Karlsruher Institut für Technologie

Vorlesung Einführung in Rechnernetze
7. Netzsicherheit

Prof. Dr. Martina Zitterbart
Sebastian Friebe (M.Sc.), Markus Jung (M.Sc.), Matthias Flittner (M.Sc.), Tim Gerhard (M.Sc.)
[zitterbart | friebe | m.jung | flittner | t.gerhard]@kit.edu

Institut für Telematik, Prof. Zitterbart

Kapitel der Vorlesung

1. Einführung
2. Anwendungsschicht
3. Transportschicht
4. Vermittlungsschicht
5. Sicherungsschicht
6. Netzarchitektur
7. Netzsicherheit

Überblick über Kapitel 3

7.
Netz-
sicherheit

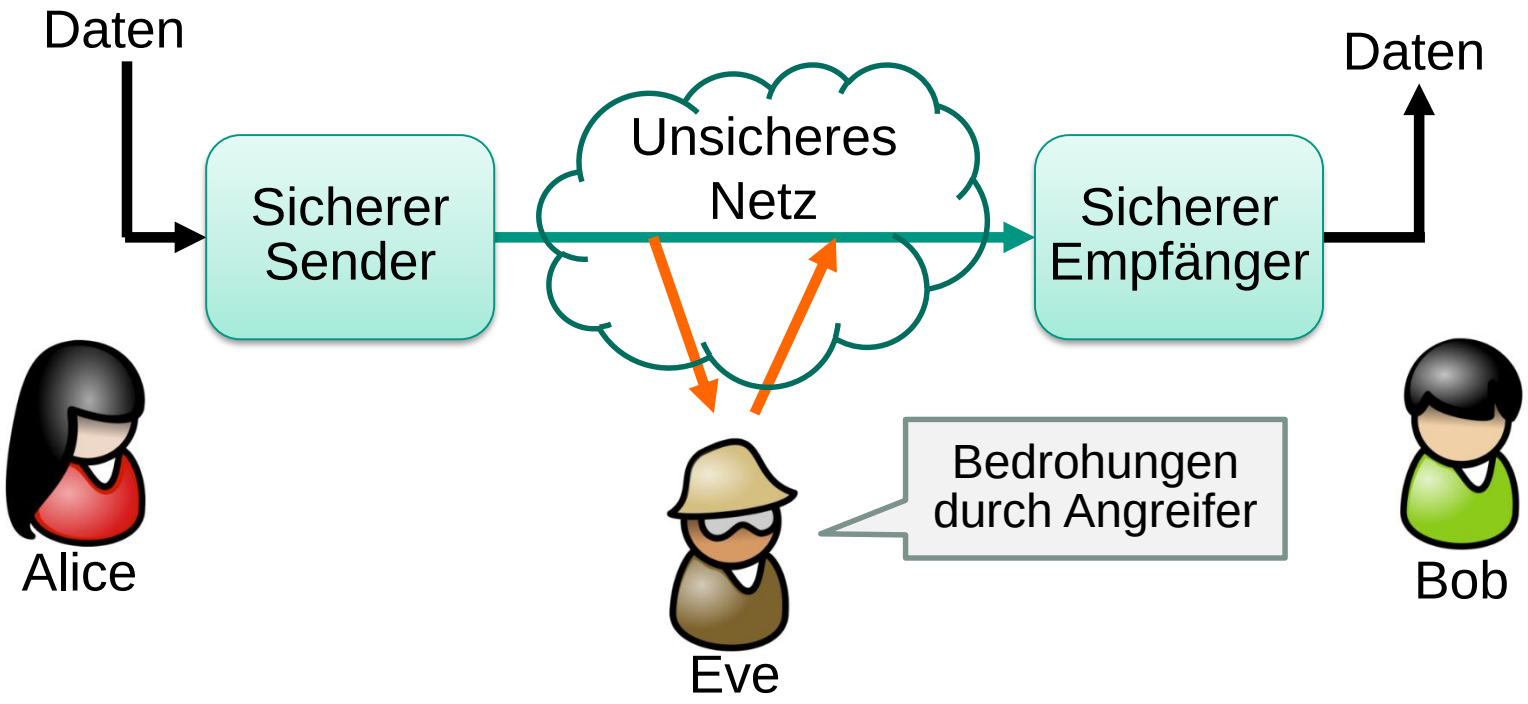
- 7.1 Einführung
- 7.2 Schutzziele
- 7.3 Verschlüsselung
- 7.4 Integritätssicherung
- 7.5 Beispiel: Email-Sicherheit
- 7.6 Infrastruktursicherheit

Kapitel 7.1

EINFÜHRUNG

Einführung

- Alice und Bob wollen „*sicher*“ über unsicheres Netz kommunizieren

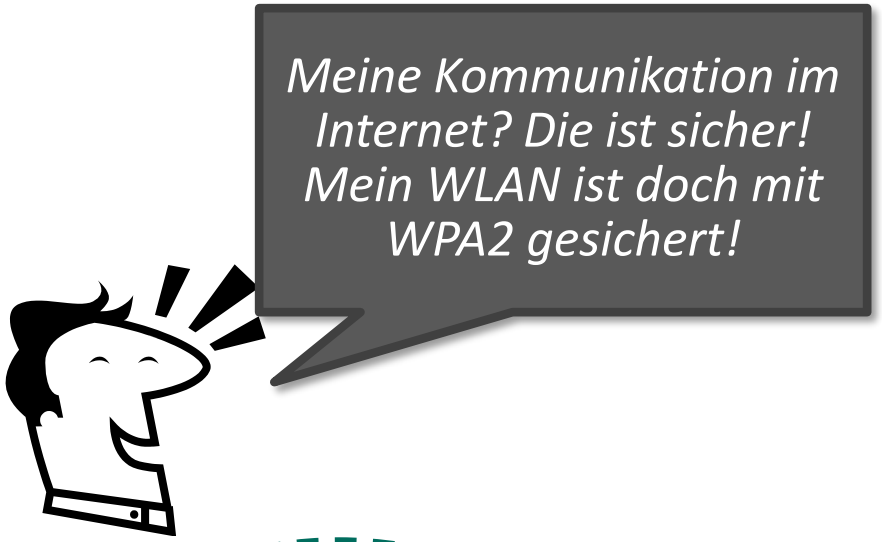


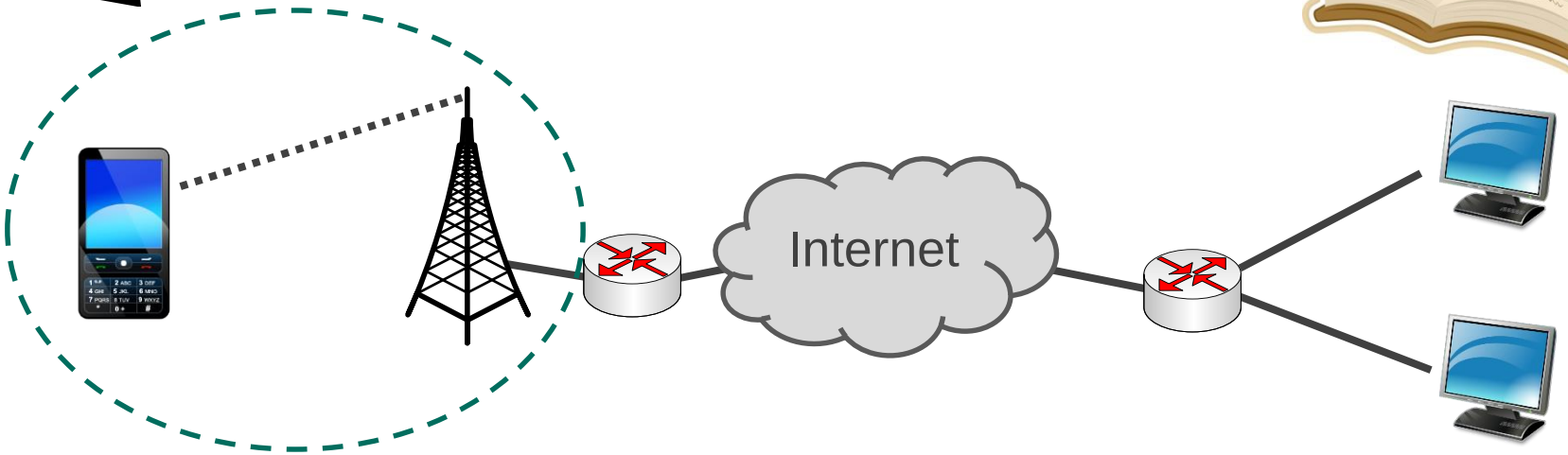
Was heißt „*sicher*“?
Welche Bedrohungen können vorliegen?
Und wie kann Sicherheit realisiert werden?

Kapitel 7.1.1 Potentielle Fallstricke

Wie/was/wo wird geschützt? Fallstricke

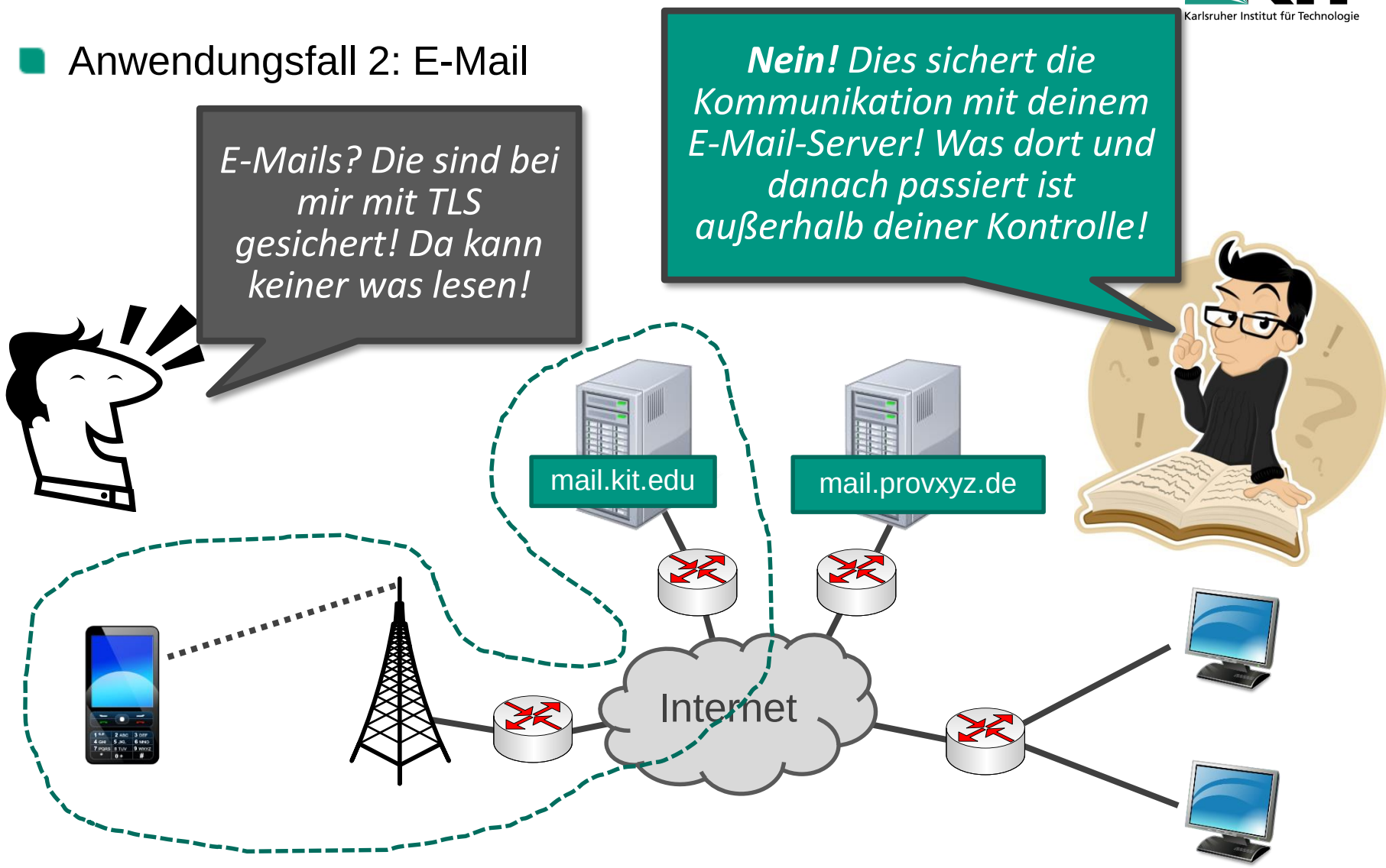
- Anwendungsfall 1: Kabelloser Zugang





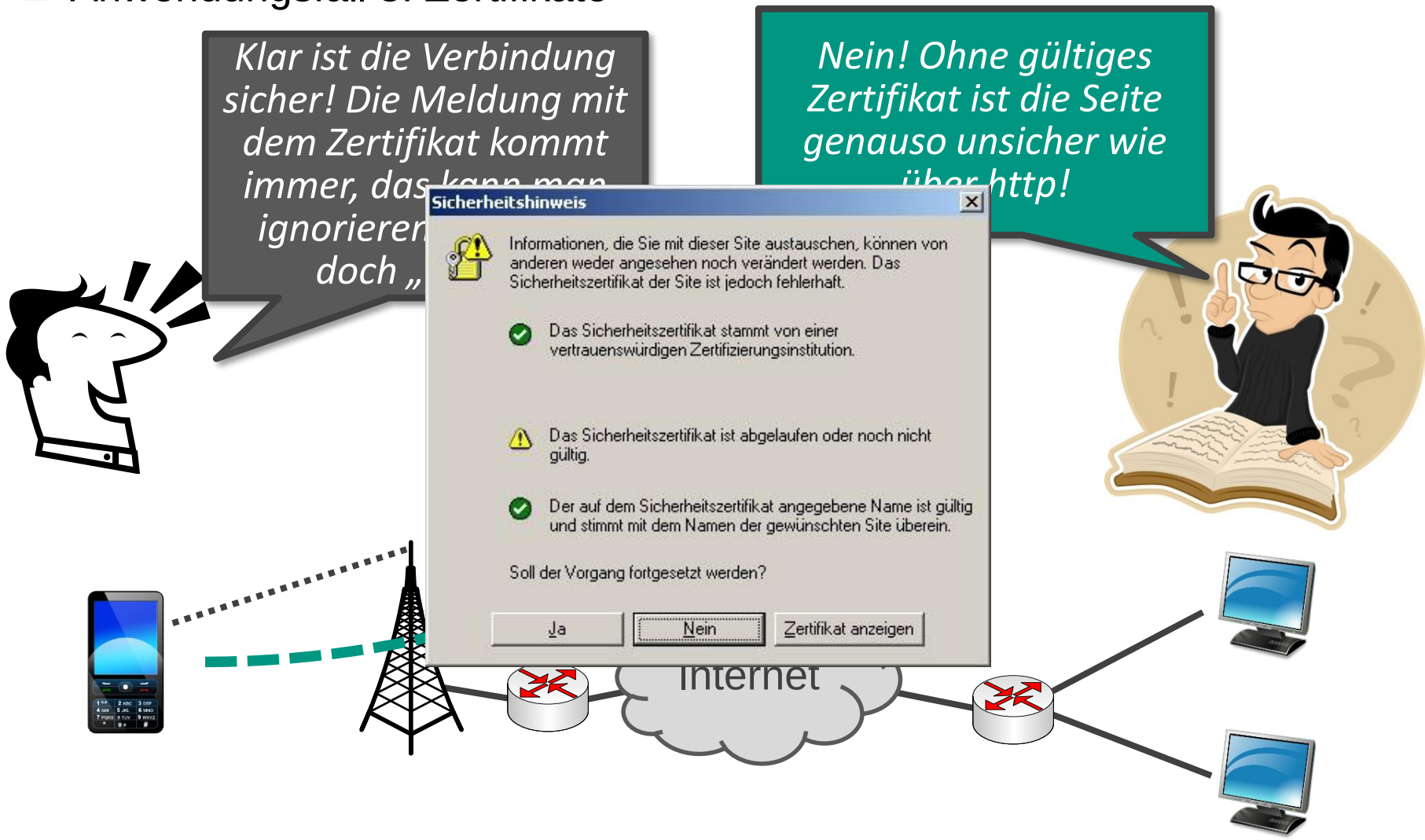
Wie/was/wo wird geschützt? Fallstricke

- Anwendungsfall 2: E-Mail



Wie/was/wo wird geschützt? Fallstricke

- Anwendungsfall 3: Zertifikate



Kapitel 7.1.2 Angreifer

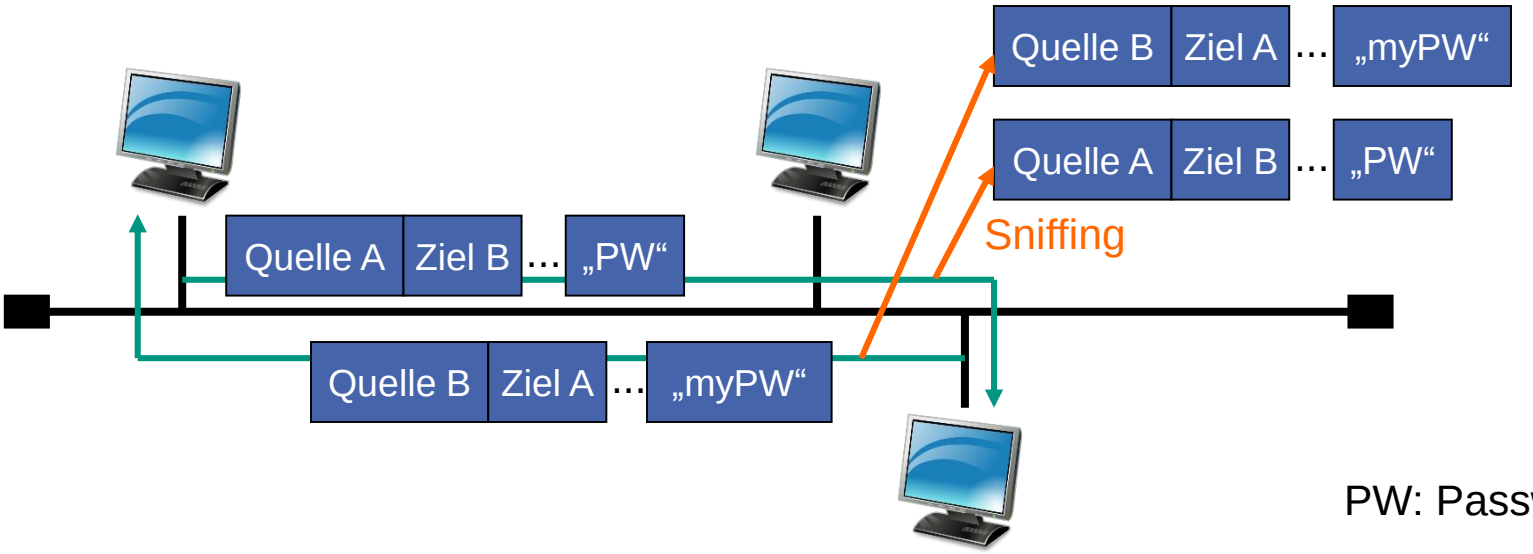
Angreifermodell

- Welche Motivation und Fähigkeiten setzt ein Angreifer ein?
- Bewertung einer Lösung hinsichtlich Sicherheit ohne Annahme eines Angreifermodells (fast) nicht möglich
- Der „Klassiker“: Dolev-Yao-Angreifer
  - Angreifer ist omnipräsent im Netz, kann sämtliche Kommunikation abhören
  - Kann eigene Pakete erzeugen und versenden
  - Kann fremde Pakete modifizieren
  - Kann allerdings nicht Entschlüsseln oder Verschlüsseln, ohne den Schlüssel zu kennen
  - Kein Zugriff auf Endsysteme

[DoYa83]

Beispiele möglicher Angriffe

- Passive Angriffe
  - Abhören und Speichern übertragener Pakete



PW: Passwort

- Aktive Angriffe
  - Wiedereinspielen: Pakete duplizieren und neu senden
  - Einfügen und Löschen von Paketen
  - Modifizieren von Paketen

Kapitel 7.2

SCHUTZZIELE

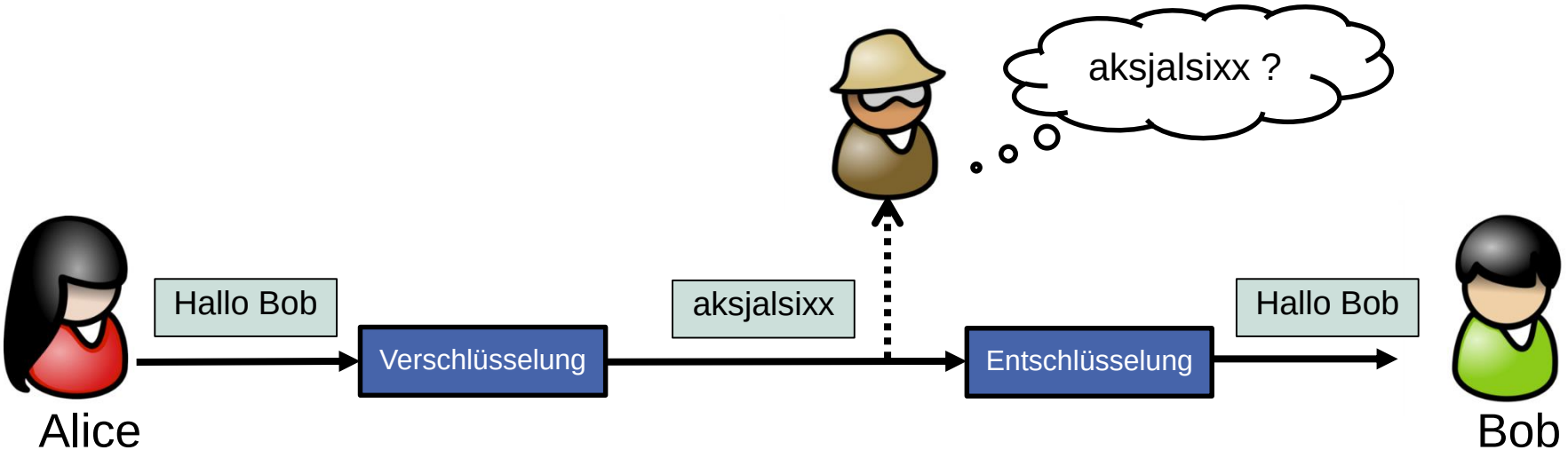
Schutzziele

- Schutzziel
  - Anforderungen an eine Komponente oder ein System, die erfüllt werden müssen, um schützenswerte Güter vor Bedrohungen zu schützen
- Häufige Kategorisierung in
  - **C**onfidentiality (Vertraulichkeit)
  - **I**ntegrity (Integrität)
  - **A**vailability (Verfügbarkeit)

- Weitere Schutzziele
  - Authentizität
  - Privatsphäre
  - (Nicht-)Abstreitbarkeit
- Im folgenden betrachtet
  - Vertraulichkeit, Integrität und Authentizität

Schutzziel Vertraulichkeit

- Vertraulichkeit
  - Ein System bewahrt Vertraulichkeit, wenn es keine unautorisierte Informationsgewinnung ermöglicht
- Baustein zur Umsetzung von Vertraulichkeit
  - Symmetrische oder asymmetrische Verschlüsselung
- Beispiel
  - Alice und Bob kommunizieren vertraulich



Schutzziel Integrität

- Integrität
  - Ein System bewahrt *starke* Integrität, wenn es nicht möglich ist, Daten unautorisiert zu manipulieren
  - Ein System bewahrt *schwache* Integrität, wenn unautorisierte Manipulationen an Daten nicht unbemerkt möglich sind

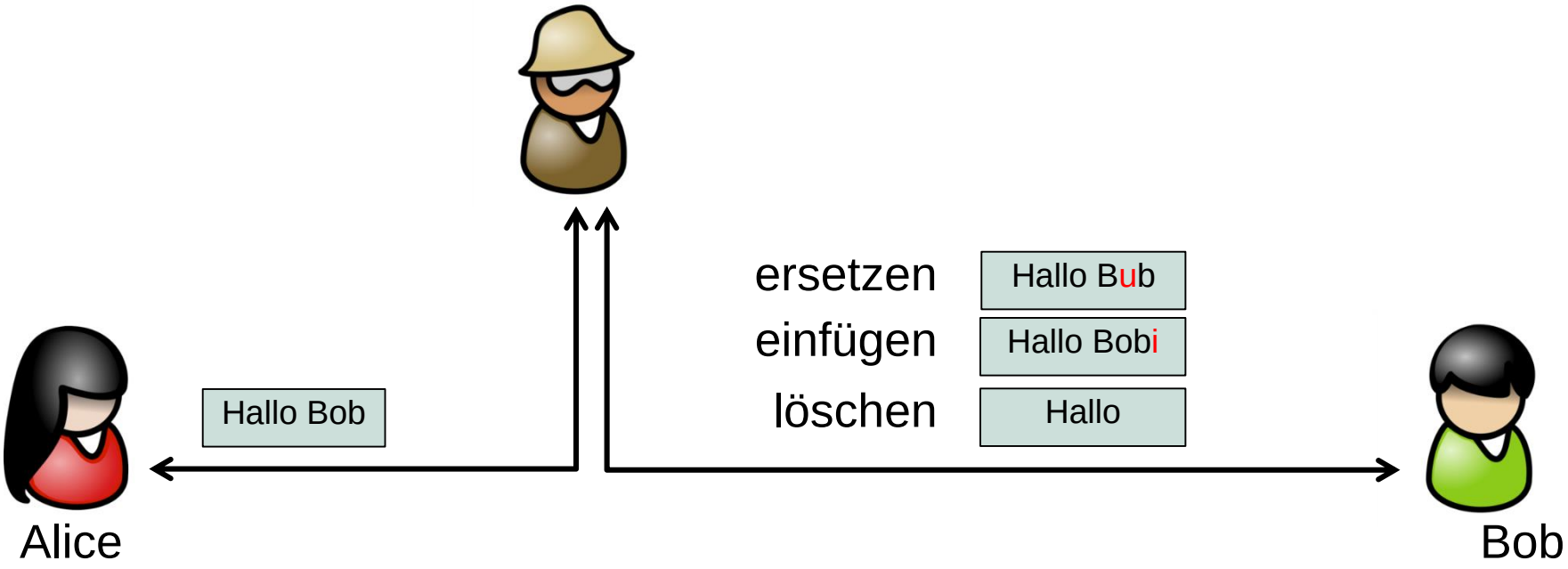
- Bausteine zur Umsetzung von Integrität
  - Tamper Proof Module (TPM)
  - Message Authentication Codes (MAC)
    - Hashfunktionen mit Schlüssel

… nur schwache Integrität ?

- Manipulationen in vielen Fällen nicht zu verhindern
- Sollten dann wenigstens nicht unbemerkt bleiben

Schutzziel Integrität

- Mögliche Manipulationen z.B.
  - Ersetzen von Daten
  - Einfügen in Daten
  - Löschen von Daten



Schutzziel Authentizität

- *Was ist Integrität der Daten wert, wenn nicht sicher ist von wem die Daten kommen?*

- Authentizität
  - Angegebene Quelle von Daten entspricht tatsächlicher Quelle + Datenintegrität
  - Echtheit von *Subjekten*
    - Bob will sicherstellen, dass er wirklich mit Alice kommuniziert
      → Authentifikation
  - Echtheit von *Daten*
    - Bob will sicherstellen, dass die Daten wirklich von Alice sind

- Bausteine zur Umsetzung von Authentizität
  - Zertifikate, Signaturen, gemeinsames Geheimnis

Pingo

http://pingo.upb.de/

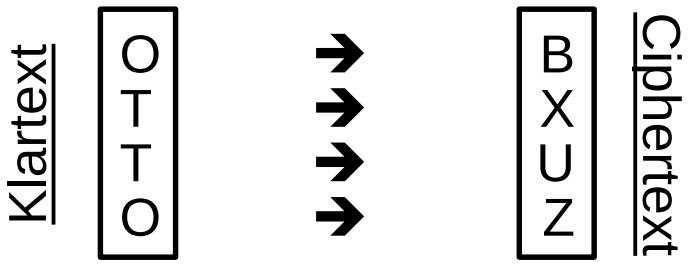
Kapitel 7.3

VERSCHLÜSSELUNG

Kryptografie

- Kryptografie untersucht und entwickelt Verfahren, mit denen ungesicherte *Klartexte* (engl. *Plaintext*) in für unautorisierte Instanzen nicht lesbare *Ciphertexte* (engl. *Cyphertext*) transformiert werden können.

- Berühmtes Beispiel: ENIGMA
  - Bedeutet *Rätsel*
  - Einsatz im zweiten Weltkrieg zur Verschlüsselung
  - Tastatur zur Klartext-Eingabe
  - Drei Walzen als Schlüssel
  - Glühlampenfeld für Ciphertext



Kryptografie

- Symmetrische Kryptografie
  - Beide kommunizierenden Instanzen besitzen einen gemeinsamen geheimen Schlüssel K
  - K wird sowohl zum Verschlüsseln als auch zum Entschlüsseln verwendet

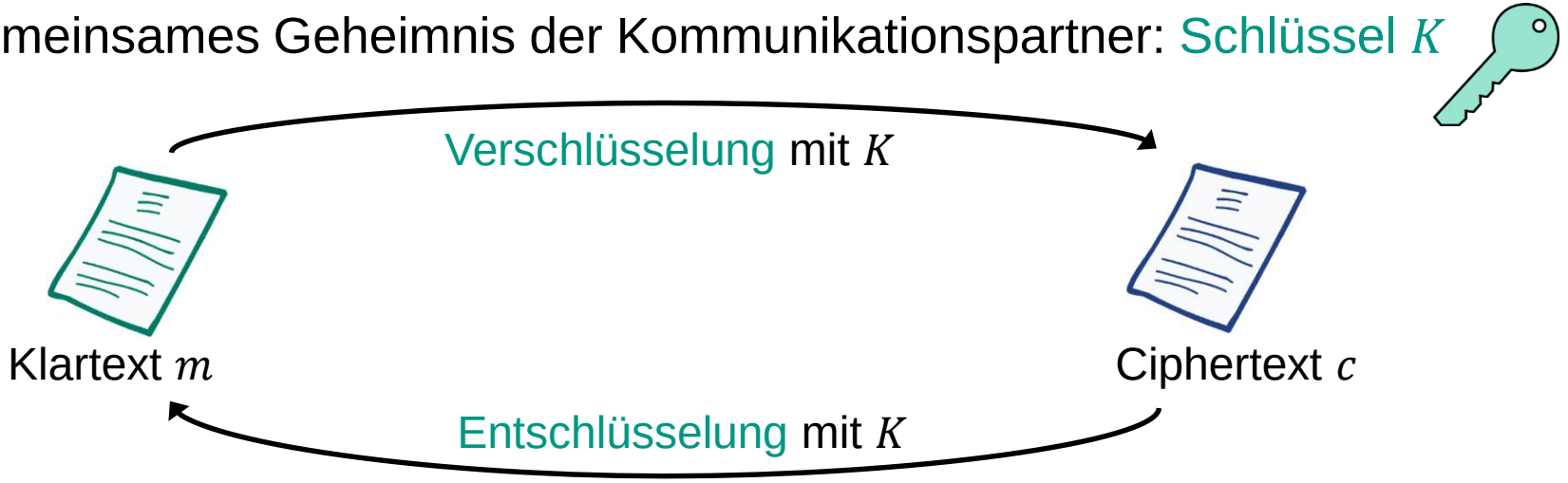
- Asymmetrische Kryptografie
  - Public-Key Cryptography
  - Jede Instanz besitzt einen geheimen, privaten Schlüssel und einen öffentlichen Schlüssel

- Kryptografische Protokolle
  - Anwendung kryptografischer Algorithmen
  - Wohldefinierte Folge von Aktionen

Kapitel 7.3.1 Symmetrische Verschlüsselung

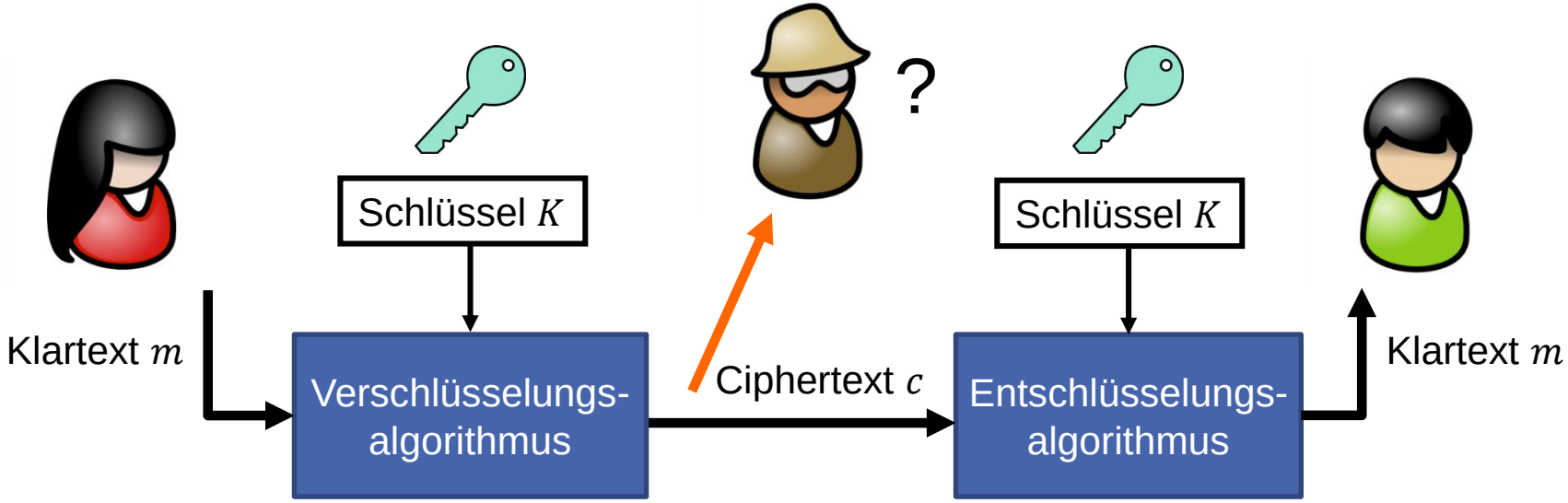
Grundlagen

- Gemeinsames Geheimnis der Kommunikationspartner: Schlüssel K



- Verschlüsselungsalgorithmus Enc
  - Eingabe: gemeinsamer Schlüssel K, Klartext m
  - Ausgabe: Ciphertext $\mathbf{c}$ - verschlüsselter Klartext. Es gilt: $c = \mathrm{Enc}(K, m)$
- Entschlüsselungsalgorithmus Dec
  - Eingabe: gemeinsamer Schlüssel K, Ciphertext $\mathbf{c}$
  - Ausgabe: Klartext m. Es gilt: $m = Dec(K, c) = Dec(K, Enc(K, m))$
- Problem: wie Schlüssel verteilen??

Schema


Schlüssel K
?
Schlüssel K
Klartext m
Verschlüsselungs-algorithmus
Ciphertext c
Entschlüsselungs-algorithmus
Klartext m

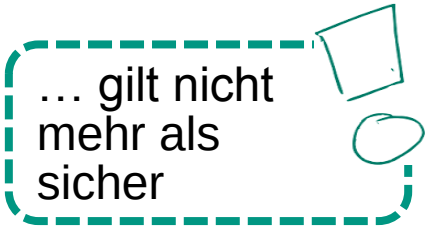
Zwei Klassen von Verfahren

- Blockchiffren
- Stromchiffren

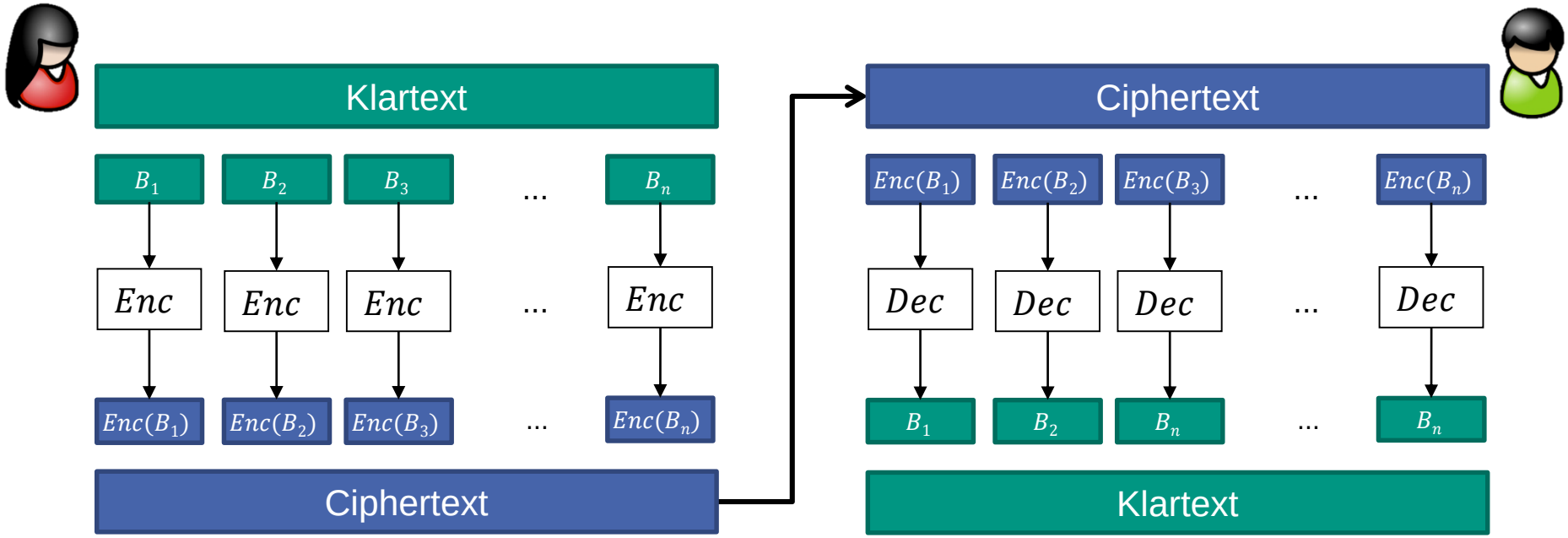
Blockchiffren

- Aufteilung der zu verschlüsselnden Daten in Blöcke der Länge k
- Daten werden blockweise verschlüsselt
  - Gleicher Schlüssel kann für aufeinanderfolgende Blöcke beibehalten werden
- Im Internet häufiger im Einsatz als Stromchiffren

- Beispiele
  - AES (Advanced Encryption Standard)
    - Blocklänge 128 Bit, Schlüssellänge 128, 192, 256 Bit
  - DES (Data Encryption Standard), 3DES
    - DES: Blocklänge 64 Bit, Schlüssellänge 56 Bit



Blockchiffren: Schema

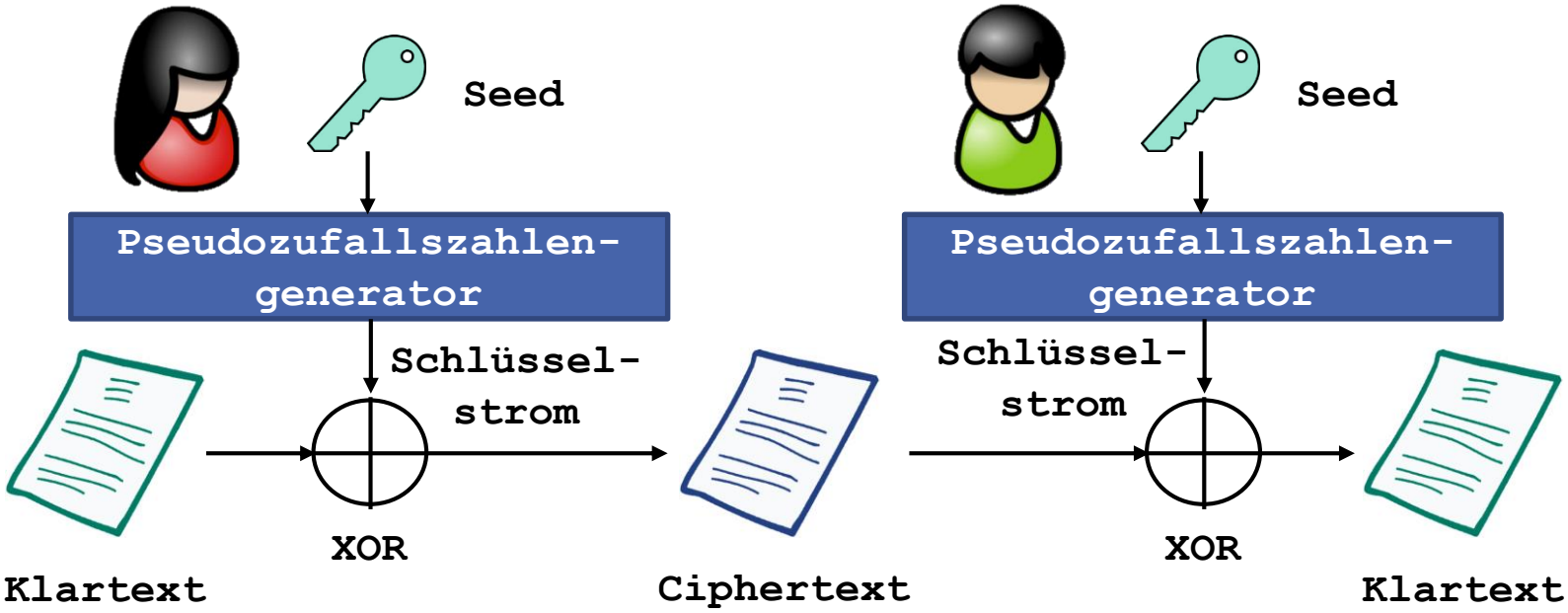


Stromchiffren

- Bit- oder Zeichenweises Verschlüsseln der Daten
  - Kein Warten bis vollständiger Block eingetroffen ist
- Schlüssel muss sich pro Einheit (typisch: 1 Byte) ändern: Nutzung eines Pseudozufallszahlengenerators
  - Gleicher *Seed* bei Sender und Empfänger
  - Generiert mit gleichem *Seed* deterministische Folge von Schlüsseln: Schlüsselstrom
  - Schlüsselstrom wesentlich für Sicherheit von Stromchiffren
    - Muss für Angreifer wie Zufallssequenz aussehen
    - Zufallszahlengeneratoren … hier nicht weiter vertieft
- Durchzuführende Operation (Verschlüsselung/Entschlüsselung): *Addition modulo 2*
- Kommen tendenziell besser mit limitierten Ressourcen aus
  - Beispiel: A5/1 in GSM zur Sprachverschlüsselung
- Auch im Internet im Einsatz, z.B. RC4

… gilt nicht mehr als sicher

Stromchiffren: Schema



Stromchiffren

- Klartext, Ciphertext und Schlüsselstrom bestehen aus Bits
  - Es gilt: $x_i, y_i, s_i \in \{0, 1\}$

- *Bitweise* Verschlüsselung: $Enc_{s_i}(x_i) = c_i \equiv x_i + s_i \; mod \; 2$
- *Bitweise* Entschlüsselung: $Dec_{s_i}(c_i) = x_i \equiv c_i + s_i \; mod \; 2$

- Gleiche Funktion für Ver- und Entschlüsselung

  - zu zeigen: $x_i \equiv Dec_{s_i}\left(Enc_{s_i}(x_i)\right)$

$$
\begin{aligned}
x_i &\equiv c_i + s_i \; mod \; 2 \\
&\equiv (x_i + s_i \; mod \; 2) + s_i \; mod \; 2 \\
&\equiv x_i + 2s_i \; mod \; 2 \\
&\equiv x_i + 0 \; mod \; 2 \\
&\equiv x_i \; mod \; 2
\end{aligned}
$$

Es gilt:
$2s_i \; mod \; 2 \equiv 0$

$x_i = 0; x_i \; mod \; 2 = 0$
$x_i = 1; x_i \; mod \; 2 = 1$

Kapitel 7.3.2 Asymmetrische Verschlüsselung

Grundlagen

- Auch als Public-Key-Kryptografie bezeichnet

- *Kein* gemeinsames Geheimnis. Zwei *unterschiedliche* Schlüssel
  - Öffentlicher Schlüssel (public key)
    - Ist *allen* bekannt (auch dem Angreifer)
  - Privater Schlüssel (private key)
    - Ist nur der kommunizierenden Instanz, z.B. Bob, bekannt

- Verschlüsselungsalgorithmus Enc()
  - Eingabe: öffentlicher Schlüssel K_E des Ziels, Klartext m
  - Ausgabe: Ciphertext $\mathbf{c}$ - verschlüsselter Klartext. Es gilt: $c = Enc_{K_E}(m)$

- Entschlüsselungsalgorithmus Dec()
  - Eingabe: privater Schlüssel K_D des Ziels, Ciphertext $\mathbf{c}$
  - Ausgabe: Klartext m. Es gilt: $m = Dec_{K_D}(c)$

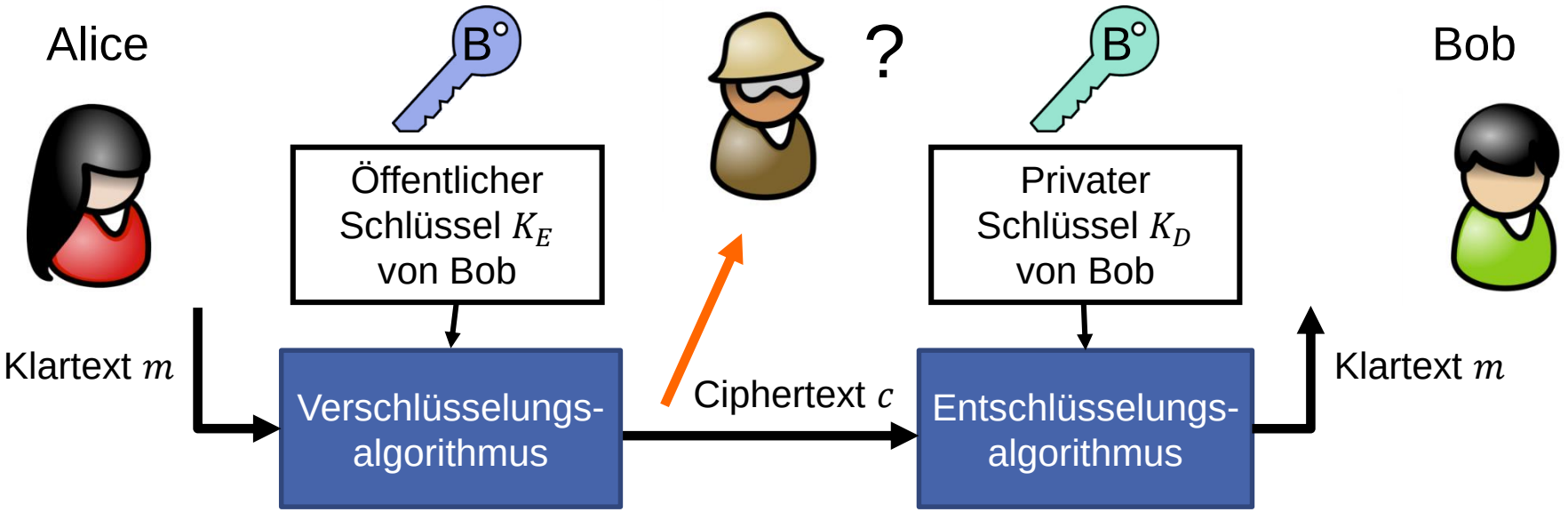
Grundlagen

- Berechnung des privaten Schlüssels bei Kenntnis öffentlicher Schlüssel, Verschlüsselungsalgorithmus und ggf. Ciphertext praktisch nicht möglich

- Benötigt werden Ein-Wege-Funktionen
  - $m = f(\mathrm{x})$ ist einfach zu berechnen
  - $f^{-1}(m)$ ist „computationally infeasible“
  - Es gilt
    - $Dec_{K_D}\left(Enc_{K_E}(m)\right) = m$

- Beliebte Beispiele
  - Integer-Faktorisierung (→ RSA)
  - Diskreter Logarithmus (→ Diffie-Hellman-Schlüsselaustausch)
  - Elliptische Kurven (Generalisierung der Algorithmen basierend auf diskretem Logarithmus)

Schema


Alice
Bob
?
Öffentlicher Schlüssel K_E von Bob
Privater Schlüssel K_D von Bob
Klartext m
Verschlüsselungs-algorithmus
Ciphertext c
Entschlüsselungs-algorithmus
Klartext m

Schlüssellängen und Sicherheitsniveau

- Algorithmen benötigen lange Operanden und Schlüssel
- Sicherheitsniveau n Bit: bester bekannter Angriff benötigt 2^n Schritte

Algorithmen-familie	Crypto-system	Sicherheitsniveau (bits)			
		80	128	192	256
Faktorisierung	RSA	1.024 bit	3.072 bit	7.680 bit	15.360 bit
Diskreter Logarithmus	DH, DSA, Elgamal	1.024 bit	3.072 bit	7.680 bit	15.360 bit
Elliptische Kurven	ECDH, ECDSA	160 bit	256 bit	384 bit	512 bit
Symmetrischer Schlüssel	AES, 3DES	80 bit	128 bit	192 bit	256 bit

- Sicherheit kostet!
  - Verschlüsselung benötigt Zeit (speziell auf Geräten mit eingeschränkten Ressourcen)

[Paar10]

„Das“ asymmetrische Verfahren: RSA

- Entwurf von Ron Rivest, Adi Shamir und Len Adleman
  - Zugrundeliegendes Problem: Integer-Faktorisierung
  - Basis: Nutzung modularer Arithmetik
    - Addition: $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
    - Subtraktion: $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
    - Multiplikation: $[(a \bmod n) * (b \bmod n)] \bmod n = (a * b) \bmod n$
    - ➔ $(a \bmod n)^d \bmod n = a^d \bmod n$

- Beispiel (x=14, n=10, d=2):
  - $(14 \bmod 10)^2 \bmod 10 = 16 \bmod 10 = 6$
  - $14^2 \bmod 10 = 196 \bmod 10 = 6$

Deutlich langsamer als symmetrische Verfahren.
Hauptnutzung: Austausch eines Schlüssels für symmetrische Verfahren.

Verschlüsselung und Entschlüsselung

- RSA arbeitet auf Integerzahlen (Integerring $\mathbb{Z}_n$)
- Nachricht als Bitmuster m, Binärwert von $m < n$
  - In Blöcke unterteilt mit Blockgrößen kleiner oder gleich $\mathrm{ld}(n) + 1$
  - Praktisch verwendete Blockgröße: 2^k bit mit $2^k < n \leq 2^{k+1}$
  - Jedes Bitmuster kann durch eine Integerzahl dargestellt werden
    - Beispiel: $m \coloneqq 10010001$ als 145

- Verschlüsselung: $Enc_{K_E}(m) = c \equiv m^e mod\ n$

- Entschlüsselung: $Dec_{K_D}(c) = m \equiv c^d mod\ n$

Schlüsselerzeugung

- Wähle zwei große Primzahlen p und q
- Berechne

$$n = p * q$$

und

$$z = (p-1)(q-1)$$

- Wähle eine Zahl $1 < e < z$, die teilerfremd zu z ist
- Finde d, so dass $ed - 1$ durch z dividierbar ist, also

$$e * d \; mod \; z \equiv 1$$

- Öffentlicher Schlüssel: $\mathrm{K_E} := (n, e)$ 
- Privater Schlüssel: $\mathrm{K_D} := (d)$ 

Beispiel

- Bob berechnet Schlüssel
  - $p = 5$ und $q = 7$ → $n = 35$ und $z = 24$
  - Wählt $e = 5$ und $d = 29$ (mit $ed - 1$ durch z teilbar)
  - Privater Schlüssel $K_D := (d) = 29$ 
  - Öffentlicher Schlüssel $K_E := (n, e) = (35,5)$ 
- Alice verschlüsselt Bitmuster $00001000 = 12 = m$
  - $Enc_{K_E}(m) = m^e mod\ n = 12^5\ mod\ 35 = 248.832\ mod\ 35 = 17 := c$
- Alice sendet Ciphertext 17 an Bob
- Bob entschlüsselt 17 zu 12 (und Klartext-Bitmuster 00001000)
  - $Dec_{K_D}(c) = c^d\ mod\ n = 17^{29}\ mod\ 35 = 12$

481.968.572.106.750.915.091.411.825.223.071.697
~480 Quintilliarde

Warum RSA funktioniert

- Zu zeigen: $c^d \; mod \; n \; = \; m$
- Mit $c \; = \; m^e \; mod \; n$

- Vorausgesetzt

  $n = p * q$ und $z \; = \; (p-1)(q-1)$ und $\; e * d \; mod \; z \; \equiv 1$

  - Dann gilt für jedes x und y

    $x^y \; mod \; n \; = \; x^{(y \; mod \; z)} \; mod \; n$

    angewandt mit $x = m$ und y $= ed$

$$
\begin{aligned}
c^d \; mod \; n \; &= \; (m^e \; mod \; n)^d \; mod \; n \\
&= \; m^{ed} \; mod \; n \\
&= \; m^{(ed \; mod \; z)} mod \; n \\
&= \; m^1 \; mod \; n \\
&= \; m
\end{aligned}
$$

Warum RSA sicher ist

- Voraussetzung
  - Öffentlicher Schlüssel K_E mit e und n ist bekannt 
  - Privater Schlüssel K_D mit d ist geheim 

- Angreifer benötigt zum entschlüsseln den privaten Schlüssel K_D
  - Aufgabe: Berechne $K_D(d)$ aus $K_E(e, n)$
    - Für d gilt: $e * d \; mod \; z \equiv 1$
    - Dabei ist $z = (p-1)(q-1)$
    - Für p und q gilt: $n = p * q$
    - ➔ Zerlege n in Primfaktoren p und q, berechne z, berechne e

- Grundlegende Annahme
  - Es existiert kein effizienter Algorithmus zur Primfaktorzerlegung von großen Zahlen n

Symmetrisch vs. Asymmetrisch

- Symmetrische Verfahren: AES
  - \+ Geringe Komplexität, höhere Effizienz (HW-Implementierung)
  - − Aufwändige Schlüsselverteilung
  - − Aufwändige Realisierung von digitalen Unterschriften
- Asymmetrische Verfahren: RSA
  - \+ Einfache Schlüsselverteilung
  - \+ Digitale Unterschriften einfach zu realisieren
  - − Höhere Komplexität, geringere Effizienz (trotz HW-Impl.)
- Empfehlung: Kombination (Hybrides Verfahren)
  - Beginn einer Interaktion mit asymmetrischem Verfahren
    - Austausch von symmetrischen Schlüsseln über gesicherten Kanal
  - Wechsel zu symmetrischem Verfahren

Welche Verfahren sind denn nun sicher?

- Blockchiffren
  - z.B. AES-128, AES-192, AES-256
- Stromchiffren
  - z.B. Salsa20/20, ChaCha
- Public-Key-Kryptografie
  - z.B. RSA-4096, DLIES, ECIES

- Informationsquellen
  - Bundesamt für Sicherheit in der Informationstechnik (BSI) https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/technischerichtlinien_node.html (2017)
  - National Institute of Standards and Technology (NIST) http://csrc.nist.gov/groups/ST/toolkit/ (2016)
  - European Union Agency for Network and Information Security (ENISA) https://www.enisa.europa.eu/topics/data-protection/security-of-personal-data/cryptographic-protocols-and-tools (2014)

Pingo

http://pingo.upb.de/

Kapitel 7.4

INTEGRITÄTSSICHERUNG

Integritätssicherung

- Gefahr eines Man-in-the-Middle-Angriffs!

- Ziel
  - Daten sollen exakt so beim Empfänger eintreffen wie vom Absender versendet
  - → Manipulationen können im Internet nicht verhindert werden, sollen aber eindeutig erkannt werden … *schwache* Integrität

Hashfunktion: Eigenschaften

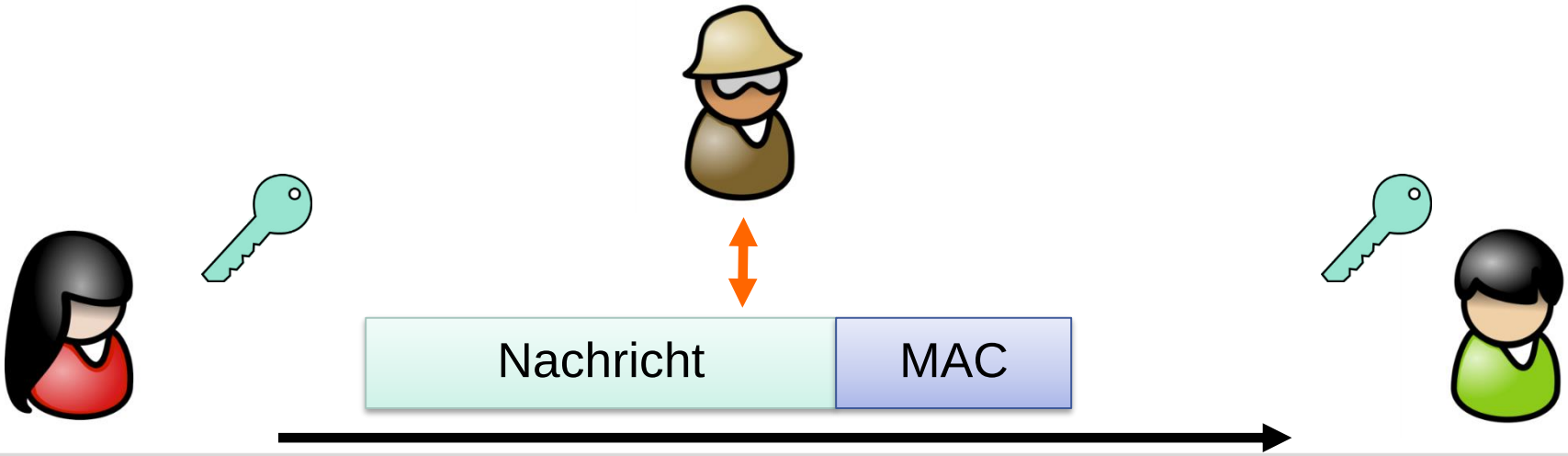
C I A

- Einwegfunktion/Urbildresistenz: zu gegebenem b ist es schwierig, ein a zu finden, so dass gilt $H(a) = b$
  - Anschauliches Beispiel: Telefonbuch
    - Nummer zu Name leicht; Name zu Nummer nicht leicht
- Schwache Kollisionsresistenz
  - Für gegebenes a ist es schwierig, ein $\bar{a} \neq a$ zu finden, so dass gilt $H(a) = H(\bar{a})$
- Starke Kollisionsresistenz
  - Es ist schwierig, zwei verschiedene Werte a und $\bar{a}$ aus der Urbildmenge A zu finden, so dass gilt $H(a) = H(\bar{a})$
- Beliebige Länge der Nachricht
  - $H(m)$ kann auf Nachrichten beliebiger Länge angewandt werden
- Hashwert fester Länge
  - $H(m)$ liefert Hashwert z einer festen Länge
- Effizient
  - $H(m)$ ist relativ einfach zu berechnen
- Beispiele: MD5, SHA-1 (gelten als nicht mehr sicher), SHA-2, SHA-3

Kapitel 7.4.1 Message Authentication Code

Message Authentication Code (MAC)

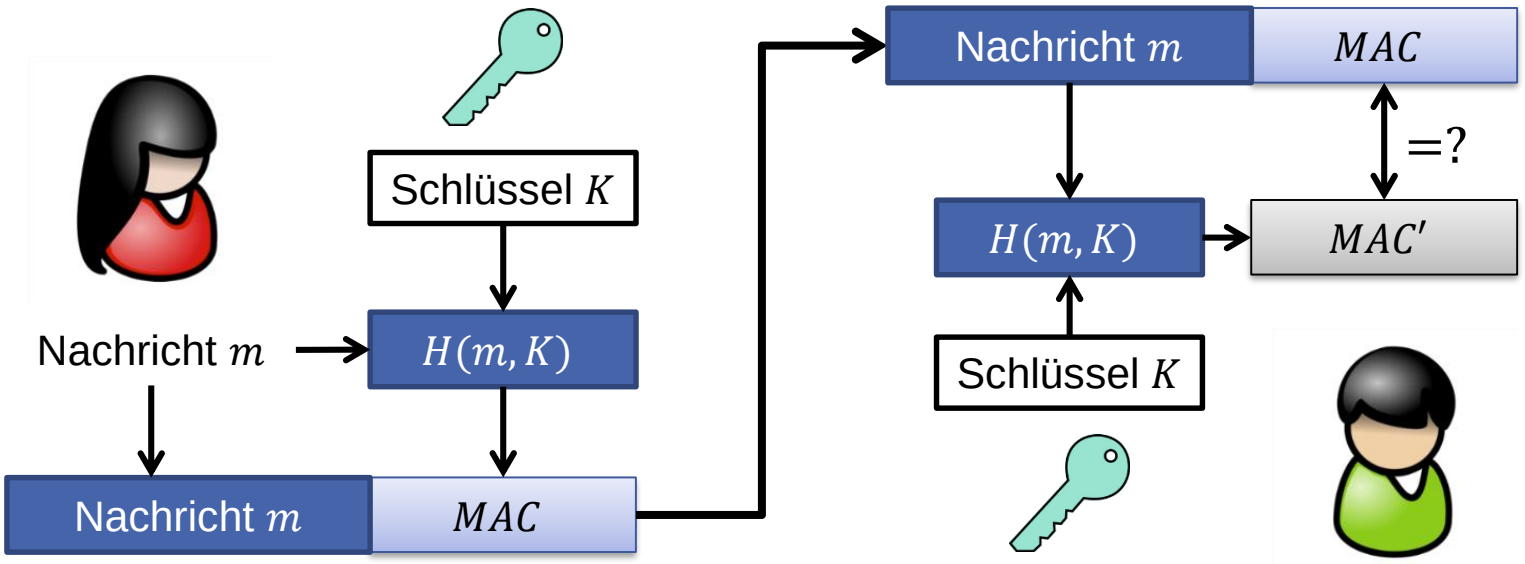
- Ziel
  - Empfänger erkennt Manipulationen an den gesendeten Daten
- Vorgehensweise
  - Alice erstellt Nachricht m und berechnet Hashwert $H(m)$
    - Feste Länge von $H(m)$, beliebige Länge von m
  - Alice hängt Hashwert an Nachricht an
- Voraussetzung
  - Alice und Bob besitzen gemeinsamen symmetrischen Schlüssel
- Schutzziele
  - Nachrichtenintegrität, -authentizität, keine Nichtabstreitbarkeit



Message Authentication Code

- Berechne Message Authentication Code auf Basis einer Hashfunktion H()
  - über zu sichernde Daten und
  - über geheimen Schlüssel
  - *Keyed-Hashing for Message Authentication*

→ Angreifer kann gültigen Hashwert nach Veränderung der Daten nicht neu berechnen



Kapitel 7.4.2 Digitale Signatur

Digitale Signatur

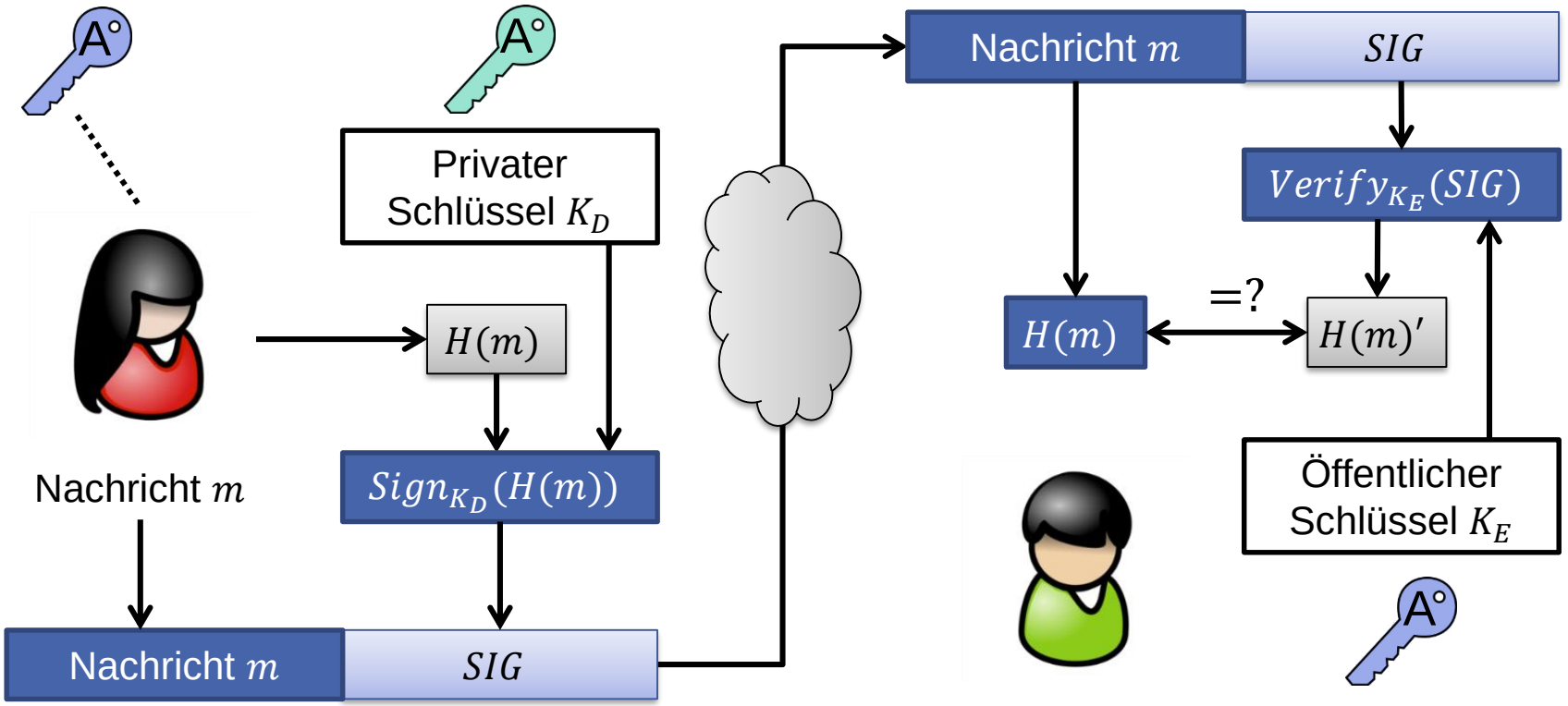
- Eigenschaften
  - Vergleichbar denjenigen von „normalen“ Unterschriften
  - Nachweisbar, nicht fälschbar, kann nicht geleugnet werden
    - Nichtabstreitbarkeit
  - Basiert auf asymmetrischer Kryptografie

- Vorgehensweise
  - Dokument m wird mit privatem Schlüssel K_D signiert: $\mathrm{SIG} = Sign_{K_D}(m)$
    - SIG dient als Signatur und wird mit m gesendet
  - Empfänger überprüft Signatur mit öffentlichem Schlüssel K_E
    - $Verify_{K_E}(SIG) = m'$
    - Signatur ist korrekt, falls gilt $m' = m$
  - Üblicherweise wird Signatur nicht über komplettes Dokument berechnet, sondern über wesentlich kürzeren Hash-Wert des Dokuments

- Alice kann damit verifizieren
  - Bob hat das Dokument unterschrieben
  - Kein anderer als Bob hat das Dokument unterschrieben
  - Bob hat das Dokument m und nicht das Dokument m' unterschrieben

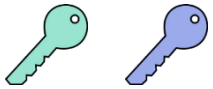
Digitale Signatur

- Hier: Signatur über Hashwert des Dokuments

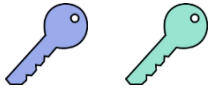


Umsetzung mit RSA

- Erinnerung: für RSA gilt
  - $Dec_{K_D}\left(Enc_{K_E}(m)\right) = m$
- Aber auch
  - $Dec_{K_E}\left(Enc_{K_D}(m)\right) = m$



„Tausch“ von öffentlichem und privaten Schlüssel



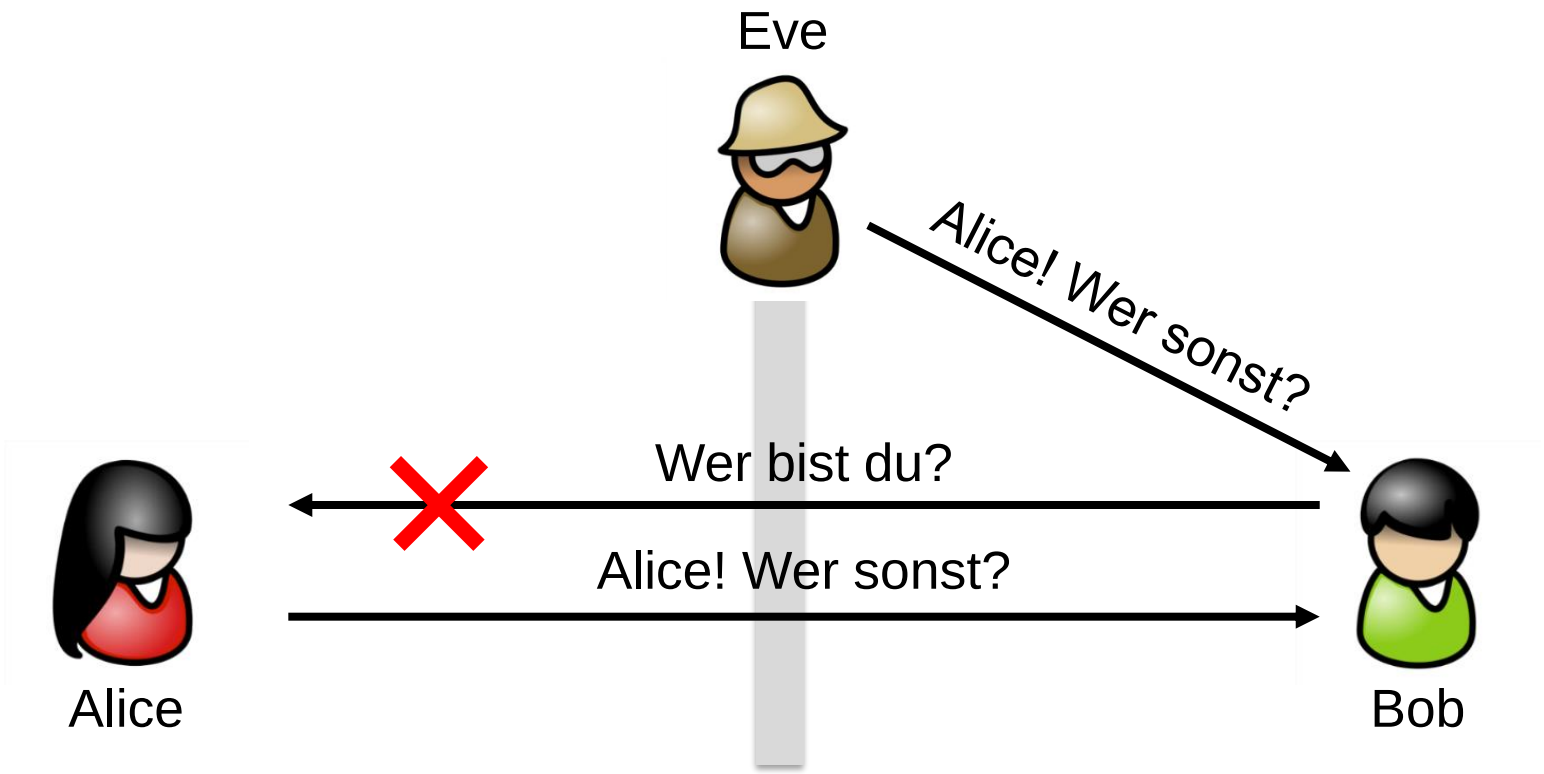
➔ Realisierung von digitalen Signaturen mit RSA

- Signaturerstellung: $Sign_{K_D}(m) := Enc_{K_D}(m)$ 
  - Nutzung des privaten Schlüssels zur Verschlüsselung
- Überprüfung: $Verify_{K_E}(SIG) := Dec_{K_E}(SIG)$ 
  - Nutzung des öffentlichen Schlüssels zur Entschlüsselung

Kapitel 7.4.3 Authentifizierung des Kommunikationspartners

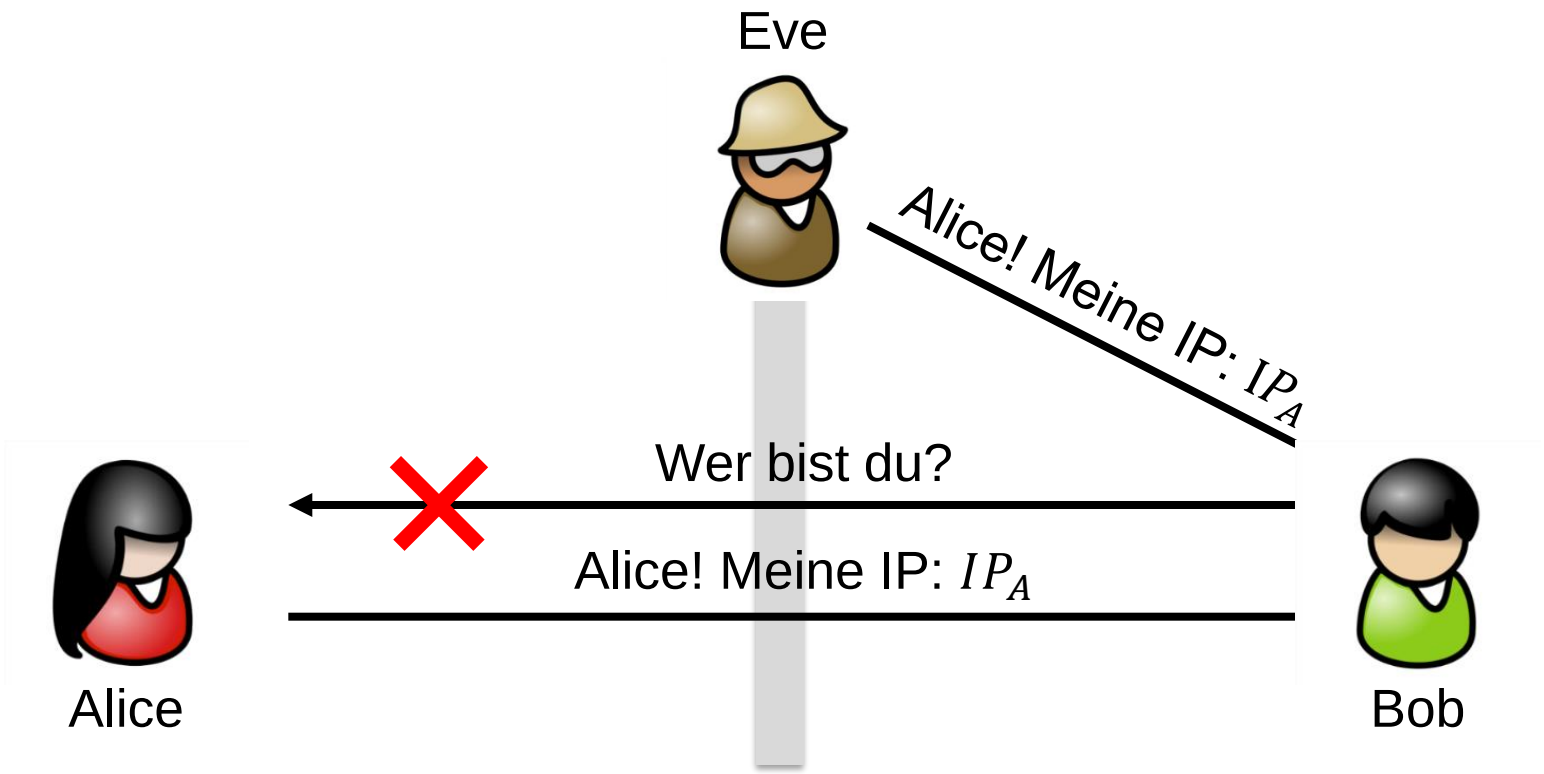
Ziel

- Bob will sicherstellen, dass er tatsächlich mit Alice kommuniziert



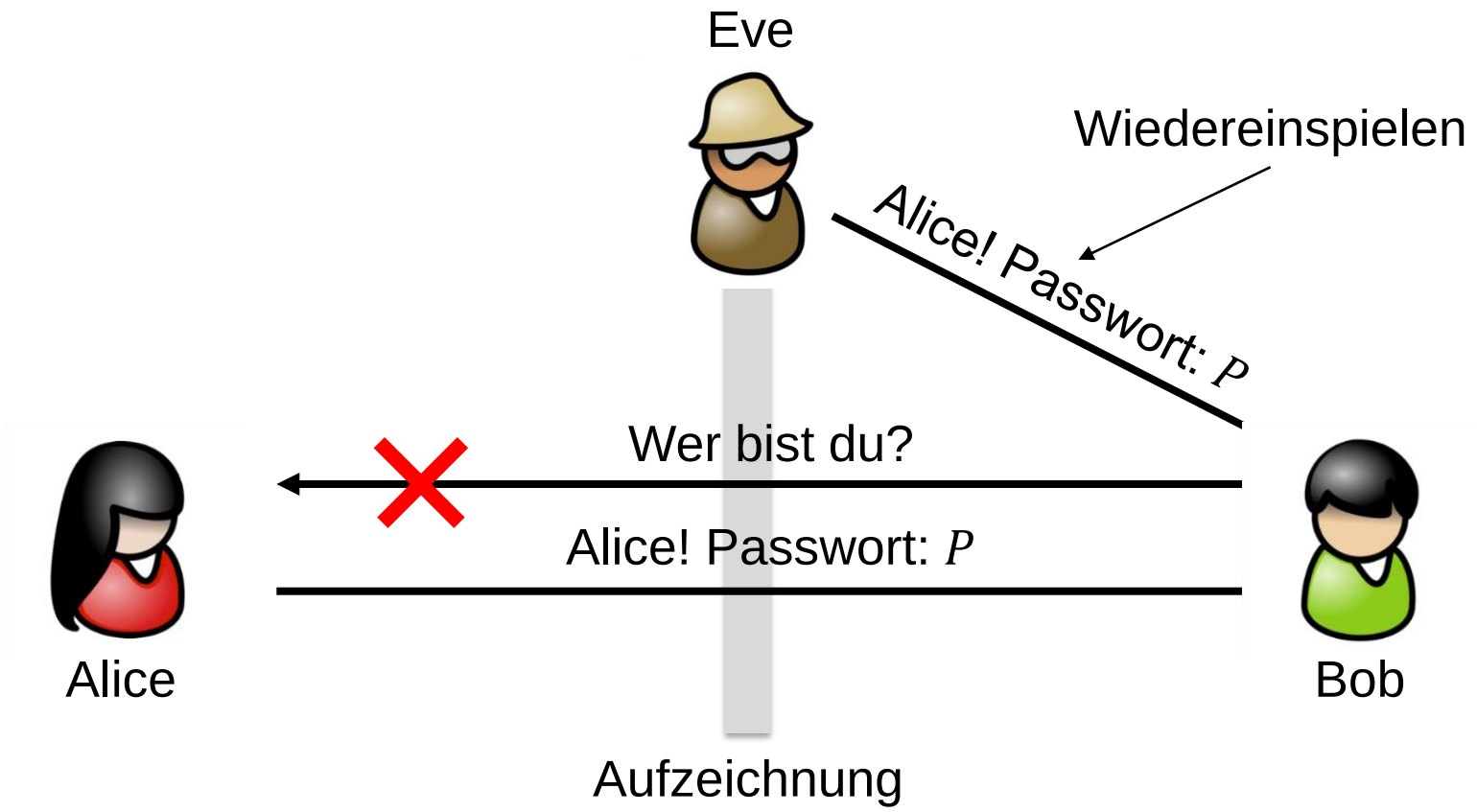
Abhilfe?

- Alice schreibt ihre IP-Adresse zur Nachricht



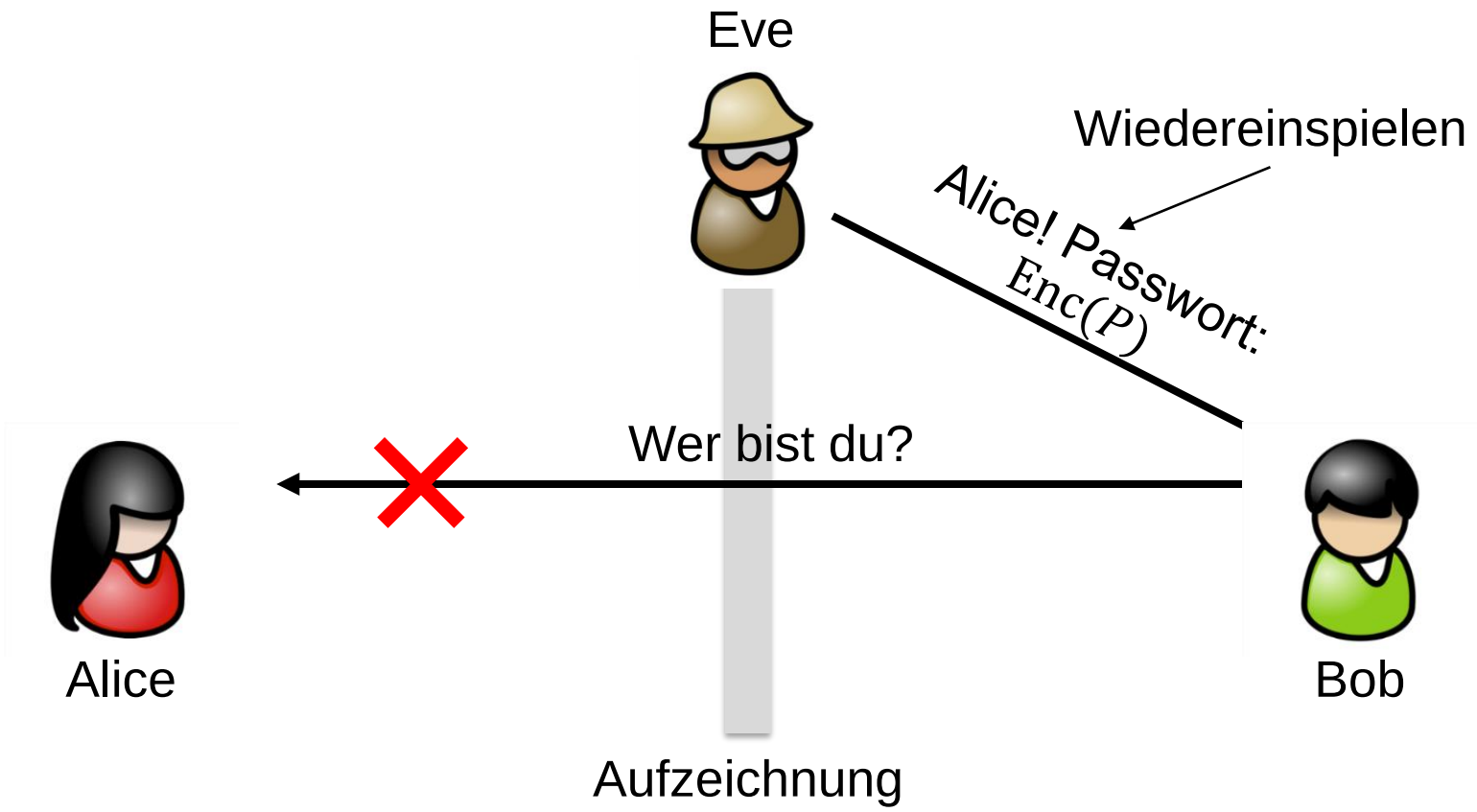
Vielleicht so?

- Alice sendet ein geheimes Passwort mit der Nachricht



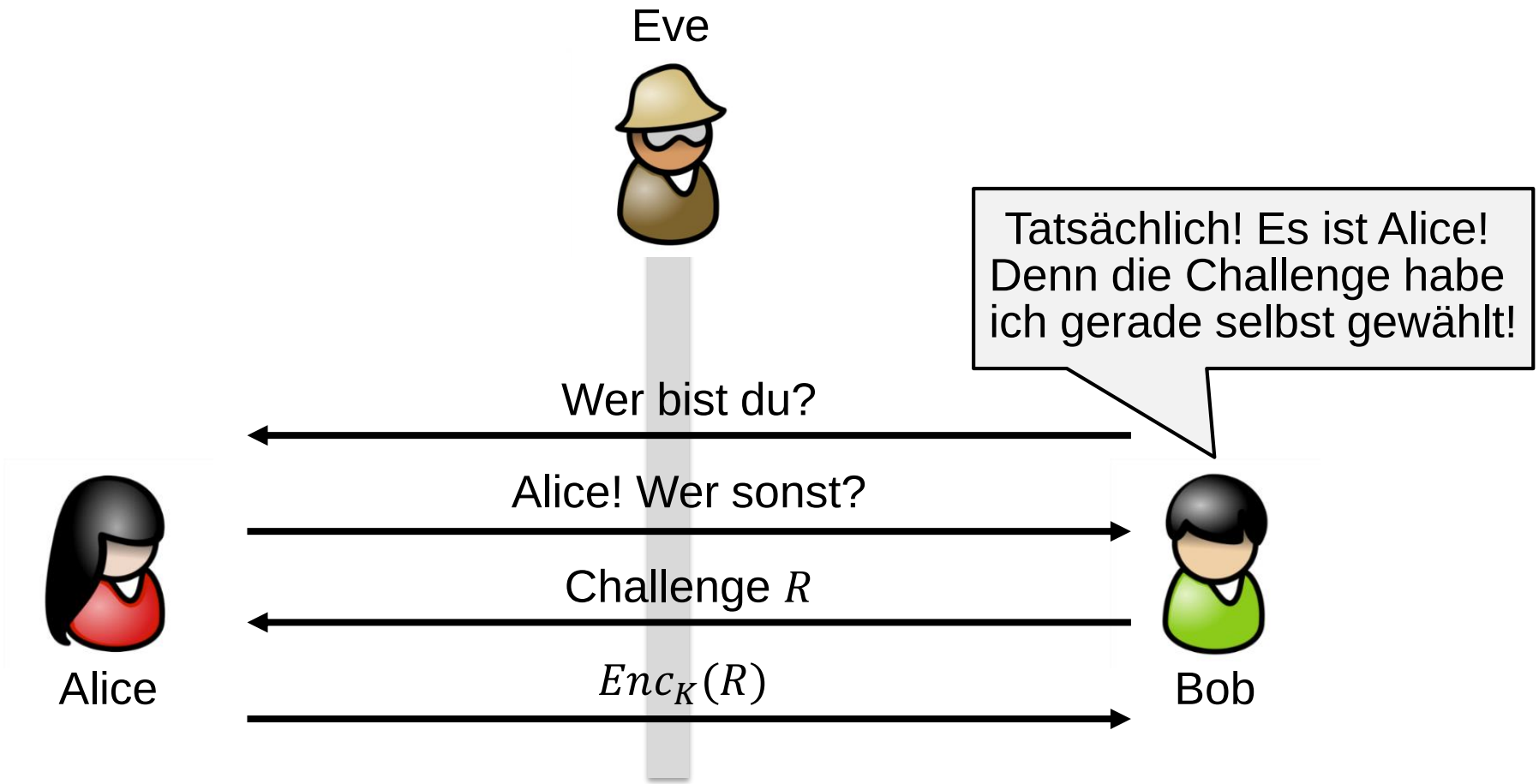
Vielleicht hilft ja verschlüsseln?

- Alice verschlüsselt ihr Passwort und sendet es mit der Nachricht
  - Ändert gar nichts!



Jetzt aber … mit gemeinsamem Schlüssel K

- Nutzung eines Challenge-Response-Verfahrens



- Aber: *Wie gelangen Alice und Bob an den gemeinsamen Schlüssel?*

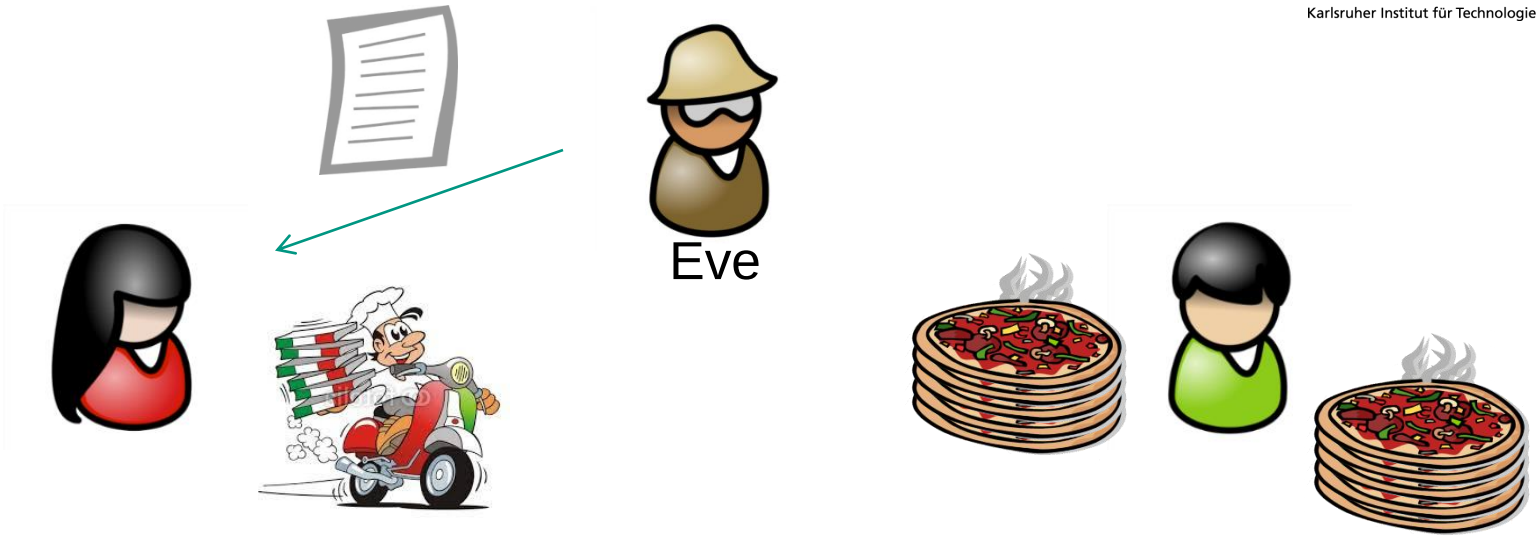
… und mit asymmetrischer Kryptografie?

- Szenario: Eine ganz normale Pizza-Bestellung …

- Alice
  - Arbeitet bei einem Online-Pizzadienst
- Bob
  - Sendet Bestellung mit Heimadresse im Klartext
  - Nachricht enthält Bob‘s Signatur und seinen öffentlichen Schlüssel
- Alice
  - *… ist die Bestellung wirklich von Bob ?*
  - Überprüft Signatur mit Bob´s, der Nachricht hinzugefügtem, öffentlichem Schlüssel

[KuRo10]

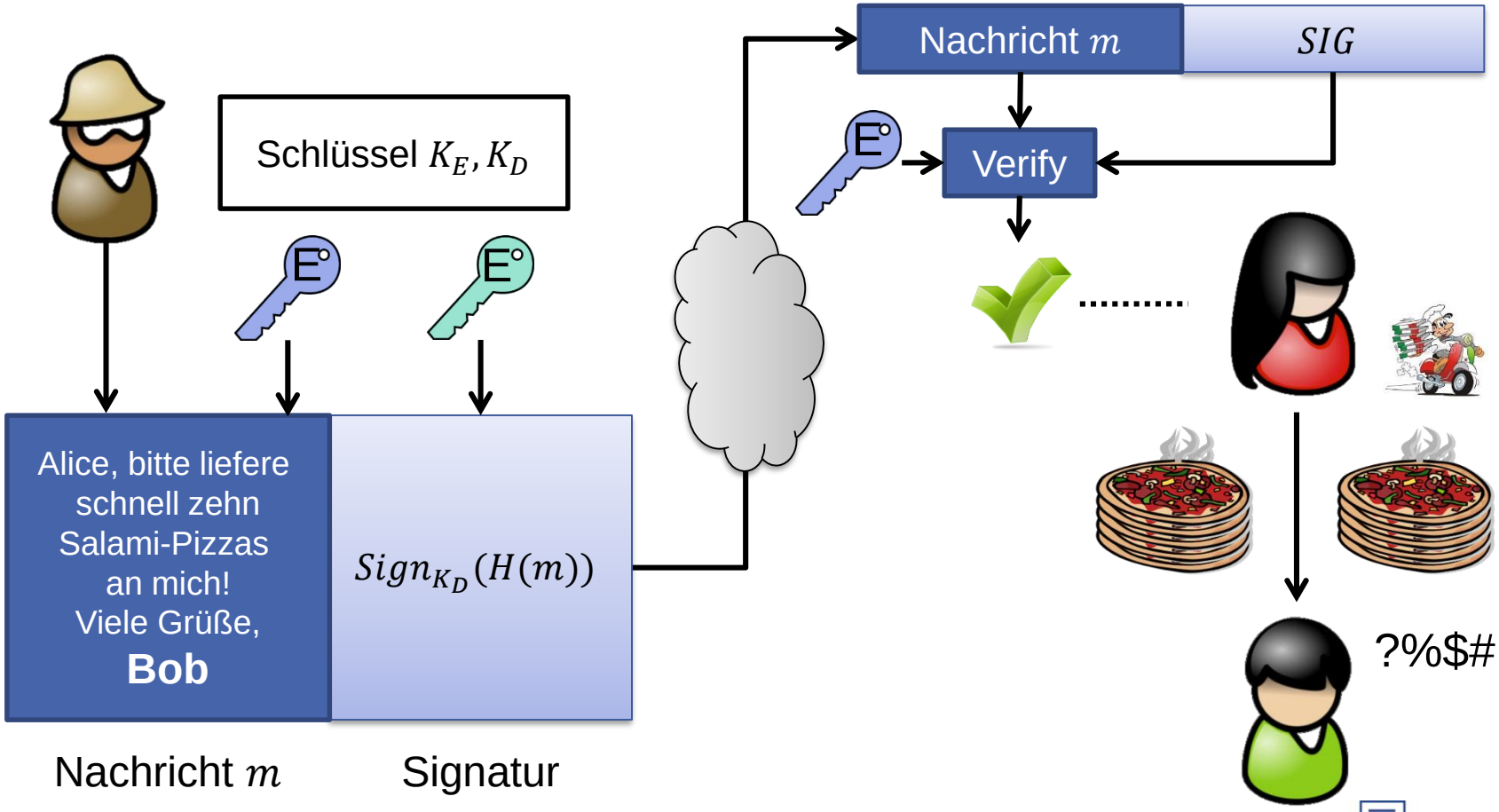
… aber



- Eve mischt sich ein
  - Gibt vor Bob zu sein, schickt Bestellung an Alice
  - Nachricht enthält Eve´s öffentlichen Schlüssel und eine Signatur
    - Signatur mit Eve´s privatem Schlüssel erstellt
- Alice
  - Nutzt „Bob´s“ öffentlichen Schlüssel
    - … naja, also den von Eve
- Bob
  - … ist ziemlich überrascht über die Pizzalieferung

[KuRo10]

Eve´s Angriff schematisch dargestellt



[KuRo10]

Grundlegendes Problem

- Signatur
  - Sichert Integrität der Nachricht
  - Authentizität wird nicht gesichert !

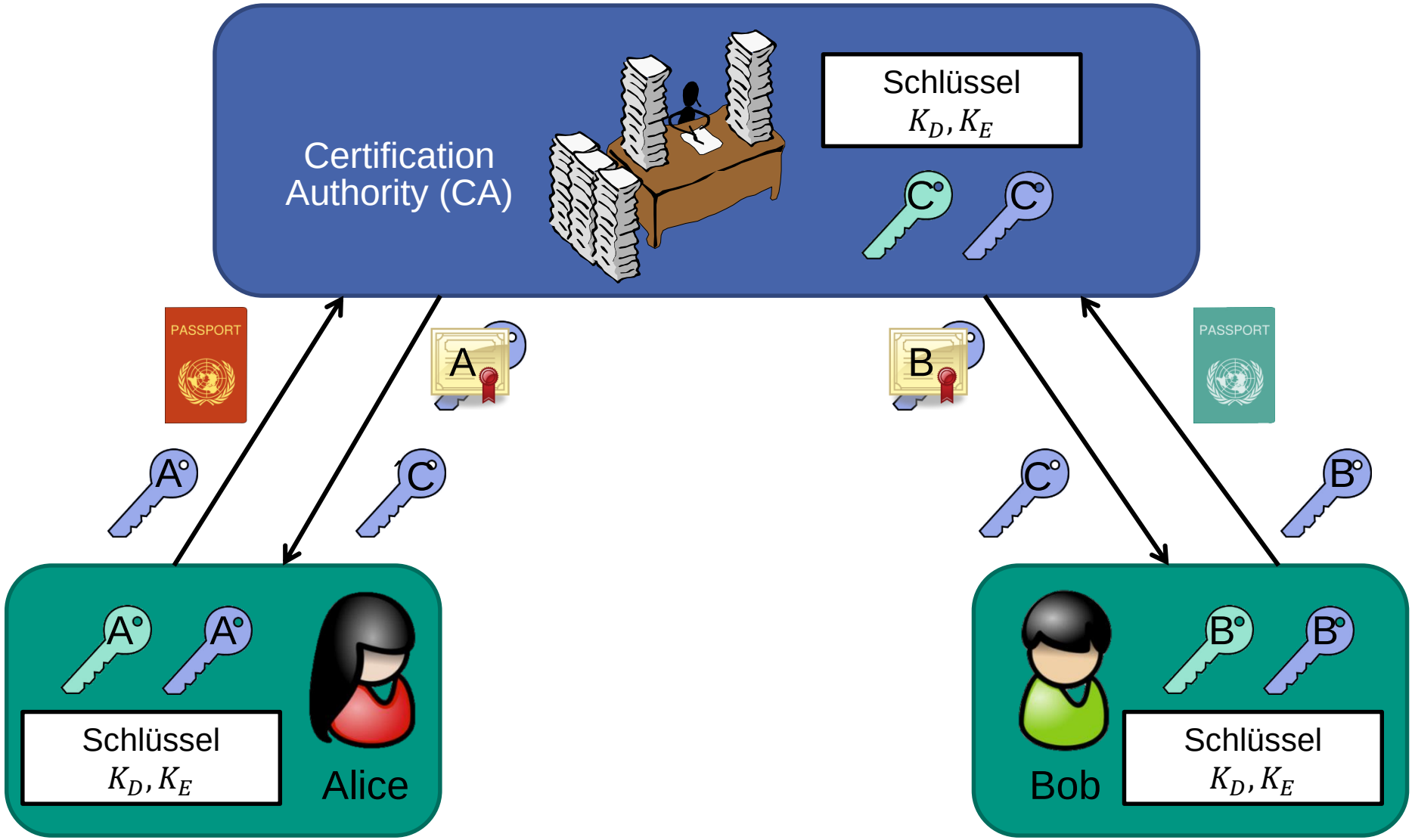
- Was fehlt?
  - Überprüfung ob der öffentliche Schlüssel demjenigen gehört, mit dem man kommunizieren möchte
  - Im Beispiel
    - Alice muss überprüfen können, ob der öffentliche Schlüssel zu Bob gehört

- Abhilfe
  - Digitale Zertifikate
    - Genauer ID-Zertifikate

Digitale Zertifikate

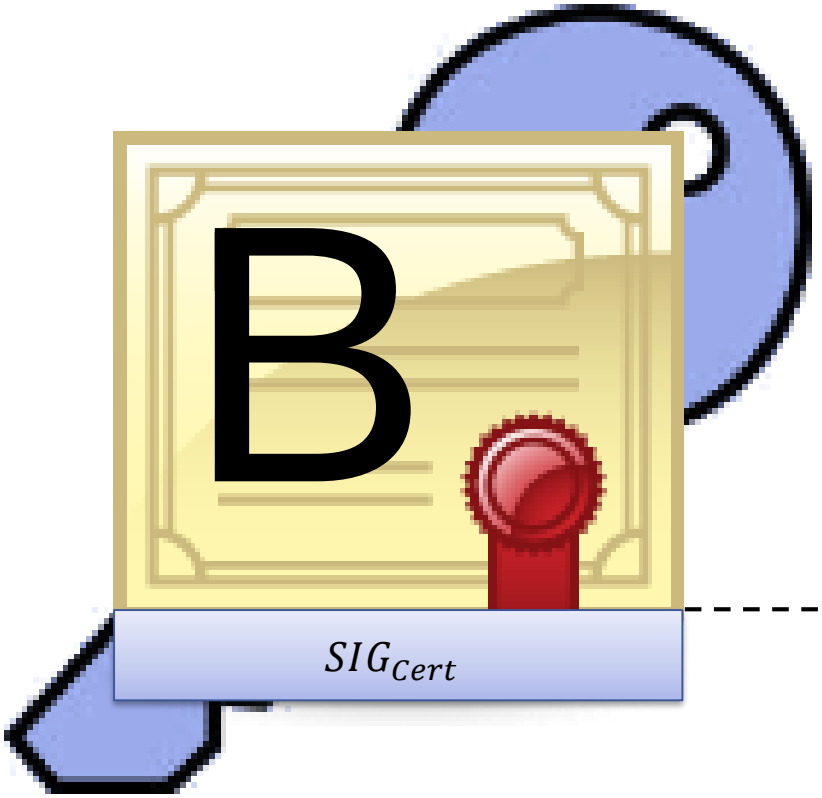
- Problemstellung
  - Authentifizierung eines Sachverhaltes, den man selbst nicht überprüfen kann
  - man verlässt sich auf vertrauenswürdige Dritte, die ihn schon kontrolliert haben
- Zertifikat ist digitales Dokument, in dem eine Instanz einen bestimmten Sachverhalt mittels digitaler Signatur bestätigt
  - erzeugt Vertrauen in den Sachverhalt
- Zertifikate werden von vertrauenswürdiger Instanz erstellt
  - Certification Authority (CA)
- ID-Zertifikate
  - öffentlicher Schlüssel → eindeutiger Name (*Identität*)
  - Authentifikation von öffentlichen Schlüsseln
    - Bindet öffentlichen Schlüssel an Identität

ID-Zertifikate für Bob und Alice



Grober Aufbau eines ID-Zertifikates

- … hier für Bob



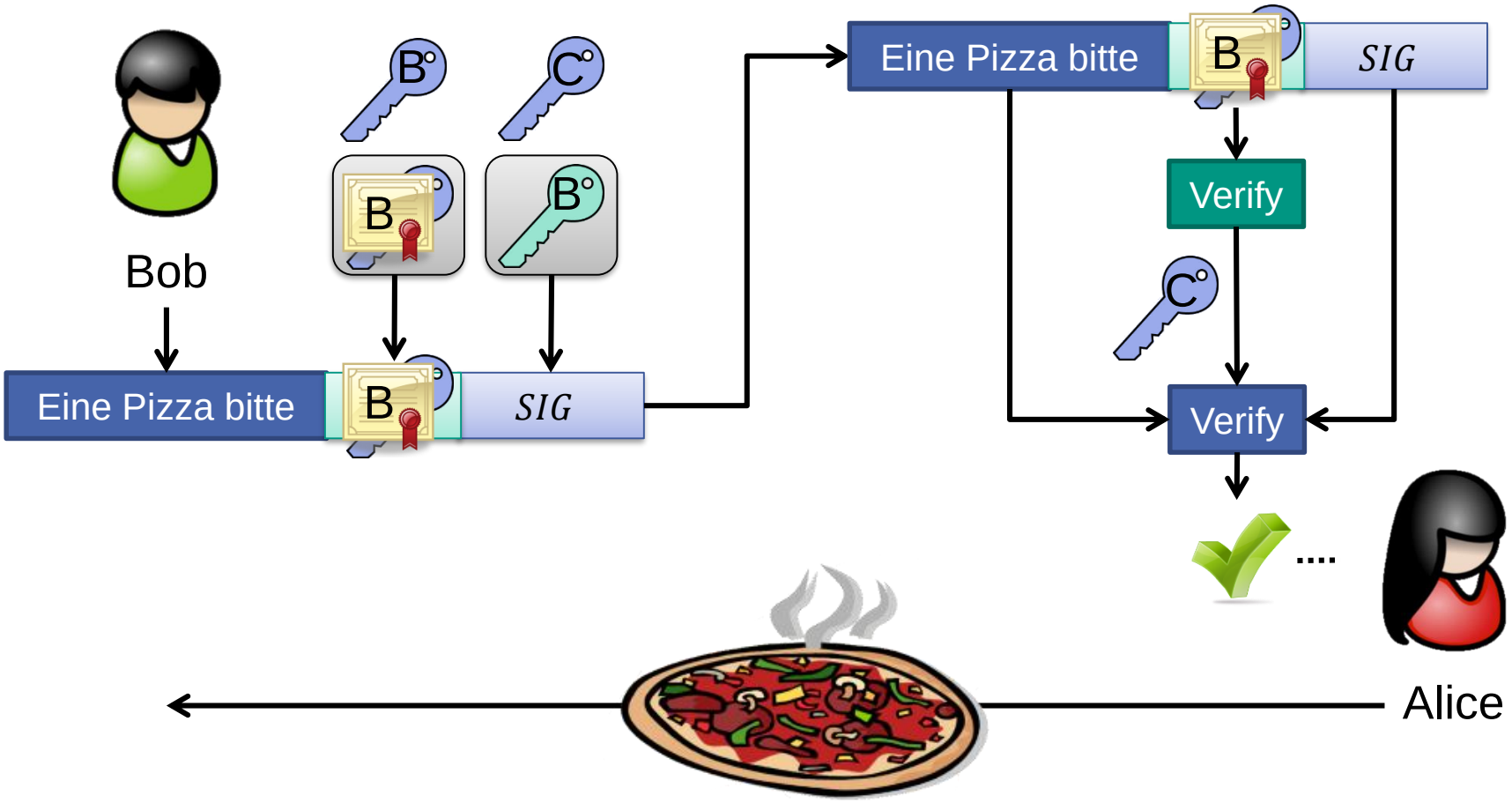
Eindeutiger Name der Identität

Öffentlicher Schlüssel der Identität K_E

Digitale Signatur der CA

Beispiel mit ID-Zertifikaten

- „Gelungene“ Pizza-Bestellung



Pingo

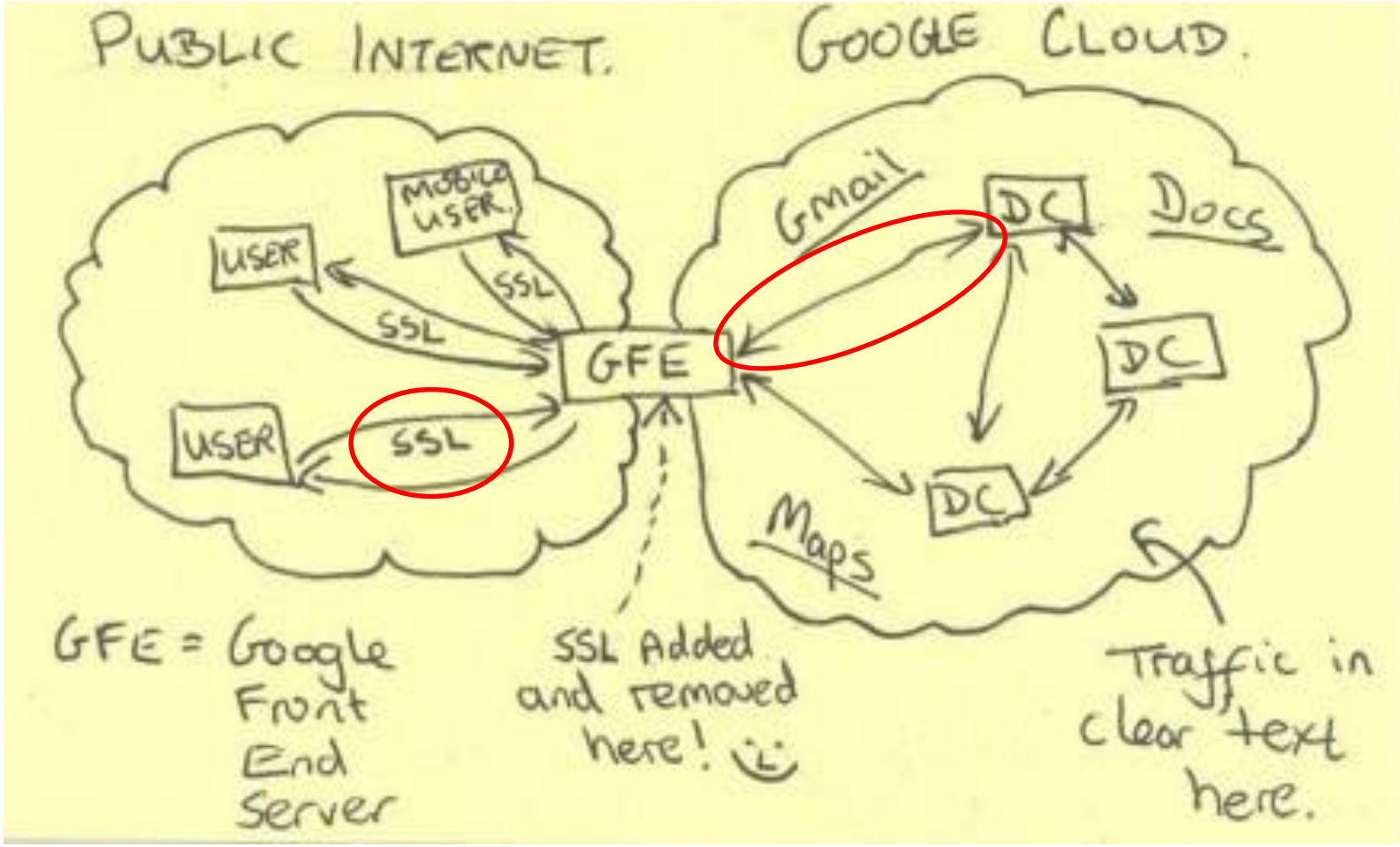
http://pingo.upb.de/

Kapitel 7.5

BEISPIEL: EMAIL-SICHERHEIT

Scheinbare Sicherheit…

- SSL zum Frontend, Klartext im Backend



[WaPo13]

PGP (Pretty Good Privacy)

- Anwendung: Verschlüsselung von EMails
- OpenPGP: durch die IETF standardisiert [RFC4880]]
  - Ähnlich zu S/MIME
  - Nutzung eines Web-of-Trust anstelle „klassischer PKI“

- Wählbare Bestandteile

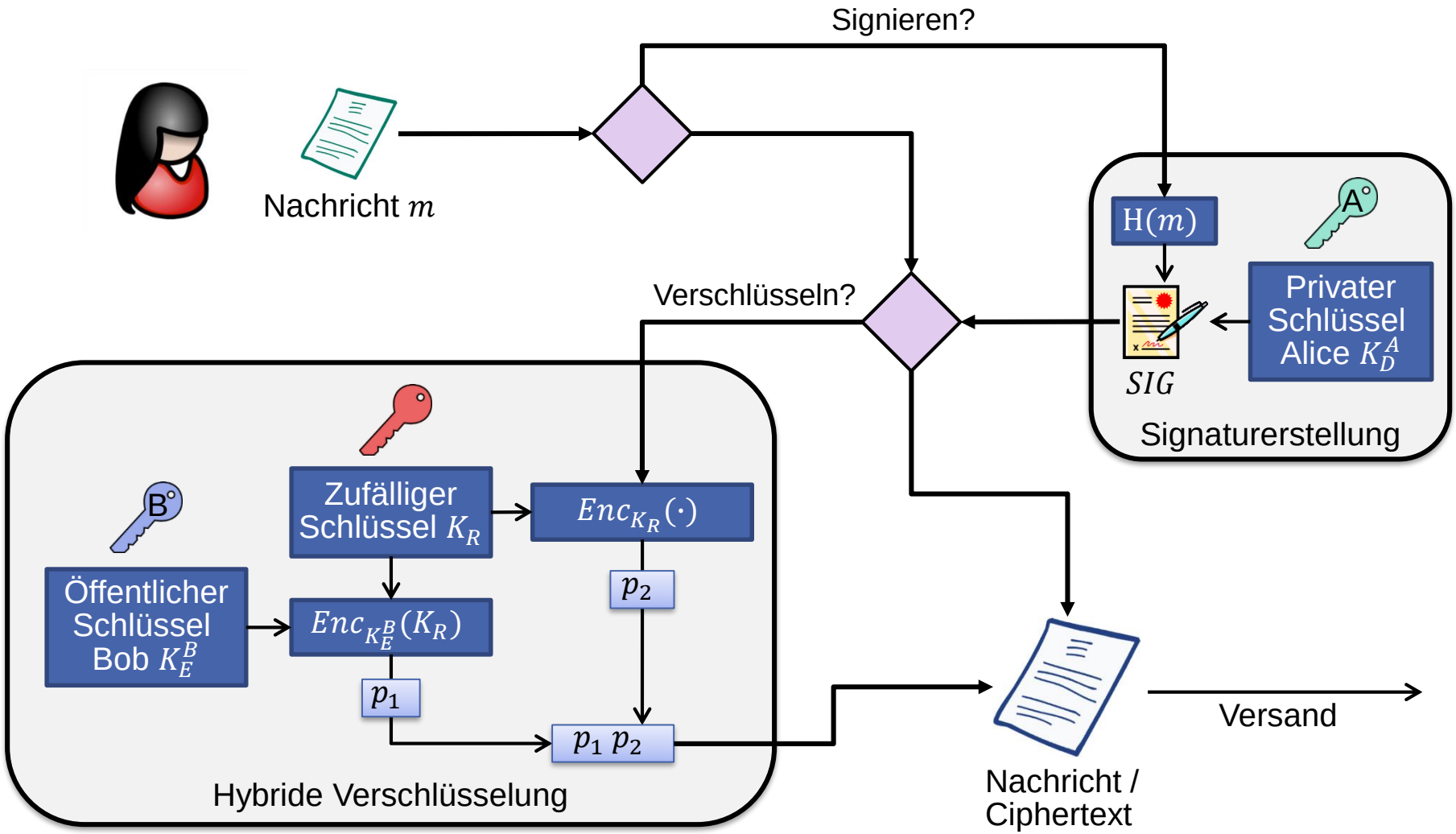
  - Vertraulichkeit und Integrität von Nachrichten
    - Auswahl eines symmetrischen Verfahrens
    - Schlüsselaustausch per asymmetrischen Verfahren
  - Authentizität des Absenders
    - Auswahl eines asymmetrischen Verfahrens
    - Signatur des Absenders mit freiwählbarer Hashfunktion
    - Abbildung zwischen Signatur und Identität muss überprüft werden
  - Authentizität des Empfängers
    - Auswahl eines asymmetrisches Verfahren
    - Durch Verwendung des öffentlichen Schlüssels des Empfängers
    - Abbildung zwischen Identität und Schlüssel muss überprüft werden

Beispiel: Nachrichtenversand von Alice an Bob

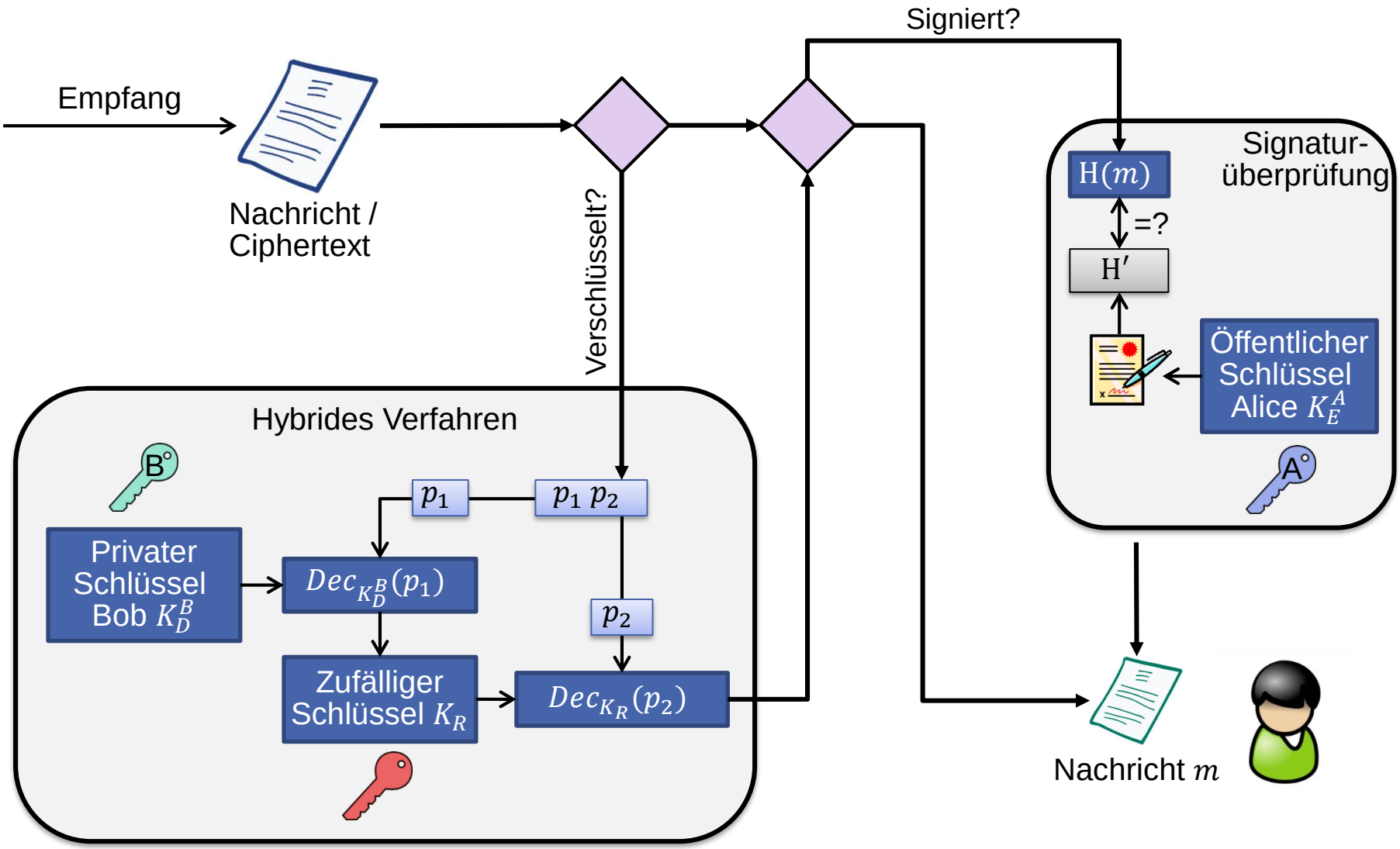
- Schlüsselgenerierung
  - Alice generiert
    - Öffentlichen Schlüssel K_E^A
    - Privaten Schlüssel K_D^A
  - Bob generiert
    - Öffentlichen Schlüssel K_E^B
    - Privaten Schlüssel K_D^B

- Versand der Nachricht
  - Alice
    - Verschlüsselt Nachricht m symmetrisch mit zufälligem Schlüssel K_R
    - Verschlüsselt Schlüssel K_R asymmetrisch mit Bobs öffentlichem Schlüssel K_E^B
    - Sendet Ciphertext der Nachricht und Ciphertext des Schlüssels an Bob
  - Bob
    - Entschlüsselt den Schlüssel K_R asymmetrisch mit privatem Schlüssel K_D^B
    - Entschlüsselt Nachricht m mit symmetrischen Schlüssel K_R

Versand von Nachrichten



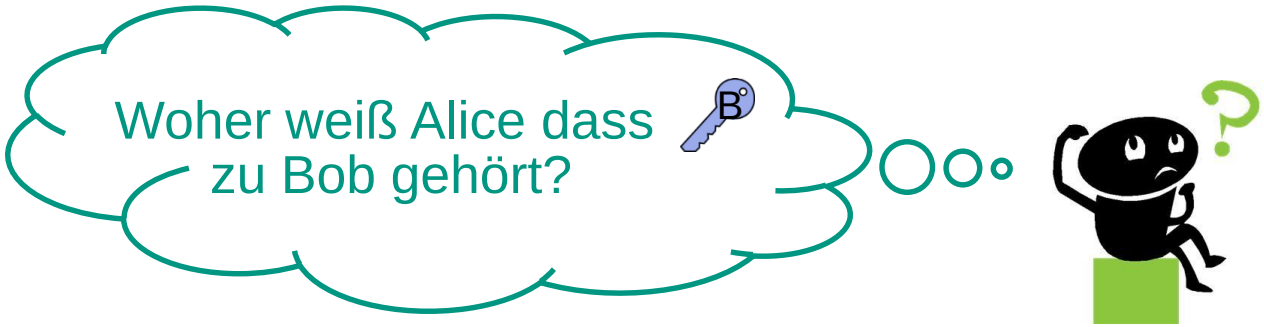
Empfang von Nachrichten



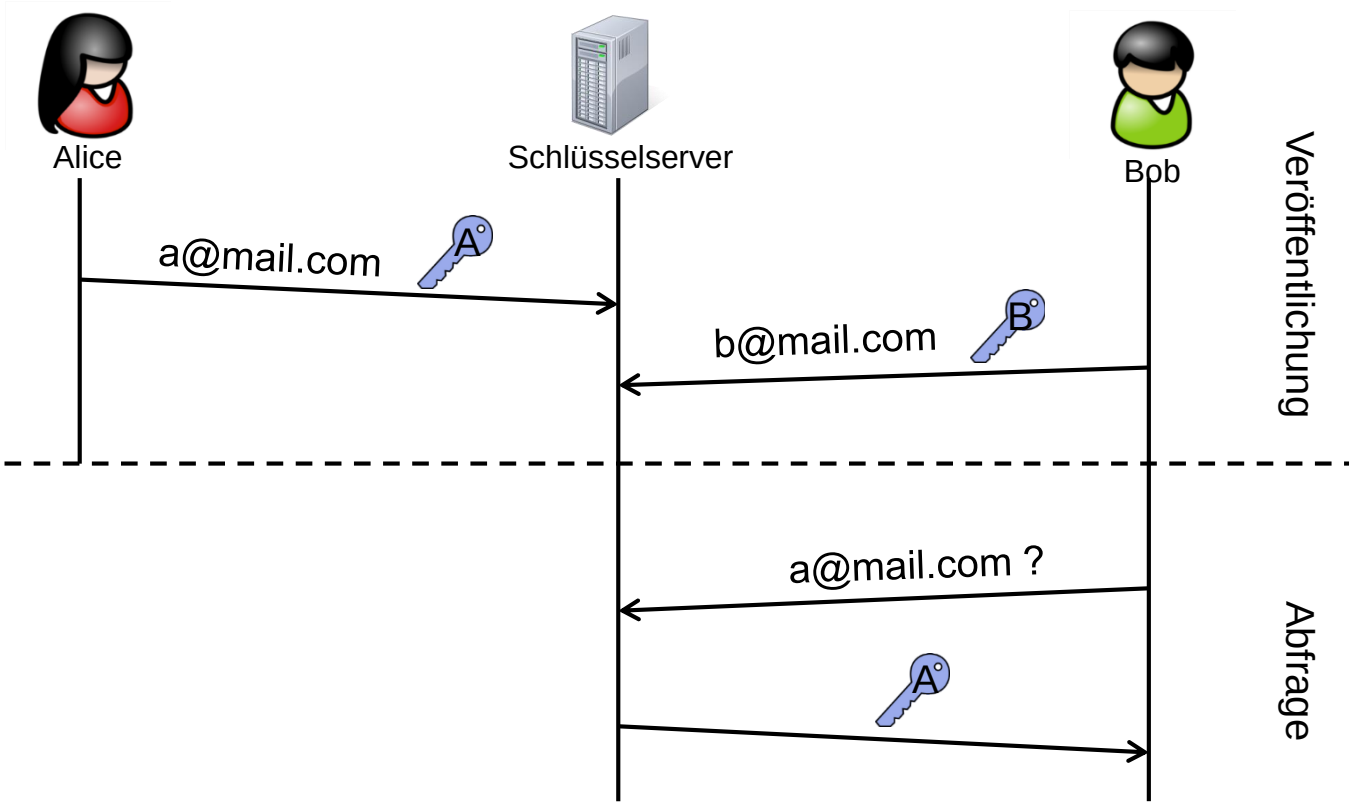
Bekanntgabe öffentlicher Schlüssel

- Verwendung von öffentlichen Schlüsseln
  - Müssen Nutzern vor Versand bekannt sein
  - Zugehörigkeit zu Email-Adresse muss überprüfbar sein
    - Für Authentizität des Absenders
    - Für Authentizität des Empfängers

- Bekanntgabe des zur Email-Adresse gehörenden öffentlichen Schlüssels
  - Persönlich oder per Email
  - Über eine bekannte vertrauenswürdige dritte Partei
  - Eintragung auf wohlbekannten Schlüsselservern
    - z.B. pgp.mit.edu, keyserver.pgp.com, zimmermann.mayfirst.org, …



Öffentliche Schlüsselserver



- Überprüfung der Abbildung Email-Adresse und öffentlicher Schlüssel
  - Per Telefon, vergleich des Hashwertes (Fingerprint) des Schlüssels
  - Durch eine vertrauenswürdige Signatur auf den öffentlichen Schlüssel

Web of Trust (WoT)

- Idee
  - Keine zentrale vertrauenswürdige dritte Partei notwendig
  - Dezentraler anarchischer Ansatz
  - Transitive Überprüfung der Authentizität eines öffentlichen Schlüssels

- Abstrahiertes Beispiel
  - Überprüfen ob öffentlicher Schlüssel K_E^X zu Bob gehört (authentisch ist)
  - Suche in Bekanntenkreis ob jemand bereits K_E^X überprüft hat (Signatur)
  - Akzeptiere K_E^X als authentisch bei bekannten Signaturen

- Aber Achtung: Trennung von
  - Vertrauen in Nutzer und deren Signaturen
  - Authentizität eines öffentlichen Schlüssels

Web of Trust (WoT)

- Grad des Vertrauens in andere Nutzer und in deren Signaturen
  - Owner Trust und daraus abgeleitet Signatory Trust
  - Unbekannt, kein, geringes, volles oder absolutes Vertrauen
- Grad der Authentizität eines öffentlichen Schlüssels
  - Unbekannt, keine, teilweise oder volle Authentizität
  - Wird berechnet
- Vorbereitung
  - Nutzer können andere öffentliche Schlüssel signieren
  - Informationen werden über Schlüsselserver an alle verteilt
- Berechnung der (transitiven) Authentizität (lokal)
  - Nutzer bauen untereinander graduelle Vertrauensbeziehungen auf
  - Nutzer vertrauen Signaturen ihrer vertrauenswürdigen Kontakte
  - Authentizität eines öffentlichen Schlüssel wird berechnet (Key Legitimacy)

WoT – Key Legitimacy

- Nutzer kann einstellen ab wann er öffentlichen Schlüssel als authentisch akzeptiert

- Durch Berechnung der sogenannten Key Legitimacy L
  - $L := \frac{Signaturen\ mit\ geringem\ Vertrauen}{X} + \frac{Signaturen\ mit\ vollem\ Vertrauen}{Y}$
  - Vertrauen in Signaturen direkt abgeleitet von Vertrauen in Nutzer
  - Parameter X und Y als Integerzahl einstellbar

- Bewertung der Key Legitimacy (Authentizität)
  - $L = 0$ überprüfter Schlüssel nicht authentisch
  - $0 < L < 1$ überprüfter Schlüssel teilweise authentisch
  - $L \geq 1$ überprüfter Schlüssel voll authentisch

- Achtung: Vertrauen in eigenen Schlüssel ist absolut
  - Daraus abgeleitet volle Authentizität selbstsignierter Schlüssel

Authentizität eines öffentlichen Schlüssels

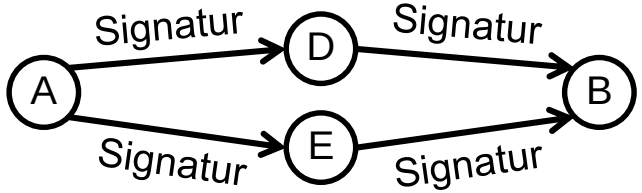
- Problemstellung
  - Alice schickt Nachricht an Bob
    - Verwendet Bob‘s öffentlichen Schlüssel
  - Gehört öffentlicher Schlüssel tatsächlich zu Bob‘s Mailadresse?

- Überprüfung der Authentizität des Schlüssels
  - Alice ruft Bob‘s Schlüssel und alle zugehörigen Signaturen ab
    - Eine Signatur stammt von Carl dem Alice bereits voll vertraut
  - Alice überprüft Carl‘s Signatur mit seinem öffentlichem Schlüssel
    - Alice ist sich natürlich bereits sicher dass Carl‘s Schlüssel authentisch ist
  - Alice ist von der Authentizität von Bob‘s Schlüssel überzeugt

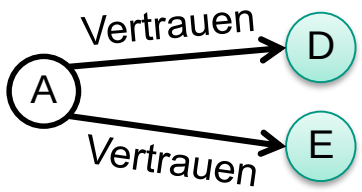
- Transitivität
  - Alice vertraut Carl voll
  - Carl hat Bob‘s Schlüssel signiert
  - Alice akzeptiert daher Bob‘s Schlüssel als voll authentisch

WoT – Beispiel

- Signaturen
  - Nutzer Alice hat den öffentlichen Schlüssel von Dan und Eve signiert
  - Nutzer Dan und Eve haben den öffentlichen Schlüssel von Bob signiert



- Vertrauen
  - Nutzer Alice hat in Dan und Eve geringes Vertrauen



- Von Alice berechnet Key Legitimacy für Bob
  - $L := \frac{2}{2} + \frac{0}{1} = 1$
  - Volle Authentizität von Bob‘s öffentlichem Schlüssel

Anarchie als Lösung für Authentizität?

- Konzept beruht auf Transitivität
  - Wer macht eigentlich mit?
  - Werden Signaturen erstellt und verteilt?
  - Wessen Signaturen vertraue ich wie?
  - Wird die Key Legitimacy überhaupt berechnet?

- In der Realität
  - Entstehen Inseln z.B. Linux-Kernel Developer, IETF-Mailingliste
  - Nutzung ohne Authentizität als Minimalschutz gegen passive Angriffe
  - Und selbst der Erfinder hat praktische Probleme…

WoT – Zusammenfassung

- Kein Vertrauen in zentrale Instanz notwendig
  - Anarchischer Ansatz
  - Öffentliche Schlüssel von Nutzern müssen bekanntgegeben werden
    - Über Schlüsselserver oder direkt
- Statt dessen graduelles Vertrauen in Nutzer: Owner Trust
  - Daraus abgeleitet Vertrauen in Signaturen dieser Nutzer: Signatory Trust
- Überprüfung von öffentlichen Schlüsseln über Signaturketten von vertrauenswürdigen Nutzern
  - Authentizität des Absenders
  - Authentizität des Empfängers
- Authentizität nur so stark wie das schwächste Glied
  - Parameter frei einstellbar, je nach Sicherheitsbedürfnis
  - Vertrauen in andere Nutzer muss wohl bedacht gewählt werden
  - Nutzer können Signaturen nach belieben ausstellen

Kapitel 7.6 – Sicherheit im Protokollstack

ANWENDUNG BEI WHATSAPP

Signal Protocol

- Seit 2014 kommt bei WhatsApp das Signal Protocol zum Einsatz
  - Sehr komplexes Protokoll
  - Entwickelt von Moxie Marlinspike (TextSecure)
  - Nutzung zertifiziert durch externen Audit

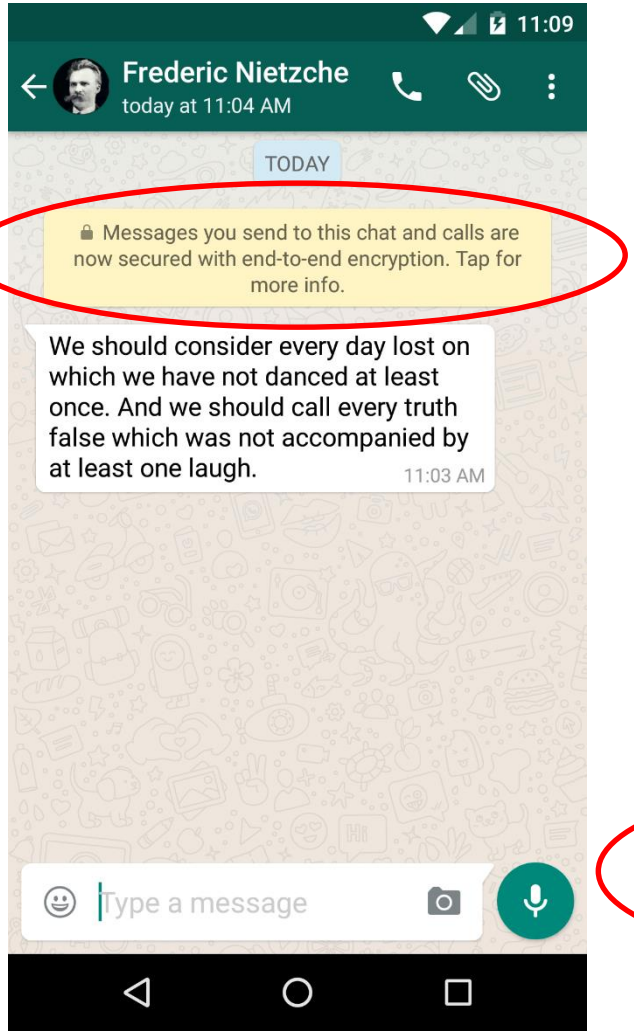
- Bietet Ende-zu-Ende Verschlüsselung für
  - Instant Messaging (auch für Gruppen)
  - Telefonanrufe
  - Videokonferenzen

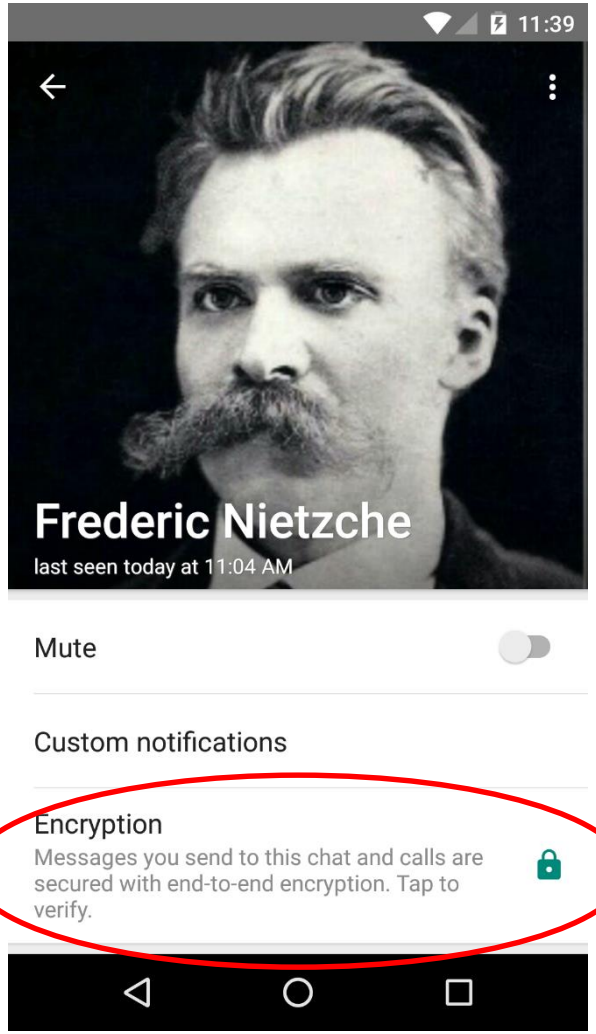
[PeMa16]

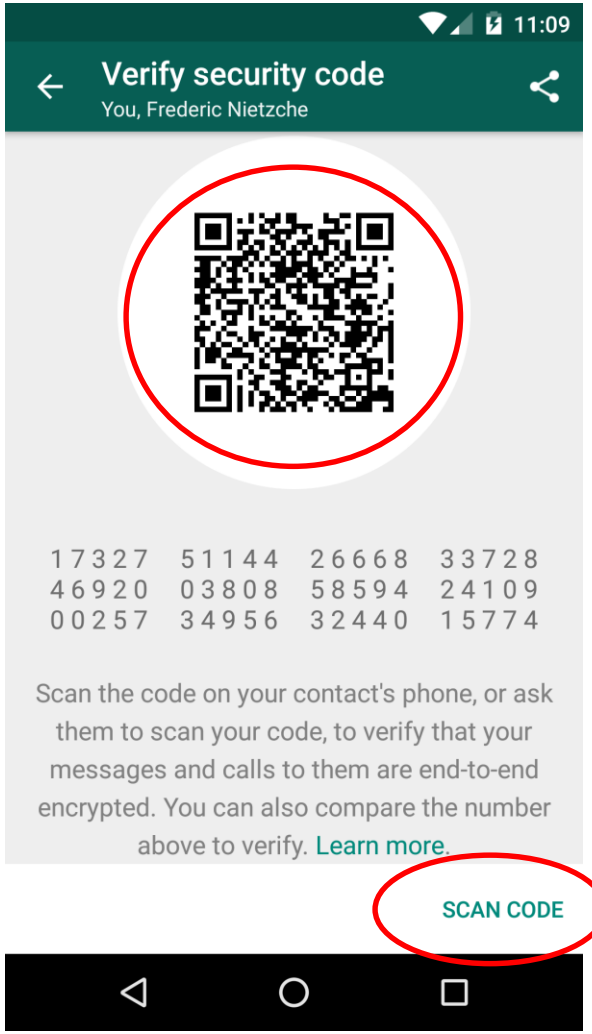
- Sicherheitseigenschaften u.a.
  - Vertraulichkeit, Integrität, Authentizität, Abstreitbarkeit von Nachrichten und der Teilnahme, Forward Secrecy, Unverknüpfbarkeit von Nachrichten

- Übersicht zum Thema Sicherheit und Instant Messaging
  - https://www.eff.org/secure-messaging-scorecard

Überprüfung des Schlüsselmaterials







Kapitel 7.6

INFRASTRUKTUR-SICHERHEIT

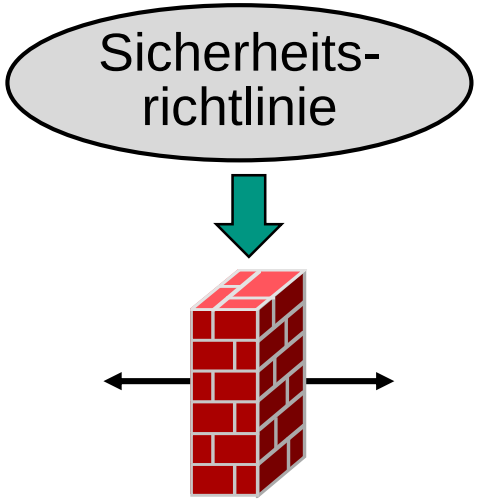
Kapitel 7.5.1 Firewall

Grundlagen

- Zielsetzung
  - Teile des Netzes sollen vor dem Eindringen unerwünschter Pakete geschützt werden
  - Firewalls isolieren hierzu Netzbereiche voneinander
    - z.B. Internet vom internen Unternehmensnetz
    - z.B. Backupnetzwerk vom restlichen Unternehmensnetz

- Analogie zum passiven Brandschutz bei baulichen Anlagen
  - Eine Brandmauer verhindert Übergreifen von Feuer und Rauch
  - Muss bei der Konstruktion mit berücksichtigt werden (vor dem Feuer)

- Firewalls realisieren Zugriffskontrolle
  - Treffen einfache Entscheidungen basierend auf Sicherheitsrichtlinie
  - Paketfilterung (engl. auch Screening)
  - ➔ Paket weiterleiten oder verwerfen



Paketfilterung

- Zustandslose Paketfilterung
  - Filtert Paket für Paket ohne Zustandsinformation
  - Basierend z.B. auf IP Adressen, TCP/UDP Ports, Transportprotokoll, eingehendem Netzinterface

- Zustandsbehaftete Paketfilterung
  - Filtert Pakete auf Basis von Zustandsinformation
  - Überwacht beispielsweise TCP Verbindungsauf- und -abbau und prüft, ob ein- und ausgehende TCP-Segmente zueinander passen
  - Timeout für Zustandsinformation (soft-state)

Beispiel: Zustandslose Paketfilterung

Sicherheitsrichtlinie	Firewall Einstellung
Verhindere TCP-Verbindungsaufbau von außerhalb	Verwerfe alle eingehenden Pakete die TCP-Flag SYN gesetzt haben
Kein Zugriff auf HTTP von innen	Verwerfe alle ausgehenden Pakete mit Zielport 80, unabhängig von der IP-Adresse
Blockiere UDP und Telnet	Verwerfe ein- und ausgehenden Pakete mit IP-Protokoll-Feld 17 und verwerfe alle Pakete, die entweder Quell- oder Zielport auf 23 gesetzt haben
Erlaube von außen nur Zugriff auf den Webserver	Verwerfe alle eingehenden Pakete, die nicht 141.3.71.224 als IP-Zieladresse und nicht Port 80 haben

- Realisierung in der Praxis
  - Konfiguration am Edge Router (Screening Router)
  - Nutzung von z.B. *iptables*
  - Implementierung durch Access Control Lists (ACLs)

Access Control List (ACL)

- Liste von Regeln, die von oben nach unten für alle eingehenden Pakete abgearbeitet wird
  - → Erster „match“ wird angewandt!
- Beispiel: ACL für ein Routerinterface
  - Zustandslose Paketfilterung
  - Hier für Unternehmen mit Adressbereich 222.22/16

Aktion	Quell-IP	Ziel-IP	Protokoll	Quell-Port	Ziel-Port	Flags
Erlauben	222.22/16	Außerhalb 222.22/16	TCP	> 1023	80	any
Erlauben	Außerhalb 222.22/16	222.22/16	TCP	80	> 1023	ACK
Erlauben	222.22/16	Außerhalb 222.22/16	UDP	> 1023	53	---
Erlauben	Außerhalb 222.22/16	222.22/16	UDP	53	> 1023	----
Verwerfen	Alle	Alle	Alle	Alle	Alle	Alle

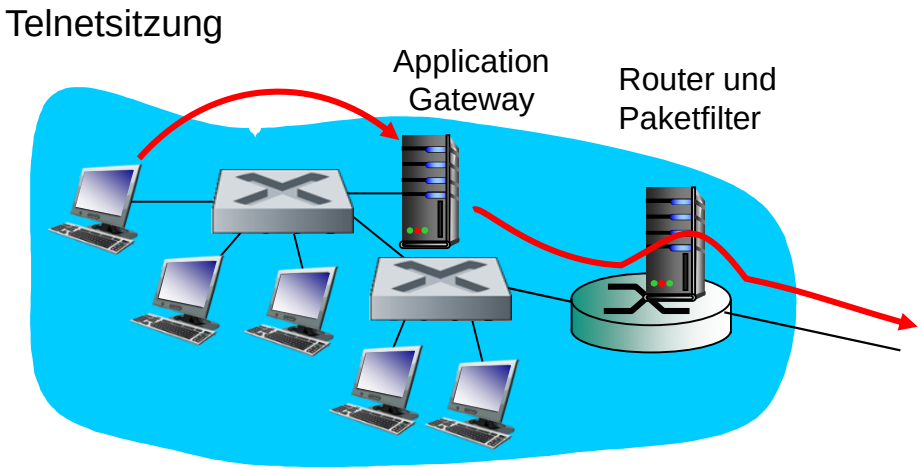
Access Control List

- Erweiterung der ACL für Zustandsüberprüfung
  - Zustand: Information über TCP-Verbindung
  - Timeout, z.B. 60 Sekunden

Aktion	Quell-IP	Ziel-IP	Protokoll	Quell-Port	Ziel-Port	Flags	Zustand
Erlauben	222.22/16	Außerhalb 222.22/16	TCP	> 1023	80	any	
Erlauben	Außerhalb 222.22/16	222.22/16	TCP	80	> 1023	ACK	X
Erlauben	222.22/16	Außerhalb 222.22/16	UDP	> 1023	53	---	
Erlauben	Außerhalb 222.22/16	222.22/16	UDP	53	> 1023	----	X
Ver-werfen	all	all	all	all	all	all	

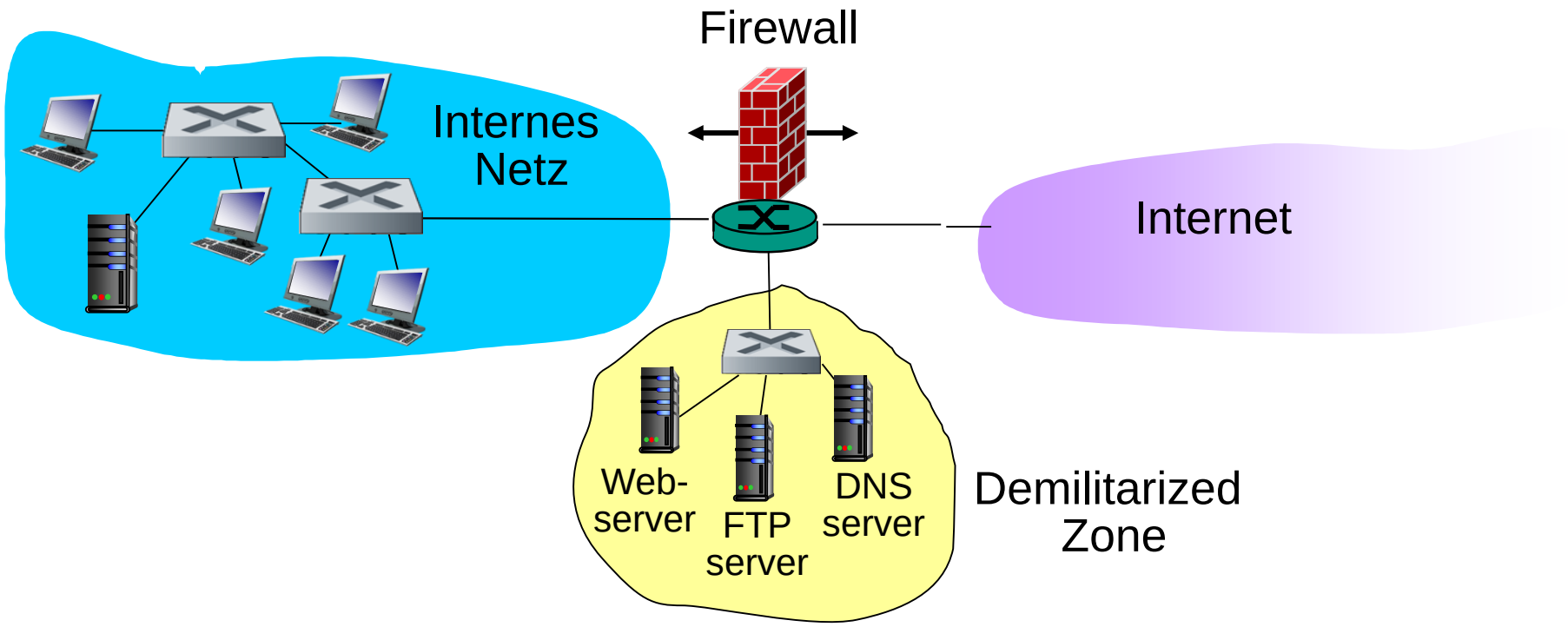
Application Layer Gateway

- Arbeiten typischerweise nicht-transparent
  - Vergleichbar mit einem Proxy
  - Brechen Ende-zu-Ende-Beziehung auf
  - Filtern auf Basis von Nutzdaten
    - z.B. Telnet-Username, d.h. nur bestimmte Nutzer dürfen Telnet nutzen



Demilitarized Zone (DMZ)

- Unternehmensnetze werden unterteilt in
  - Internes Netz: Nur Verbindungen von Innen nach Außen
  - Demilitarized Zone: Erlaubt auch Verbindungen von Außen nach Innen
- Trennung soll Schutz vor Angriffen bieten
  - Bei Übernahme eines Endsystems ist Server durch Firewall geschützt
  - Und umgekehrt



Grenzen von Firewalls

- Absender-Informationen in Paketen können gefälscht sein
  - z.B. IP-Spoofing

➔ Problematisch für darauf basierende Sicherheitsrichtlinien

- Verfügbarkeit
  - Filterung schützt nicht vor Überlastung der Firewall
  - Verbindung zur Firewall kann überlastet sein
  - Single-Point-of-Failure

- Filterung mit Application Layer Gateways sind sehr aufwändig
  - Benötigen viele Ressourcen
  - Sicherheitsrichtlinien können komplex werden
  - Transparenz ist schwierig zu gewährleisten (z.B. bei TLS)

- Firewalls sind nicht 100%ig sicher!
  - Angriffe auch durch legitime Nutzer möglich
  - Firewalls können z.B. durch Hardware-Hintertüren übernommen werden

Kapitel 7.5.2 Intrusion Detection und Prevetion

Erkennen und Verhindern von Angriffen

- Firewalls sind „nur“ gute Filter
  - ➔ Keine Korrelation zwischen verschiedenen Sitzungen

- Intrusion Detection System (IDS)
  - Regelbasierte Korrelation zwischen Sitzungen
  - Zusätzlich Nutzung verschiedenster Sensoren im Netz
    - CPU-Auslastung, Traffic-Informationen, Sitzungen pro Nutzer, etc.
  - Mehrere spezialisierte IDSe können gleichzeitig genutzt werden

➔ Können bekannte Angriffe erkennen

- Intrusion Prevention System (IPS)
  - Arbeitsweise wie IDS
  - Zusätzlich, automatisierte Aktionen bei Angriff
    - z.B. Nutzer vom Netz trennen
    - z.B. Kritische Systeme abschalten
    - z.B. Firewall auf Notfallbetrieb stellen (nur noch Admins kommen durch)

➔ Können bekannte Angriffe verhindern

Techniken zur Angriffserkennung

- Deep Packet Insepction (DPI)
  - Sobald neben Kopffeldern auch Nutzdaten von Paketen betrachtet werden
    - z.B. auch bei transparenten Application Layer Gateways
  - Analyse der Nutzdaten auf bekannte Viren, Trojaner, Angriffsmuster
  - Überprüfung auf Protokollkonformität von Anwendungsprotokollen
    - HTTP, SMTP, IMAP, XMPP

- Anomaly Detection
  - Versucht neben bekannten Angriffen auch noch nicht bekannte zu entdecken
  - Dazu kontinuierliches Monitoring
  - Alarm bei Abweichung vom Normalverhalten
  - Verwendung von Machine Learning
  - Schwierig Normalverhalten zu ermitteln
    - Beispielsweise ist der viele Verkehr ein Angriff, ein iOS-Update oder läuft gerade die Fußball-WM?

Kapitel 7.5.3 Organisatorische Maßnahmen

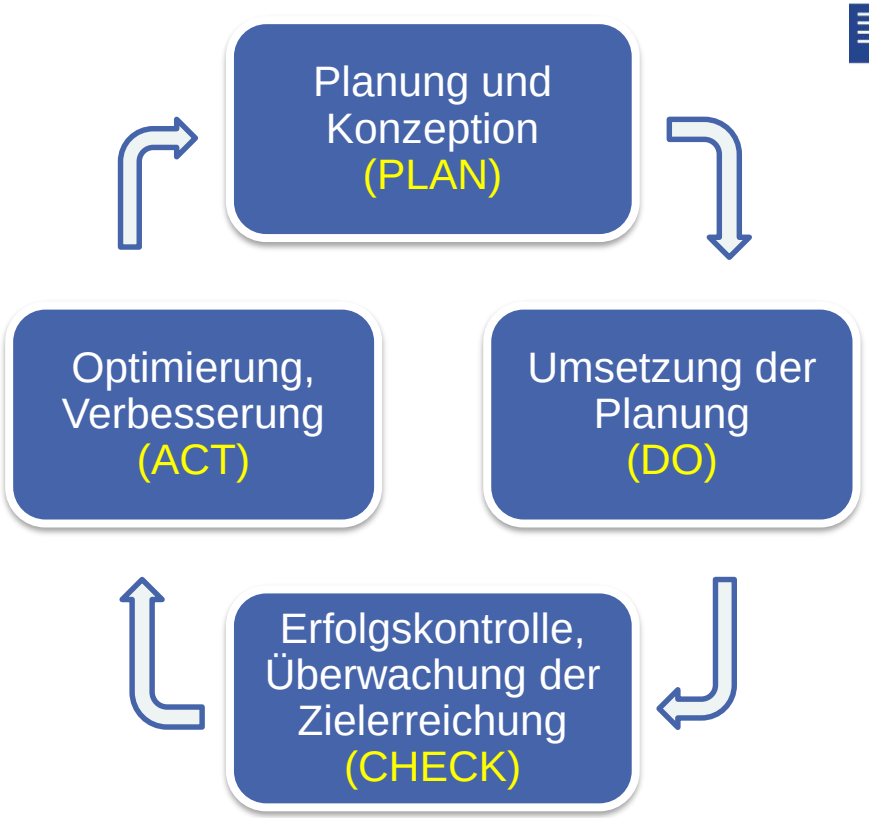
Organisatorische Maßnahmen

- Netzsicherheit lässt sich nicht allein durch Technik erreichen
  - Faktor Mensch
  - Physische Angriffe auf Netze

- Beispiele für Maßnahmen
  - Schulung von Mitarbeitern
  - Festlegung eines IT-Sicherheitsverantwortlichen
  - Erstellung eines Notfallplans
  - Sicherheitsrichtlinien für Mitarbeiter
  - Betrieb von Datensicherung
  - Virenschutzkonzept
  - Dokumentation
  - Vier-Augen-Prinzip
  - Gewährleistung von Aktualität (Updates)

- Zertifizierung der Sicherheitsmaßnahmen (ISO 27001)

IT-Sicherheit als Prozess



[BSI17]

- Beispiele
  - Wechsel des Kryptoverfahrens
    - WEP => WPA2
    - DES => AES
  - Anpassung der Schlüssellänge
    - AES-128 => AES-256
    - RSA-512 => RSA-4096
  - Neue Bedrohungen
    - Heartbleed
    - WannaCry

- Sicherheit ist kein Zustand, sondern ein Prozess!
  - Kontinuierliche Verbesserung
  - Reagieren auf Veränderungen

ZUSAMMENFASSUNG

Abgleich mit Schutzzielen

- Vertraulichkeit
  - Symmetrische Verschlüsselung
  - Asymmetrische Verschlüsselung
- Integrität
  - Message Authentication Codes
  - Digitale Signaturen
  - Zertifikate zur Authentifizierung
- Verfügbarkeit
  - Nicht Bestandteil dieser Grundlagenvorlesung
  - Schwierig zu realisieren
  - ➔ Netzsicherheit: Architekturen und Protokolle

Zusammenfassung

- Schutzziele
  - CIA: Vertraulichkeit, Integrität, Availability (nicht betrachtet) …
- Grundprinzipien der Verschlüsselung
  - Symmetrisch und asymmetrisch
- Integritätssicherung
  - Message Authentication Codes
  - Digitale Signaturen
  - Authentifizierung der Kommunikationspartner
    - Challenge/Response-Verfahren
    - Zertifikate und Certificate Authorities
- Beispiel: Sicherheit bei Email
  - PGP: dezentral, keine vertrauenswürdige dritte Instanz, Web of Trust
- Infrastruktursicherheit
  - Firewalls, Application Level Gateways …
  - Organisatorische Maßnahmen nicht vergessen!
  - Sicherheit ist Prozess

Literatur

[BBCH05] R. Bless, E.-O. Blaß, M. Conrad, H.-J. Hof, K. Kutzner, S. Mink, M. Schöller; Sichere Netzwerkkommunikation: Grundlagen, Protokolle und Architekturen; Springer-Verlag, 2005

[BSI17] Bundesamt für Sicherheit in der Informationstechnik (BSI); IT-Grundschutz, 2.2 Sicherheitsprozess;https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzSchulung/WebkursITGrundschutz/Sicherheitsmanagement/Sicherheitsprozess/sicherheitsprozess_node.html

[DiHe76] W. Diffie, M. E. Hellman; New Directions in Cryptography. In: IEEE Transactions on Information Theory. 22, Nr. 6, 1976, S. 644–654

[DoYa83] Danny Dolev, Andrew Yao; On the security of public key protocols; IEEE Transactions on Information Theory. 29, Nr. 2, 1983, S. 198–208

[KaPS02] C. Kaufman, R. Perlman, M. Speciner; Network Security: Private Communication in a Public World; Prentice Hall, 2002

[KuRo10] J. Kurose, K. Ross; Computer Networking; Addison Wesley, 2010, 5th Edition

- Kapitel 8

[PeMa16] T. Perrin, M. Marlinspike; The Double Ratchet Algorithm; Open Whisper Systems, 2016, https://whispersystems.org/docs/specifications/doubleratchet/

[Paar10] Christof Paar, Jan Pelzl; Understanding Cryptography; Springer, 2010,

- Kapitel 6

Literatur

[RFC2104] H. Krawczyk, M. Bellare, R. Canetti; HMAC: Keyed-Hashing for Message Authentication; RFC 2104, Internet Engineering Task Force, Februar 1997

[RFC4301] S. Kent, K. Seo; Security Architecture for the Internet Protocol; RFC 4301, Internet Engineering Task Force, Dezember 2005

[RFC4880] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, R. Thayer; OpenPGP Message Format; RFC 4880, Internet Engineering Task Force, November 2007

[RFC5246] T. Dierks, E. Rescorla; The Transport Layer Security (TLS) Protocol Version 1.2; RFC 5246, Internet Engineering Task Force, August 2008

[Schm09] K. Schmeh; Kryptografie: Verfahren, Protokolle, Infrastrukturen; Dpunkt-Verlag, 2009

[Schn04] B. Schneier; Secrets & Lies – IT-Sicherheit in einer vernetzten Welt; dpunkt.Verlag, 2004

- Interessante Diskussion zum Thema Sicherheit – keine Präsentation technischer Verfahren

[Sta10a] W. Stallings; Cryptography and Network Security: Principles and Practices; 5. Auflage, Prentice Hall, 2010

[Sta10b] W. Stallings; Network Security Essentials: Applications and Standards; 4. Auflage, Prentice Hall, 2010