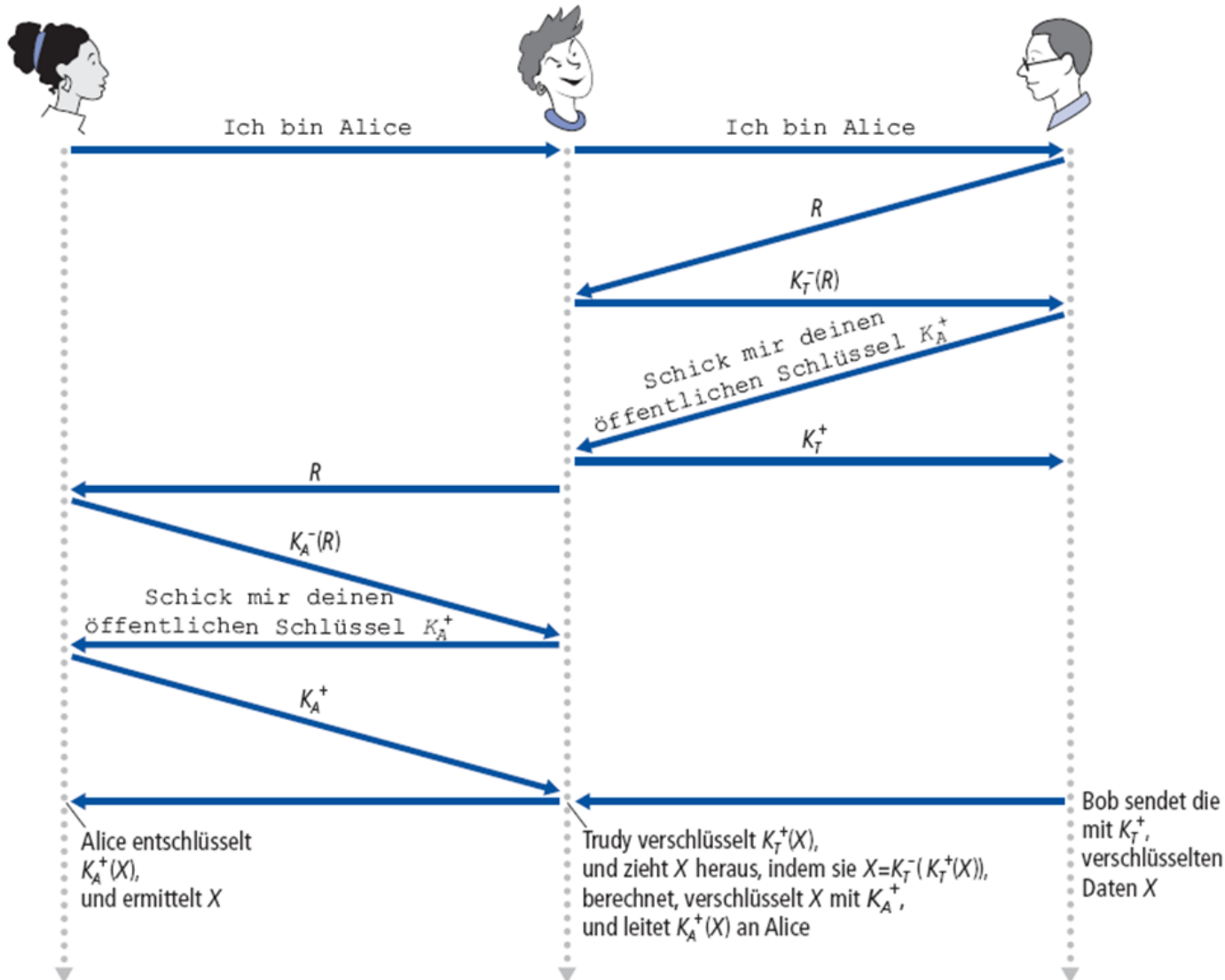
# Authentifizierung

**Angriff, falls keine Zertifikate eingesetzt werden:**

**ap5.0 benötigt Zertifikate!**



# Authentifizierung



## Man-in-the-Middle-Angriff:

- Trudy gibt sich gegenüber Bob als Alice und gegenüber Alice als Bob aus

Schwierig zu entdecken:

- Bob empfängt alles, was Alice sendet und umgekehrt (also könnten sich auch beide, wenn sie sich z.B. eine Woche später treffen, an die Unterhaltung erinnern)
- Das Problem ist, dass auch Trudy alle Nachrichten lesen konnte!

**Lösung: Zertifikate verwenden!**